MULTI-DIMENSIONAL DIRECT-SEQUENCE SPREAD SPECTRUM MULTIPLE-ACCESS COMMUNICATION WITH ADAPTIVE CHANNEL CODING

By

Estian Malan

Submitted in partial fulfillment of the requirements for the degree

Master of Engineering (Electronic)

in the

Department of Electrical, Electronic and Computer Engineering

of the

Faculty of Engineering, Built Environment and Information Technology

at the

UNIVERSITY OF PRETORIA

Promoter: Professor L.P. Linde

Co-Promoter: Mr. L. Staphorst

November 2006

SUMMARY

MULTI-DIMENSIONAL DIRECT-SEQUENCE SPREAD SPECTRUM MULTIPLE-ACCESS COMMUNICATION WITH ADAPTIVE CHANNEL CODING

by

Estian Malan

Promoter: Professor L.P. Linde

Department of Electrical, Electronic and Computer Engineering

of the

Master of Engineering (Electronic)

During the race towards the *4th generation* (4G) cellular-based digital communication systems, a growth in the demand for high capacity, multi-media capable, improved *Quality-of-Service* (QoS) mobile communication systems have caused the developing mobile communications world to turn towards better *Multiple Access* (MA) techniques, like *Code Division Multiple Access* (CDMA) [5]. The demand for higher throughput and better QoS in future 4G systems have also given rise to a scheme that is becoming ever more popular for use in these so-called 'bandwidth-on-demand' systems. This scheme is known as *adaptive channel coding*, and gives a system the ability to firstly sense changes in conditions, and secondly, to adapt to these changes, exploiting the fact that under good channel conditions, a very simple or even no channel coding scheme can be used for *Forward Error Correction* (FEC). This will ultimately result in better system throughput utilization. One such scheme, known as *incremental redundancy,* is already implemented in the *Enhanced Data Rates for GSM Evolution* (EDGE) standard. This study presents an extensive simulation study of a *Multi-User* (MU), adaptive channel coded *Direct Sequence Spread Spectrum Multiple Access* (DS/SSMA) communication system.

This study firstly presents and utilizes a complex *Base Band* (BB) DS/SSMA transmitter model, aimed at user data diversity [6] in order to realize the MU input data to the system. This transmitter employs sophisticated *double-sideband* (DSB) *Constant-Envelope Linearly Interpolated Root-of-Unity* (CE-LI-RU) filtered *General Chirp-Like* (GCL) sequences [34, 37, 38] to band limit and spread user data. It then utilizes a fully user-definable, complex *Multipath Fading Channel Simulator* (MFCS), first presented by *Staphorst* [3], which is capable of reproducing all of the physical attributes of realistic

mobile fading channels. Next, this study presents a matching DS/SSMA receiver structure that aims to optimally recover user data from the channel, ensuring the achievement of data diversity.

In order to provide the basic channel coding functionality needed by the system of this study, three simple, but well-known channel coding schemes are investigated and employed. These are: binary Hamming (7,4,3) block code, (15,7,5) binary *Bose-Chadhuri-Hocquenghem* (BCH) block code and a rate 1/3 *Non-Systematic* (NS) binary convolutional code [6].

The first step towards the realization of any adaptive channel coded system is the ability to measure channel conditions as fast as possible, without the loss of accuracy or inclusion of known data. In 1965, *Gooding* presented a paper in which he described a technique that measures communication conditions at the receiving end of a system through a device called a *Performance Monitoring Unit* (PMU) [12, 13]. This device accelerates the system's *Bit Error Rate* (BER) to a so-called *Pseudo Error Rate* (PER) through a process known as *threshold modification*. It then uses a simple PER extrapolation algorithm to estimate the system's true BER with moderate accuracy and without the need for known data. This study extends the work of *Gooding* by applying his technique to the DS/SSMA system that utilizes a generic *Soft-Output Viterbi Algorithm* (SOVA) decoder [39] structure for the trellis decoding of the binary linear block codes [3, 41-50], as well as binary convolutional codes mentioned, over realistic MU frequency selective channel conditions. This application will grant the system the ability to sense changes in communication conditions through real-time BER measurement and, ultimately, to adapt to these changes by switching to different channel codes. Because no previous literature exists on this application, this work is considered novel. Extensive simulation results also investigate the linearity of the PER vs. modified threshold relationship for uncoded, as well as all coded cases. These simulations are all done for single, as well as multiple user systems. This study also provides extensive simulation results that investigate the calculation accuracy and speed advantages that Gooding's technique possesses over that of the classic Monte-Carlo technique for BER estimation. These simulations also consider uncoded and coded cases, as well as single and multiple users.

Finally, this study investigates the experimental real-time performance of the fully functional MU, adaptive coded, DS/SSMA communication system over varying channel

conditions. During this part of the study, the channel conditions are varied over time, and the system's adaptation (channel code switching) performance is observed through a real-time observation of the system's estimated BER. This study also extends into cases with multiple system users.

Since the adaptive coded system of this study does not require known data sequences (training sequences), inclusion of Gooding's technique for real-time BER estimation through threshold modification and PER extrapolation in future 4G adaptive systems will enable better Quality-of-Service (QoS) management without sacrificing throughput. Furthermore, this study proves that when Gooding's technique is applied to a coded system with a soft-output, it can be an effective technique for QoS monitoring, and should be considered in 4G systems of the future.

Keywords, phrases and acronyms: 4G, Adaptive channel coding, Bandwidth-on-demand, BB, BER, Binary Hamming (7, 4, 3) block code, Binary BCH block code, CDMA, CE-LI-RU filtered GCL sequences, Channel conditions, Diversity, DSB, DS/SSMA, EDGE, FEC, Incremental redundancy, MFCS, Monte-Carlo technique, MU, PER, PMU, QoS, Rate 1/3 NS binary convolutional code, SOVA, Training sequences.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

LIST OF ABBREVIATIONS

3G	3rd Generation
AWGN	Additive White Gaussian Noise
BB	Base Band
BCH	Bose-Chaudhuri-Hocquenghem
BER	Bit-Error-Rate
CDD	Code Decision Device
CDMA	Code-Division Multiple-Access
CE	Constant Envelope
CE-LI-RU	Constant-Envelope Linearly Interpolated Root-of-Unity
CSI	Channel State Information
CSS	Complex Spreading Sequences
DSB	Double-Sideband
DSSS	Direct-Sequence Spread Spectrum
DT	Discrete-Time
EDGE	Enhanced Data Rates for GSM Evolution
FDMA	Frequency-Division Multiple-Access
FEC	Forward Error Correction
FFCS	Flat Fading Channel Simulator
FHSS	Frequency-Hopped Spread Spectrum
FM	Frequency modulation
GCL	General Chirp-Like
GSM	Global Systems for Mobile Communication
HPC	High-Performance Computing
IIR	Infinite Impulse Response
LOS	Line-of-Sight
MA	Multiple-Access
MAP	Maximum a-Posterior
MD	Multi-Dimensional
MFCS	Multipath Fading Channel Simulator
ML	Maximum-Likelihood
MT	Modified Threshold
MU	Multi-User

NS	Non-Systematic
OFDM	Orthogonal Frequency-Division Multiplexing
OSI	Open Standards Interface
PDF	Probability Density Function
PER	Pseudo-Error-Rate
PHY	Physical
PMU	Performance Monitoring Unit
PSD	Power Spectral Density
QoS	Quality-of-Service
QoSMU	Quality-of-Service Monitoring Unit
QPSK	Quadrature-Phase Shift Keying
SNR	Signal-to-Noise Ratio
SOVA	Soft-Output Viterbi Algorithm
SS	Spread-Spectrum
SSB	Single Sideband
TDMA	Time-Division Multiple-Access
UMTS	Universal Mobile Telephony Service
VA	Viterbi-Algorithm
ZC	Zadoff-Chu

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

During the last ten years, continued growth in the demand for high capacity, multi-media capable, better *Quality-of-Service* (QoS) and ultimately faster wireless mobile communication systems forced the developing wireless communications world to move towards better MA techniques, such as *Code-Division Multiple-Access* (CDMA) and *Orthogonal Frequency-Division Multiplexing* (OFDM). These techniques allow for a further increase in traffic capacity and security. We are also seeing more of the higher-layers of the *Open Standards Interface* (OSI) being merged with lower layers in order to meet more of the demanding communication requirements of today. Certain countries are currently undergoing a metamorphosis toward a *3rd generation* (3G) cellular-based digital communications standard known as the *Universal Mobile Telephony Service* (UMTS) which will ultimately be a seamless broadband communications platform for all types of communication services.

As more of the higher-level layers of the 7-layer OSI model are merged with lower levels, an increase in system *Quality-of-Service* (QoS) is achieved. For example, more powerful *Forward Error Correction* (FEC) schemes are being implemented at the *Physical* (PHY) layer, because of faster processing capability. Higher level layers handle retransmission of data only when degraded channel conditions cause this layer to fail, and data to become corrupt. Because of the demand for better QoS and higher throughput in the race toward 3G, another scheme being implemented at the PHY layer is that of adaptive channel coding. Many existing systems suffer from the fact that the channel codes being used for FEC limits the system to a fixed throughput or data rate. The ability to adapt to changes in channel conditions exploits the fact that, under good channel conditions, very simple, or even no channel coding may be used, enabling an increase in throughput. One such scheme, known as *incremental redundancy*, is already implemented in the *Enhanced Data Rates for GSM Evolution* (EDGE) standard [3]. This study considers a complex wideband multi-user CDMA digital communication system that includes a novel adaptive coding scheme. The goal of this study is to firstly present the structure of the complete system, and then analyze the system’s adaptation performance through a series of simulations using a user-programmable software mobile fading channel simulator.

1.1 RELEVANT RESEARCH TOPICS AND LITERATURE

1.1.1 Quality-of-Service in Digital Communication Systems

Quality-of-Service (QoS) is a term that has become widely popular in all facets of modern-day communication systems. The demand for higher quality communication infrastructures and bandwidth-on-demand for multi-media support is always escalating. This is forcing all system designers to enforce techniques to manage the performance of their communication systems and offer some guarantee that their system will retain a certain level of performance, according to a set of fixed system parameters. This concept is called *QoS provisioning*. There are currently a vast variety of wired and wireless communications networks and because these systems carry many different types of traffic, the determination of QoS parameters are not only becoming system specific, but also traffic dependant. Because of this, the field of QoS has evolved into a far reaching study into the complexities of many different systems and architectures. It has become apparent that QoS parameters are more consumer-driven than provider driven. It has also become apparent within the field of QoS that these parameters can be mainly divided into three QoS groups: Bandwidth, Delay and Reliability.

Bandwidth: The bandwidth of a system is directly linked to the throughput that the system is able to maintain and is a function of the available bandwidth as well as the characteristics of the channel. QoS parameters for bandwidth are influenced extensively by bandwidth hungry applications, such as video streaming, which needs a high throughput to operate properly.

Delay: System delay is the time-delay between system input and output and is linked to system processing time and propagation. For example, large propagation delays can be present in most packet-switching systems because of the routing and processing of packets through the system. Systems incorporating large channel codes can also suffer from unwanted levels of propagation delay, because of the processing involved. Most real-time multi-media applications like video-conferencing and streaming voice communications will suffer under large system latency. Hence, QoS parameters regarding system delay are thus mostly governed by high priority real-time traffic and applications.

Reliability: Reliability is linked to the error-rate performance of a communication system. This means that the system has to have a certain level of error-correcting capability to ensure transfer of information with acceptable error-rate. Systems should thus try to keep the BER-performance below a predetermined level without sacrificing too much bandwidth or delay. The liability to errors in a system comes from the nature of the communication channel being used. It is thus common to assume that wired communication systems suffer less from errors in transmission than wireless systems, because of the unreliable and error-prone nature of the wireless channel. The common solution to this problem is to incorporate powerful error-correcting channel codes but at the expense of larger system delay as well as increased system cost, complexity and bandwidth.

There are of course many trade-offs between the three categories mentioned above that can help a specific system to maintain a certain level of QoS. Because future systems will carry such diverse traffic types and have to cater for many user needs under random conditions, systems will have to be able to adapt to these changing environments and needs. From there the term: ‘bandwidth-on-demand’. By incorporating an adaptive channel coding scheme, a system is able to take advantage of good channel conditions by switching to less powerful and less redundant channel codes, thus delivering an increase in throughput. The focus of this study is on the trade-off between throughput and reliability by monitoring the BER performance of a system and attempting to keep this parameter under a predetermined value over varying channel conditions. It should finally be noted that because QoS is such a diverse subject, only superficial details of QoS methodologies will be given in this study.

1.1.2 Real-Time BER Measurement

In any adaptive system the need exists for the system to sense when adaptation is required. One such way to achieve this is through some subsystem that monitors the conditions in real-time and then generates a ‘trigger’ signal whenever the changes in these conditions are large enough. This subsystem then has to feed back this signal to the main system to instruct it to adapt, after which the process is repeated until a certain QoS parameter is satisfied. In older literature that deals with this feature, these types of subsystems have been dubbed *Performance Monitoring Units* (PMUs) [12, 13]. In the light of the previous sections, a more modern approach to these subsystems would be *Quality-of-Service Monitoring Units* (QoSMUs), because of their ability to monitor a predetermined QoS system parameter. In a wireless communication system, typical conditions being monitored by these QoSMUs are that of the channel. The speed and accuracy at which these QoSMUs are able to estimate these parameters is of course, of utmost importance. The tradeoff between speed and accuracy is one of the most important tradeoffs in any adaptive system.

There are currently many techniques with which these QoSMUs determine these parameters. In some systems these QoSMUs include known sequences, referred to as *preamble sequences* inside data packets that the receiver uses to determine channel conditions [9]. Other systems interrupt main data flow to send known data sequences in order to determine channel statistics, after which the system returns to normal. These subsystems are all acceptable, but lack in speed because of the inclusion of known data in order to estimate their parameters, and consequently forfeits throughput.

It would, of course, be ideal if these QoSMUs place no extra burden on the system by subjectively monitoring the wanted parameters accurately without the need for known data or interruption of traffic. Some of the current techniques include estimating the *Signal-to-Noise Ratio* (SNR) of the channel by analyzing the statistics of the received signal envelope or eye pattern [10]. Other systems induce additional noise parameters at the receiver in order to raise the probability of error, which in turn raises the BER at the receiver [11]. The QoSMU then counts these extra errors as *Pseudo-Errors* and determines a *Pseudo-Error-Rate* (PER). From this measurement, an estimate of the BER is then calculated by using some predetermined algorithm. Raising the probability of error causes these ‘pseudo-errors’ to be generated more often, and enables a faster estimation of the BER. This enables an increase in the speed at which these types of QoSMUs monitor the BER QoS parameter, unfortunately at the cost of reduced accuracy. In 1965, *Gooding* [12, 13] presented a paper in which he described one such technique. Many researchers have since extended his work and have published many papers on the application of this technique for different modulation and pre-coding schemes [14-19]. None of the authors have, however, tested this technique on a fully coded, wideband CDMA system under more realistic frequency-selective fading conditions. It is the goal of this study to firstly review and analyze the details of this technique under classic *Additive White Gaussian Noise* (AWGN) channel conditions. The performance of this technique will then be analyzed through extensive simulations on a wideband multi-dimensional CDMA system over a more realistic frequency-selective fading channel simulator. This will be done for both coded and uncoded circumstances. Finally, the technique will be incorporated into a fully functional, adaptively coded communication system, where it will form the core QoSMU that measures the BER and triggers adaptations when needed.

1.2 MOTIVATIONS FOR THIS STUDY

Since the inception of the bandwidth-on-demand concept, wireless communication system designers have started to realize that future communication applications will cause an extreme diversity of traffic characteristics over communication systems. It will become more and more complex to ensure user satisfaction and, therefore, be more difficult to maintain high levels of QoS. This fact has lead to the primary motivations for this study:

1. Current 2G systems have the shortcoming that they are mostly bound to a fixed throughput under all channel conditions, because of the incorporation of fixed-length channel codes. Some of the advanced 2.5G systems of today, such as EDGE already utilize advanced adaptive coding and modulation systems like the *incremental redundancy* scheme in order to switch between different FEC codes and modulation techniques, resulting in large improvements in throughput. The main driving force behind the need for adaptive systems includes the following: Taking advantage of the time-varying nature of wireless channels and through the adaptation of channel codes, improve on throughput. A need to explore and research new and more practical ways to realize adaptive channel coded systems has been sparked.

2. Employing a multitude of channel codes in an adaptive communication system requires excessive processing power. Work by *Staphorst* [3] suggests a generic ML decoder structure that can be used for all types of channel codes, including convolutional and block codes. This enables a much more practical and possibly

less expensive way in which a variety of codes can be implemented to realize an adaptive coded system.

Secondary motivations for this study, emerging from unexplored areas, as well as areas for improvement within the adaptive channel coding field, are:

1. The real-time BER performance measurement of a system is a very attractive technique for measuring QoS conditions, because of the fact that all system non-idealities are taken into account when measuring BER at the receiving end of a system. This includes all system imperfections as well as all of the unavoidable channel perturbations. This forms the main motivational factor behind the exploration of real-time BER measurement techniques for use in an adaptive system.

2. The most important part of any adaptive channel coding system is the subsystem that measures the QoS parameter needed for channel code switching. These subsystems have to be able to measure this parameter as fast as possible, without sacrificing too much accuracy. They should also be simple to implement, without placing a burden on system throughput. *Gooding's* 1965 [12] real-time BER estimation technique proves to be a very promising technique to use in these subsystems.

3. Because of its nature, the previously mentioned QoSMU operates with a soft-valued input for proper functionality. It is therefore essential in a fully coded system, that the channel decoder being implemented provides soft-output values. The well known Soft-Output Viterbi Algorithm (SOVA), proposed by *Hagenauer* and *Hoeher* in [39], will be utilized to realize this requirement. *Gooding's* pseudo-error generation and extrapolation techniques to systems that incorporate the SOVA has not been investigated or applied by any previous authors. The research done in this paper includes introductory investigations into the performance of the pseudo-error extrapolation techniques in a fully coded system that incorporates the SOVA for 3 different types of channel codes.

4. DS/SSMA systems presented in modern literature are mostly evaluated under non-realistic channel conditions. These systems also employ low-capacity spreading sequences, which are impractical in the emerging world of higher capacity wireless communication systems. Thus, these systems need to be evaluated under more realistic multi-user multipath or frequency-selective fading channel conditions, as well as employ larger capacity spreading sequences in order to realize their full potential. The powerful user-programmable frequency-selective fading channel model derived by *Staphorst* [3] and the proposed MD DS/SSMA modem provide very attractive platforms on which to evaluate such a system.

5. Simulating communication systems which include the realistic and complex techniques mentioned above is an extremely processing-intensive and time-consuming process. Stable, fast and reliable simulation platforms and systems are expensive and therefore scarce. The *University of Pretoria's I-percube High-Performance Computing* (HPC) *Cluster* (donated by Intel) was used to realize a software-based simulation platform that reduces simulation time by distributing the computational load of simulations over 16 processors.

1.3 OBJECTIVES FOR THIS STUDY

The primary objective of this study was the design of an adaptive channel coded communication system employing sophisticated MD DS/SSMA and real-time BER measurement techniques, as well as the performance evaluation of this system in a multi-user multipath fading channel environment. Secondary objectives stemming from this study are:

1. Study and evaluate a sophisticated MD DS/SSMA spreading technique and incorporate this technique into a QPSK-based communication system (see *Chapter* 2).
2. Study an existing sophisticated multi-user multipath fading channel model and incorporate this model into the system (see *Chapter* 2).
3. Study and revise a well-known real-time BER measurement technique under AWGN channel conditions and incorporate this technique into the system (see *Chapter* 3).

4. Study three powerful FEC codes and incorporate these codes into a fully functional wideband adaptive communication system (see *Chapter* 4).
5. Study and revise the well-known SOVA (see *Chapter 4*).
6. The design of an adaptive coding scheme, incorporating the real-time BER measurement technique of Chapter 3, the FEC codes of Chapter 2, as well as the SOVA decoder into the MD DS/SSMA QPSK-based communication system (see *Chapter* 4).
7. Explain and illustrate the different simulation setups and parameters for the performance evaluation of the different subsystems, as well as the complete communication system. This includes the following (see *Chapter* 5):

    a) Explanation of the simulation setup and parameters for the evaluation of the time and spectral characteristics of the proposed MD/DSSS transmitter.
    b) Explanation of the simulation setup and parameters for the evaluation of the spectral and temporal characteristics of the complex channel.
    c) Explanation of the simulation setup and parameters for the evaluation of the uncoded and coded BER performance of the fully functional wideband communication system.
    d) Explanation of the simulation setup and parameters for the evaluation of the real-time BER measurement technique under multi-user multipath fading channel conditions. This includes PER vs. threshold characteristics, accuracy performance and speed performance.
    e) Explanation of the simulation setup and parameters of the real-time adaptive coding simulations.

8. Evaluation of the results corresponding to the simulation setups of Chapter 5 (see *Chapter 6*)

1.4 NOVEL CONTRIBUTIONS MADE IN THIS STUDY

This study has made generous contributions to the field of wireless communications, with the largest stemming from MD DSSS/MA communication, real-time BER estimation, as well as adaptive coding. Excluding this chapter, as well as the concluding chapter of this dissertation, each chapter ends with a subsection that highlights the novel contributions made by that particular chapter. The following summarizes the most prolific contributions made by this study:

1. Contributions made in the field of MD DSSS/MA communication are:
   a) A complex MD DS/SSMA transmitter structure that improves spectral efficiency and throughput was proposed (see Chapter 2, section 2.2).
   b) A matching receiver structure was developed (see Chapter 2, section 2.4).

2. Contributions made in the field of real-time BER estimation are:
   a) A proper derivation of the general expression for the quadratic PE-extrapolation technique was presented (see Chapter 3, section 3.5).
   b) A unique standard for evaluating the speed performance of a PE-extrapolation technique was proposed (see Chapter 3, section 3.6.2).
   c) *Gooding's* extrapolation technique [12, 13] was extensively tested under proper frequency selective fading conditions (i.e. over a proper MU-MFCS).
   d) *Gooding's* extrapolation technique [12, 13] was tested on a coded system that employs a fully functional SOVA [39].

3. Contributions made in the field of adaptive channel coding are:
   a) The principles of the SOVA by *Hagenauer* [39] was applied to the VA decoding of linear block codes by *Staphorst* [3] (see Chapter 4, section 4.3.3).
   b) The structure of a Code Decision Device (CDD) algorithm that will ensure proper switching between codes, as well as optimal system performance under varying channel conditions, was presented (see Chapter 4, section 4.4).
   c) The evaluation of the accuracy and speed performance of a fully adaptive coded, wideband system, employing the SOVA, was presented (see Chapter 6, section 6.6).

1.5 ORGANIZATION OF THE DISSERTATION

This dissertation consists of seven chapters. The content of these chapters can be summarized as follows: Chapter 1 consists of a historical overview of the revolution towards wireless communications. It then presents and shortly discusses the most relevant research topics of this study, followed by the motivations and main objectives of this study. Lastly, this chapter presents a summary of the novel contributions emanating from this study. Chapter 2 focuses on the structure of a MD DSSS/MA communication system. It starts off with the proposed MD DSSS/MA transmitter structure and is followed by a complete structure and explanation of a user-definable complex multi-user multipath fading channel, as presented by *Staphorst* [3]. Lastly, this chapter presents a matching receiver structure for the transmitter. The focus of Chapter 3 is that of real-time BER estimation and the application thereof to the complete wideband system of Chapter 2. This chapter firstly presents the structure of a QoSMU, as first proposed by *Gooding* [12]. This is followed by a review of the PER vs. threshold characteristics on AWGN conditions. The following two sections review the theoretical principles of the three PER-extrapolation techniques investigated in this study, i.e. classic linear, improved linear and quadratic extrapolation. The last two sections present the principles and standards for evaluating the accuracy and speed performance of these extrapolation techniques and then evaluates these performances for all three extrapolation techniques under AWGN conditions. Chapter 4 focuses on the principles of channel coding, channel codes and the principles for the application of these codes to the fully adaptive channel coded system. The first section presents the three relevant channel codes of this study, i.e. binary Hamming (7,4,3) block code, (15,7,5) binary *Bose-Chadhuri-Hocquenghem* (BCH) block code and a rate 1/3 *Non-Systematic* (NS) binary convolutional code and briefly discusses the most important details of these codes. The following section shortly reviews and discusses the most important details of the generic block code ML decoder as proposed by *Staphorst* in [3], and uses a graphical decoding example of the binary Hamming (7,4,3) block code to illustrate its operation. This section also contains a very important subsection that applies the principles of the SOVA to the generic ML decoder of *Staphorst* [3] by using an extended graphical decoding example, and also discusses the importance and application of the SOVA to the fully adaptive channel coded system. This chapter is concluded with a section that explains a code decision device (CDD) algorithm that will ensure proper switching between codes, as well as optimal system performance under varying channel conditions. Chapter 5 presents the setup configurations, as well as all relevant system parameters for the simulation and evaluation of the different subsections of the system. The first section discusses the setups and parameters for evaluating the time and spectral characteristics of the proposed MD DSSS/MA transmitter structure of Chapter 2. This is followed by a section containing three subsections that discusses all setups and parameters for evaluation of the spectral characteristics of the proposed complex *Multi-User Multipath Fading Channel Simulator* (MU-MFCS) of Chapter 2. The following section explains the setups and parameters for the simulation of the uncoded and coded BER performance of the coded wideband system over the proposed MU-MFCS. The next section discusses the setups and parameters for evaluating the PER vs. threshold characteristics of the three PE-extrapolation techniques of this study over the MU-MFCS. Then, the second last section presents the setups and parameters for the evaluation of the accuracy and speed performance of these three extrapolation techniques. The last section presents all additional setups and parameters for the real-time simulation of the fully functional, adaptively coded wideband system over frequency-selective (MU-MFCS) conditions. Chapter 6 is dedicated to the presentation, discussion and evaluation of the simulation results that correspond to the setups of Chapter 5. The first section presents the simulation results for the time and spectral characteristics of the proposed MD DSSS/MA transmitter structure of Chapter 2, and discusses these findings. This is followed by a section containing three subsections which presents and discusses simulation results of the spectral characteristics of the complex MU-MFCS. The followed section presents and discusses the simulations results of the uncoded and coded BER performances of the fully functional, fully coded wideband system. The next section presents and explains the simulation results of the PER vs. threshold characteristics of the three PE-extrapolation techniques of this study over MU-MFCS. Then, the second last section presents and discusses the results of the accuracy and speed performance of these three extrapolation techniques. The last section presents and discusses the simulation results for the real-time simulation of the fully functional, adaptively coded wideband system over frequency-selective (MU-MFCS) conditions. In Chapter 7 conclusions are drawn from the results of this study and also propose opportunities for future research stemming from this study.

This dissertation contains three appendixes that explains and discusses additional information needed for the understanding of the subject matter of this study. The contents of the appendixes can be summarized as follows: *Appendix A* focuses on the realization of the Doppler spread spectral shaping filters, as explained by *Staphorst* in [3]. The first section shortly discusses and presents the principles of Doppler spreading and is followed by a section that explains the practical realization and trade-offs of a filter structure that will attempt to realize these principles. *Appendix B* focuses on the principles and generation of the complex spreading sequences, employed by this study, i.e. DSB CE-LI-RU filtered GCL complex spreading sequences. The first section explains the generation of the unfiltered ZC sequences. This is followed by a section that explains the adaptation to these ZC sequences to create the DSB CE-LI-RU GCL complex spreading sequences. The last appendix, *Appendix C* supplies the reader with an index of the developed C++ classes used in the simulations of this study.

CHAPTER TWO

MD DS/SSMA SYSTEM

2.1 OVERVIEW

2.1.1 Direct-Sequence Spread Spectrum Multiple-Access

Because of the increased frequency re-use capacity that CDMA techniques have over current 2G TDMA-based systems, it has become very popular and an indispensable technique for implementations of current 3G and future 4G cellular systems [4]. A few of the current commercial wireless cellular systems that employ CDMA techniques are IS-95 [5], 3G UMTS and cdma2000. *Spread-Spectrum* (SS), which forms the underlying principle of all CDMA-based communication systems, has been exhaustively researched since its inception in the military community after the Second World War. It was originally employed for its security and ant-jamming properties, but it was soon discovered that it had many more promising properties that could be exploited commercially.

SS techniques are mainly divided into two categories: *Direct-Sequence Spread Spectrum* (DSSS) and *Frequency-Hopped Spread Spectrum* (FHSS) [6]. The technique relevant to this study is DSSS. With this technique, each user in a communication system is assigned a unique user code that is modulated onto a common carrier for transmission, causing the bandwidth of the signal to increase. Hence the term: *wideband communication*. At the receiver, each user's data can then be successfully extracted through the use of the corresponding code by extracting it from the carrier through a demodulation and de-spreading process. This allows a common carrier frequency to be used simultaneously by many users. For successful de-spreading of user's data, the codes being used ultimately need to have low cross-correlation properties and, simultaneously, strong autocorrelation properties. The maximum number of possible users within a set of codes using the same carrier frequency is also a function of these properties. Ideally, if all the codes within a code set are mutually orthogonal, all the codes can effectively be used. Generation of orthogonal or low cross-correlation sequences forms part of a notable proportion of research in the CDMA field. Some of the first binary sequences discovered include: semiorthogonal *Gold* and *Kasami* sequences and orthogonal *Walsh* sequences. One problem with these codes is that their auto- and cross-correlation properties compared to their family size make them, in many cases, unsuitable for use. This has caused the world of SS systems to look more toward the use of non-binary, or *Complex Spreading Sequences* (CSS), which, apart from their superior correlation properties when compared to binary sequences, enable the generation of *Constant Envelope* (CE) as well as *Single Sideband* (SSB) signals.

The CSSs used in this study is known as the family of *double-sideband* (DSB) *Constant-Envelope Linearly Interpolated Root-of-Unity* (CE-LI-RU) filtered *General Chirp-Like* (GCL) sequences [3]. This study will also use a *Multi-Dimensional* (MD) DSSS-based *Quadrature-Phase Shift Keying* (QPSK) scheme that utilize these CE-LI-RU CSSs to create a wideband communication system platform for the evaluation of the adaptive coding scheme in this study.

2.1.2 Realistic Mobile Channel Models and Simulation

The degrading effects of non-ideal communication channels cause transmitted data to become unreliable and possibly unusable as a result of irregularities of the propagation medium and other system impairments. Shannon's breakthrough research in 1948 showed that we are able to overcome these effects through the use of powerful error-correcting codes, even in the presence of noise.

There are mainly five types of perturbations that cause potential corruption of data in a channel: *AWGN*, *signal reflection, signal scattering, signal diffraction* and *Doppler spreading.*

AWGN: This phenomenon is mainly caused by thermal noise generated by the electronic components inside a receiver and is thus added to the information-bearing signal at the receiver. Some environmental and galactic phenomenon can also be sources of noise. The combined *probability density function* (PDF) of these noise sources has a Gaussian distribution, whilst the *power spectral density* (PSD) is characteristically flat [20].

Reflection, Scattering and Diffraction: When a transmitted signal propagates through free-space, surrounding objects allow the signal to diffract, scatter or reflect, causing multiple versions of the signal propagated towards the receiver. These multiple versions arrive at the receiver at different times and through constructive and destructive interference at the receiver, cause the final observed signal's envelope and phase to vary stochastically [7].

Doppler spreading: When the surrounding objects in an environment are in motion, reflection of transmitted waves from the surrounding objects, or the relative motion between transmitter and receiver cause a shift in the frequency of the transmitted signal and ultimately a 'smearing' effect on the spectral characteristics of the signal [7], causing the signal to occupy more bandwidth than usual, adding to the variations in the envelope and phase of the received signal.

The latter two phenomena mentioned above combine to form the well-known effect of received signal fading. Signal fading can be divided into four main categories: *Fast Fading, Slow Fading, Flat Fading* and *Frequency-Selective Fading* [7].

Fast Fading: This is caused when the rate of change of the channel is much larger than that of the transmitted signal, i.e. the impulse response of the channel changes rapidly within the transmitted symbol duration. This causes Doppler spreading, which has the effect of signal distortion. From a spectral perspective, a signal undergoes fast fading when the symbol duration is larger than the coherence time of the channel and when the bandwidth of the transmitted signal is smaller than the Doppler spread of the channel [7].

Slow Fading: In contrast to fast fading, this fading occurs when the symbol duration is much smaller than the coherence time of the channel and when the bandwidth of the transmitted signal is much larger than the Doppler spread of the channel, i.e. when the rate of change of the channel is much smaller than that of the transmitted signal [7].

It should be noted that the two phenomenons above occur because of the velocity of moving objects in the channel and the baseband signaling rate.

Flat Fading: This type of fading occurs when the bandwidth of the signal is much smaller than the coherence bandwidth of the channel. It is the most common type of fading used in literature, because of its easily modeled fading characteristics. The PDF of the received signal envelope is characterized by a Rayleigh distribution [6].

Frequency Selective Fading: In contrast to flat fading, this fading occurs when the coherence bandwidth of the channel is much smaller than the bandwidth of the transmitted signal. The main cause for this type of fading is the multipath dispersion of the transmitted signal through the channel. When this occurs, the received signal includes multiple versions of the transmitted signal, which are attenuated and delayed in time, and cause the received signal to be distorted or faded [7].

The modeling and accurate simulation of a realistic transmission channel is an indispensable tool in the analysis of wireless digital communication systems. Being able to artificially create realistic channel environments to be used as test beds for sophisticated communication systems has been the driving force behind countless hours of research on channel modeling. Realistic models have to be able to produce all of the above mentioned fading phenomenon in order to be authentic, let alone accurate. Many effective and well-known models have been proposed, including the *Clarke and Gans Flat Fading Model* [7], *Saleh and Valenzuela Indoor Statistical Model*, *SIRCHIM and SMRCIM Indoor and Outdoor Statistical Models* [8], and many more. The most difficult fading channel environment to model is that of frequency selective fading, since each one of the multipath signals must be modeled separately and the channel considered a linear filter. Many of the channel parameters that channel simulators are configured with, originate from measurements taken in realistic wideband communication environments.

The adaptive coding scheme of this study will be tested on a realistic frequency-selective fading channel, using a sophisticated MD/DSSS QPSK-based communication system. The channel model used in this study is based on the work by *Staphorst* [3]. This model was initially derived from the well-known *Clarke and Gans Flat Fading Channel Model* [7] and was expanded and adapted to create a sophisticated complex model of a realistic frequency-selective fading channel.

This chapter considers a short mathematical overview of each of the functional units of the uncoded MD DS/SSMA QPSK-based communication system used in this study. First, a mathematical description of the complex multi-dimensional DS/SSMA QPSK-based transmitter structure will be presented. Secondly, a short mathematical overview of the complex implementation of a *flat fading channel simulator* (FFCS) as proposed by *Staphorst* [3], will be given. This channel will then be expanded to a complex *multipath* *fading channel simulator* (MFCS) and discussed. The channel will then be further expanded into a *multi-user multipath fading channel simulator* (MU-MFCS) and discussed. A short discussion on power delay profiles for frequency selective fading channels follows this section. Lastly, the structure for the matching complex DS/SSMA QPSK-based receiver used will be presented and discussed.

Fig. 2.1 shows a generalized and simplified illustration of the composition of the different functional units that the uncoded MD DS/SSMA digital communication system is comprised of.

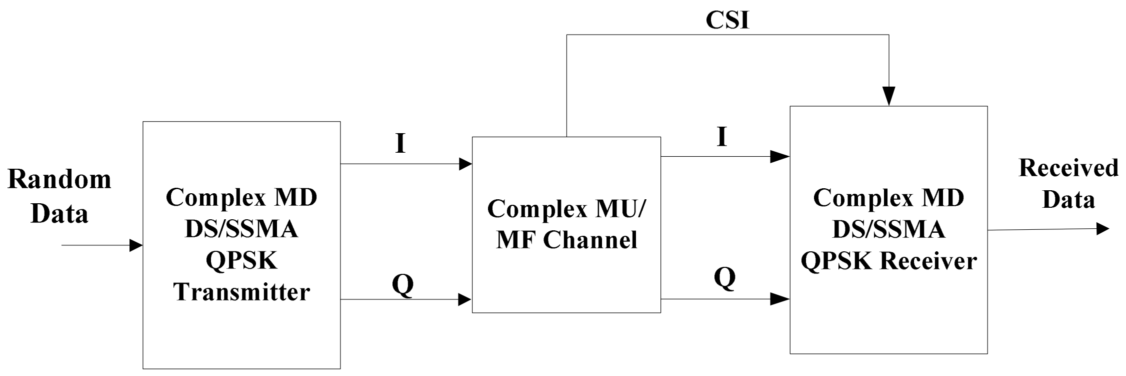

CSI
Random Data
Complex MD DS/SSMA QPSK Transmitter
I
Q
Complex MU/ MF Channel
I
Q
Complex MD DS/SSMA QPSK Receiver
Received Data

Figure 2.1: MD DS/SSMA System with Channel Simulator

2.2 COMPLEX MD DS/SSMA TRANSMITTER STRUCTURE

Multidimensional multicode spread spectrum techniques are becoming a popular choice for the realization of high rate transmission in the current 3G, and future 4G communication systems and have been evaluated and improved by many authors [21-24]. This section will propose a transmitter structure which incorporates one such a technique that improves throughput, as well as spectral efficiency. Fig. 2.2 illustrates the general structure of the complex baseband MD DS/SSMA QPSK-based transmitter of user-*n* in a multi-user CDMA system. Firstly, user-*n*'s transmitter is assigned two complex spreading sequences $C_1^n(t)$ and $C_2^n(t)$ denoted by:

$$C_1^n(t) = C_1^{n,\mathrm{Re}}(t) + j.C_1^{n,\mathrm{Im}}(t)\,, \tag{2.1}$$

and

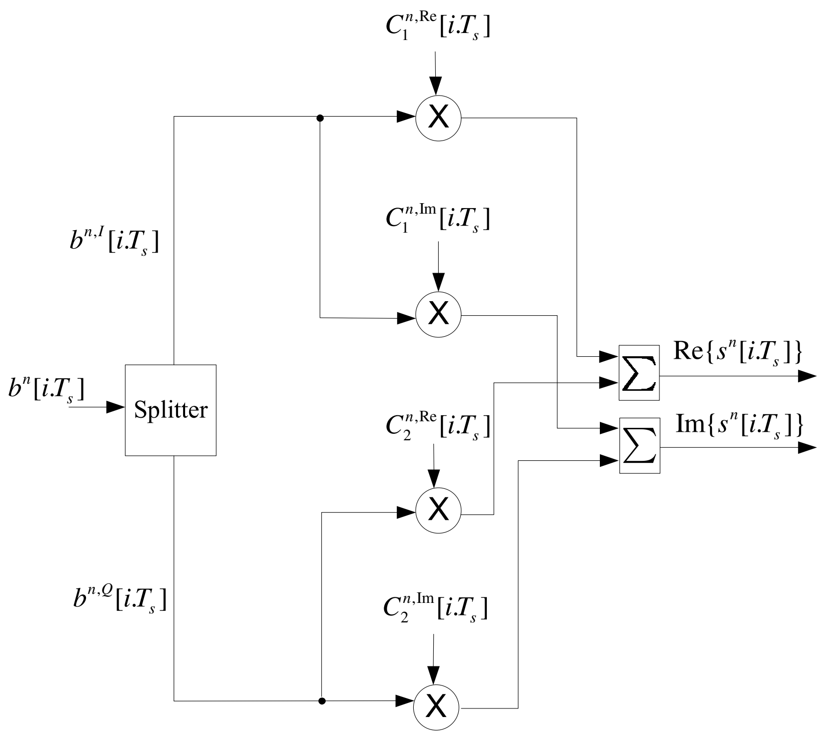

$C_1^{n,\mathrm{Re}}[i.T_s]$
$C_1^{n,\mathrm{Im}}[i.T_s]$
$b^{n,I}[i.T_s]$
$b^n[i.T_s]$
Splitter
$C_2^{n,\mathrm{Re}}[i.T_s]$
$b^{n,Q}[i.T_s]$
$C_2^{n,\mathrm{Im}}[i.T_s]$
$\mathrm{Re}\{s^n[i.T_s]\}$
$\mathrm{Im}\{s^n[i.T_s]\}$

Figure 2.2: Complex MD DS/SSMA QPSK Transmitter Structure for the n^{th} User

$$C_2^n(t) = C_2^{n,\mathrm{Re}}(t) + j.C_2^{n,\mathrm{Im}}(t). \tag{2.2}$$

The CSSs used throughout this study are known as *double-sideband* (DSB) *Constant-Envelope Linearly Interpolated Root-of-Unity* (CE-LI-RU) filtered *General Chirp-Like* (GCL) sequences. The details of these spreading codes and their generation are summarized in *Appendix B*. Next, assume that the real and imaginary parts of each sequence, i.e. $C_1^{n,\mathrm{Re}}(t)$, $C_1^{n,\mathrm{Im}}(t)$, $C_2^{n,\mathrm{Re}}(t)$ and $C_2^{n,\mathrm{Im}}(t)$, are sampled at a rate of L samples per chip and that each sequence contains J chips. We now declare a constant R that defines the number of samples per bit as:

$$R = L.J \text{ (samples/bit)}. \tag{2.3}$$

Next, assume that random data bits are being generated by a random binary source of user-n at a rate of f_b (bits per second) from an antipodal alphabet {-1, 1}. Next, assume that each bit is sampled at a rate of f_s (samples/second) and that each output sample is passed to the transmitter as $b^n[i.T_s]$. For each sample value $b^n[i.T_s]$ within a bit period $T_b = 1/f_b$, we have $T_s = 1/f_s$ and i = 1, 2, 3, …, R-1, R, where R is defined by *Eq*.(2.3). We finally define the sampled sequences as $C_1^{n,\mathrm{Re}}[i.T_s]$, $C_1^{n,\mathrm{Im}}[i.T_s]$, $C_2^{n,\mathrm{Re}}[i.T_s]$ and $C_2^{n,\mathrm{Im}}[i.T_s]$.

Eq. (2.1) and (2.2) can now be redefined as:

$$C_1^n[i.T_s] = C_1^{n,\mathrm{Re}}[i.T_s] + j.C_1^{n,\mathrm{Im}}[i.T_s], \tag{2.4}$$

and

$$C_2^n[i.T_s] = C_2^{n,\mathrm{Re}}[i.T_s] + j.C_2^{n,\mathrm{Im}}[i.T_s], \tag{2.5}$$

respectively. The transmitter now operates as follows: The transmitter receives two consecutive input bits $\overline{b}^n = \{b_1, b_2\}$, which is sampled to give $b^n[i.T_s]$ within one symbol period T_{sym}. Because the system is QPSK-based, we have $T_{sym} = 2.T_b$. The first function of the transmitter is to separate the two sample streams containing the statistically independent input bit values contained in $b^n[i.T_s]$ to form primary I-channel and Q-channel input sample streams, represented by $b^{n,I}[i.T_s]$ and $b^{n,Q}[i.T_s]$, respectively by using a *serial-to-parallel converter.* Next, these primary I and Q-channel sample streams are further split into secondary I- and Q-channel vectors containing the exact same samples as the primary I- and Q-channel vectors, respectively, to create a *balanced* configuration [3]. Spreading is now performed as follows (see Fig. 2.2): Firstly, the i^{th} sample $b^{n,I}[i.T_s]$ of the I-channel input sample stream is multiplied by both the i^{th} samples of $C_1^{n,\mathrm{Re}}[i.T_s]$ and $C_1^{n,\mathrm{Im}}[i.T_s]$. Secondly, the i^{th} sample $b^{n,Q}[i.T_s]$ of the Q-channel input sample stream is multiplied by both the i^{th} samples of $C_2^{n,\mathrm{Re}}[i.T_s]$ and $C_2^{n,\mathrm{Im}}[i.T_s]$. Lastly, the I- and Q-parts of the resultant (spread) sample streams are linearly combined (as illustrated by Fig. 2.2) into the main I-and Q-channel sample streams, $s^{n,I}[i.T_s]$ and $s^{n,Q}[i.T_s]$, respectively. These two outputs form the main sampled complex output signal and is defined by:

$$\mathrm{Re}\{s^n[i.T_s]\} = s^{n,I}[i.T_s] = b^{n,I}[i.T_s].C_1^{n,\mathrm{Re}}[i.T_s] + b^{n,Q}[i.T_s].C_2^{n,\mathrm{Re}}[i.T_s], \tag{2.6}$$

and

$$\mathrm{Im}\{s^n[i.T_s]\} = s^{n,Q}[i.T_s] = b^{n,I}[i.T_s].C_1^{n,\mathrm{Im}}[i.T_s] + b^{n,Q}[i.T_s].C_2^{n,\mathrm{Im}}[i.T_s] \tag{2.7}$$

From Fig. 2.2 and Eq. (2.6) and (2.7) it can be seen that this transmitter configuration is aimed at producing transmit diversity [6] by spreading two bits of user-*n* equally between the four dimensions of the complex spreading codes. Recovering these bits with a properly matched receiver (see section 2.4) will ensure that diversity gains are achieved. This effectively improves system BER, but sacrifices half the throughput.

2.3 COMPLEX MULTI-USER MULTIPATH FADING CHANNEL STRUCTURE

This section will firstly give a mathematical overview of the complex implementation of a FFCS, as proposed by *Staphorst* [3]. Thereafter, the proposed FFCS will be expanded into a complex multipath fading channel with unique channel parameters for each user in the system. Lastly, this multipath fading channel will be further expanded into a multi-user multipath fading channel with AWGN.

2.3.1 Complex Flat Fading Channel Structure

Fig. 2.3 shows an illustration of the complex implementation of a FFCS for the k^{th} multipath component, as proposed by *Staphorst* [3].

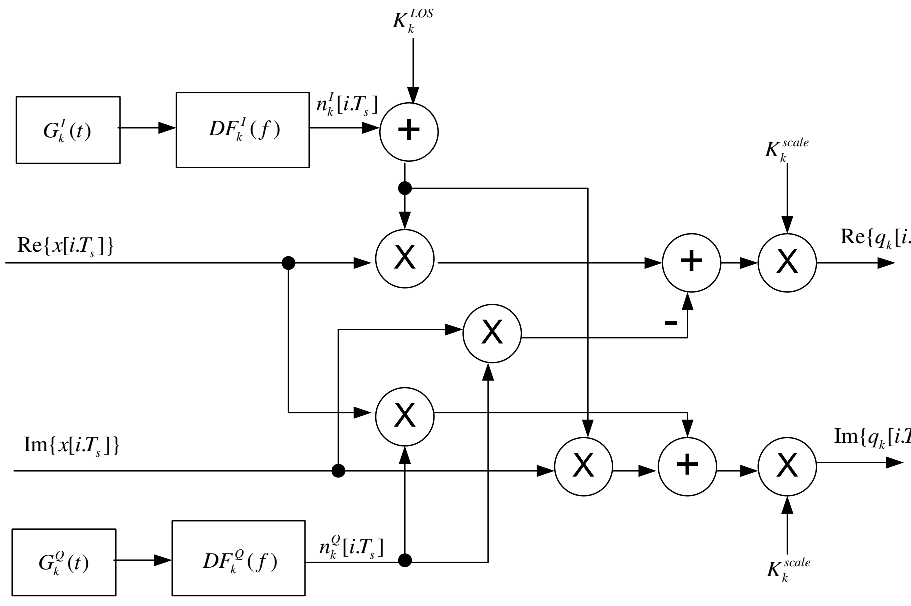


Figure 2.3: Complex Flat Fading Channel Simulator Structure

The sampled complex transmitted signal $x[i.T_s]$ at the input to the channel consists of a real and imaginary part, defined as follows [3]:

$$x_k[i.T_s] = \mathrm{Re}\{x_k[i.T_s]\} + j.\mathrm{Im}\{x_k[i.T_s]\}. \tag{2.8}$$

A complex flat fading process is defined as [3]:

$$\psi_k[i.T_s] = \vartheta_k[i.T_s].\cos(\theta_k[i.T_s]) + K_k^{LOS} + j.\vartheta_k[i.T_s].\sin(\theta_k[i.T_s]), \tag{2.9}$$

where $\vartheta_k[i.T_s]$ and $\theta_k[i.T_s]$ are the instantaneous fading amplitude and phase of the k^{th} multipath signal component, respectively, and K_k^{LOS} is known as the *line-of-sight* (LOS) component, defined as [3]:

$$K_k^{LOS} = \sqrt{2}.(10^{\frac{W_k}{20}}), \tag{2.10}$$

where W_k is defined as the *Rician factor* (in dB) [3]. The k^{th} complex multipath fading channel component is now given by [3]:

$$b_k[i.T_s] = \vartheta_k[i.T_s].x_k[i.T_s] = \mathrm{Re}\{q_k[i.T_s]\} + j.\mathrm{Im}\{q_k[i.T_s]\}. \tag{2.11}$$

The in-phase and quadrature outputs of the k^{th} FFCS, for a complex input signal, are thus given by [3]:

$$\mathrm{Re}\{q_k[i.T_s]\} = \mathrm{Re}\{\vartheta_k[i.T_s]\}.\mathrm{Re}\{x_k[i.T_s]\} - \mathrm{Im}\{\vartheta_k(t)\}.\mathrm{Im}\{x[i.T_s]_k\}, \tag{2.12}$$

and:

$$\mathrm{Im}\{q_k[i.T_s]\} = \mathrm{Re}\{\vartheta_k[i.T_s]\}.\mathrm{Im}\{x_k[i.T_s]\} + \mathrm{Im}\{\vartheta_i[i.T_s]\}.\mathrm{Re}\{x_k[i.T_s]\}, \tag{2.13}$$

respectively. Finally, from Fig. 2.3 it is clear that [3]:

$$\mathrm{Re}\{q_k[i.T_s]\} = K_k^{scale}\left[\left(n_k^I[i.T_s]+K_k^{LOS}\right).\mathrm{Re}\{x_k[i.T_s]\} - n_k^Q[i.T_s].\mathrm{Im}\{x_k[i.T_s]\}\right], \quad (2.14)$$

and:

$$\mathrm{Im}\{q_k[i.T_s]\} = K_k^{scale}\left[\left(n_k^I[i.T_s]+K_k^{LOS}\right).\mathrm{Im}\{x_k[i.T_s]\} + n_k^Q[i.T_s].\mathrm{Re}\{x_k[i.T_s]\}\right], \quad (2.15)$$

where $n_k^I[i.T_s]$ and $n_k^Q[i.T_s]$ are the *Discrete-Time* (DT) *Infinite Impulse Response* (IIR) Doppler filtered versions of the sampled, zero-mean Gaussian noise generated by the statistically independent noise generators $G_k^I(f)$ and $G_k^Q(f)$, respectively. Moreover, $G_k^I(f)$ and $G_k^Q(f)$ make use of the well-known *Marsaglia-Bray* mapping algorithm [26] that use *Wichmann-Hill* generated uniformly random numbers [25] to generate noise samples $G_k^I[i.T_s]$ and $G_k^Q[i.T_s]$ with the proper Gaussian statistics. The DT IIR Doppler lowpass filters $DF_k^I(f)$ and $DF_k^Q(f)$ are designed to realize realistic Doppler spreads [3]. The details on the realization of this filter can be found in *Appendix A*. K_k^{scale} is a scaling factor, incorporated to ensure that input and output signal power are equal on both I- and Q-channels of the system and is defined as [3]:

$$K_k^{scale} = \frac{1}{\sqrt{2+(K_k^{LOS})^2}}, \quad (2.16)$$

where K_k^{LOS} is defined by *Eq.* (2.10). In order to perform proper coherent demodulation at the receiver, we need to extract *Channel State Information* (CSI) from this channel. Because we will not be including CSI in the decoding process, we do not need the CSI of the instantaneous fading amplitude $\vartheta_k[i.T_s]$. The only CSI needed at the receiver to perform phase correction is that of the instantaneous phase of the fading channel $\theta_k[i.T_s]$. The instantaneous phase change experienced by the k^{th} multipath component is defined by [3]:

$$\theta_k[i.T_s] = -\arctan\left(\frac{n_k^Q[i.T_s]}{n_k^I[i.T_s]+K_k^{LOS}}\right). \quad (2.17)$$

2.3.2 Complex Multipath Fading Channel Structure

We expand the FFCS to a unique frequency-selective fading channel for each user in the system by creating a *Multipath Fading Channel Simulator* (MFCS) structure, which is capable of simulating a time-invariant complex multipath fading channel, consisting of *H* discrete and independently faded multipath components. Fig. 2.4 illustrates the structure of such a channel simulator.

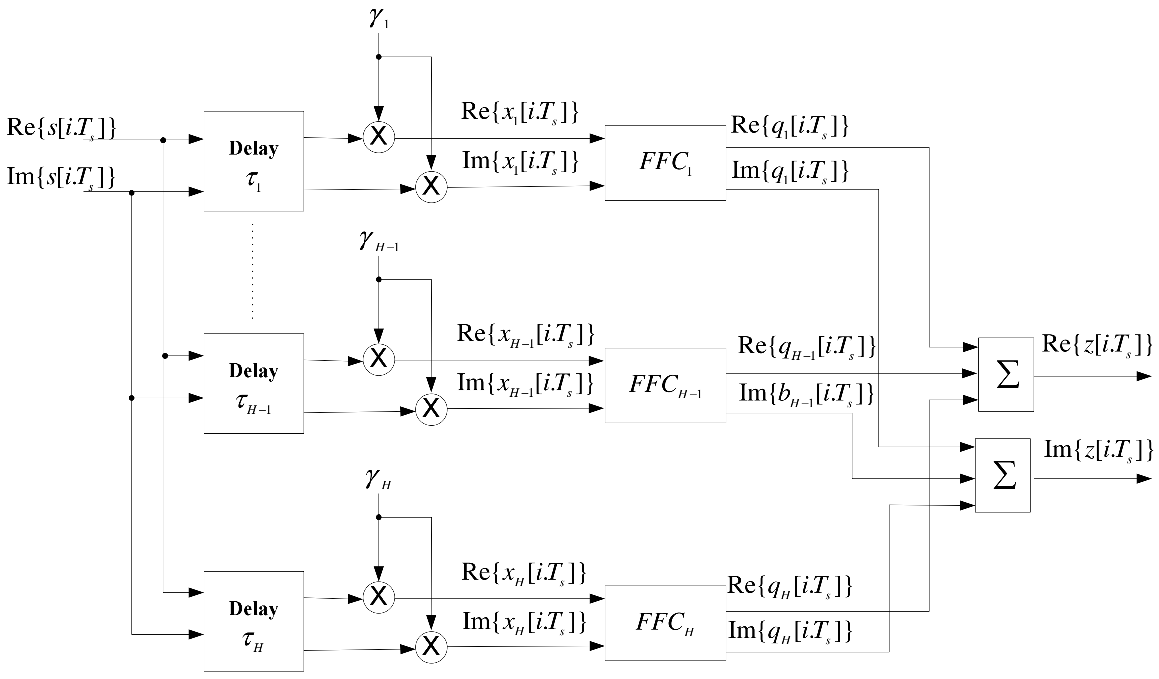

γ_1
$\mathrm{Re}\{s[i.T_s]\}$
$\mathrm{Im}\{s[i.T_s]\}$
Delay τ_1
$\mathrm{Re}\{x_1[i.T_s]\}$
$\mathrm{Im}\{x_1[i.T_s]\}$
FFC_1
$\mathrm{Re}\{q_1[i.T_s]\}$
$\mathrm{Im}\{q_1[i.T_s]\}$
γ_{H-1}
Delay τ_{H-1}
$\mathrm{Re}\{x_{H-1}[i.T_s]\}$
$\mathrm{Im}\{x_{H-1}[i.T_s]\}$
FFC_{H-1}
$\mathrm{Re}\{q_{H-1}[i.T_s]\}$
$\mathrm{Im}\{b_{H-1}[i.T_s]\}$
Σ
$\mathrm{Re}\{z[i.T_s]\}$
Σ
$\mathrm{Im}\{z[i.T_s]\}$
γ_H
Delay τ_H
$\mathrm{Re}\{x_H[i.T_s]\}$
$\mathrm{Im}\{x_H[i.T_s]\}$
FFC_H
$\mathrm{Re}\{q_H[i.T_s]\}$
$\mathrm{Im}\{q_H[i.T_s]\}$

Figure 2.4: Complex Multipath Fading Channel Structure

The channel functions as follows [3]: First, an appropriate *H*-path power delay profile is chosen for the particular user. *H* time delayed versions of the real and imaginary parts of the complex transmitted signal are then created using a *H*-tap delay line with delay times equal to that of the required power delay profile [3]. These delayed signals are then scaled by γ_k, with $k = 1, 2, 3, \ldots, H$, and then passed through *H* unique complex FFCSs (Fig. 2.3) represented by $FFCS_1$ to $FFCS_H$. It is thus possible to define a unique Doppler spread and Rician factor for each multipath component. Finally, the resultant outputs $q_k[i.T_s]$ of each of the *H* flat fading channels, with $k = 1, 2, 3, \ldots, H$, are then linearly combined to give the resultant complex output signal $z[i.T_s]$. An important fact is that the power of the multipath fading channel simulator's complex output signal $z[i.T_s]$, must equal the power of the complex transmitted signal $s[i.T_s]$. Because the H propagation paths experience statistically independent flat fading, the scaling factors γ_k with k = 1, 2, 3, …, H, must satisfy the equation [3]:

$$\sum_{k=1}^{H} (\gamma_k)^2 = 1. \tag{2.18}$$

2.3.2.1 Power Delay Profiles

Multipath channel parameters are obtained from the power delay profiles of different types of environments. Because there is such a diverse number of environments, comparing these multipath channels are practically impossible. We will therefore use the most common type of power delay profile found in most simulation platforms for the performance evaluation of digital communication systems. This profile is called the *exponential decay power delay profile* model and is defined by [3]:

$$Y(\tau) = \frac{1}{Y_T} e^{(-\tau/\tau_e)}, \tag{2.19}$$

where τ is the time delay (in seconds). In *Eq* .(2.19), Y_T is a normalization factor, defined by [3]:

$$Y_T = \sum_{k=1}^{H} Y(\tau_k) = \sum_{k=1}^{H} (\gamma_k)^2, \tag{2.20}$$

with τ_k the time delay of the k^{th} path. Also, τ_e is the time constant of the profile, defined by [3]:

$$\tau_e = -\frac{\tau_{\max}}{\ln\left(10^{\frac{Y_{drop}}{10}}\right)}. \tag{2.21}$$

Finally, in *Eq.* (2.21), $\tau_{\max}$ is called the maximum excess delay, and is defined by [3]:

$$\tau_{\max} = \tau_H - \tau_1, \tag{2.22}$$

where τ_H is the maximum delay at which a multipath component is within Y_{drop} (dB) of the strongest arriving multipath component, and τ_1 the time delay of the first multipath component. Note that the strongest component is not necessarily the first arrival component, but this fact is true for the case of the exponential decay power delay profile, hence *Eq.* (2.22) holds. Also, the CSI of *Eq.* (2.17) is obtained from this particular (strongest) path. Moreover, a typical value for Y_{drop} is -30 dB [3].

2.3.3 Complex Multi-User Multipath Fading Channel Structure

We now expand the MFCS of *section* 2.3.2 to a *Multi-User* (MU) MFCS. Fig. 2.5 illustrates the structure of the simulator.

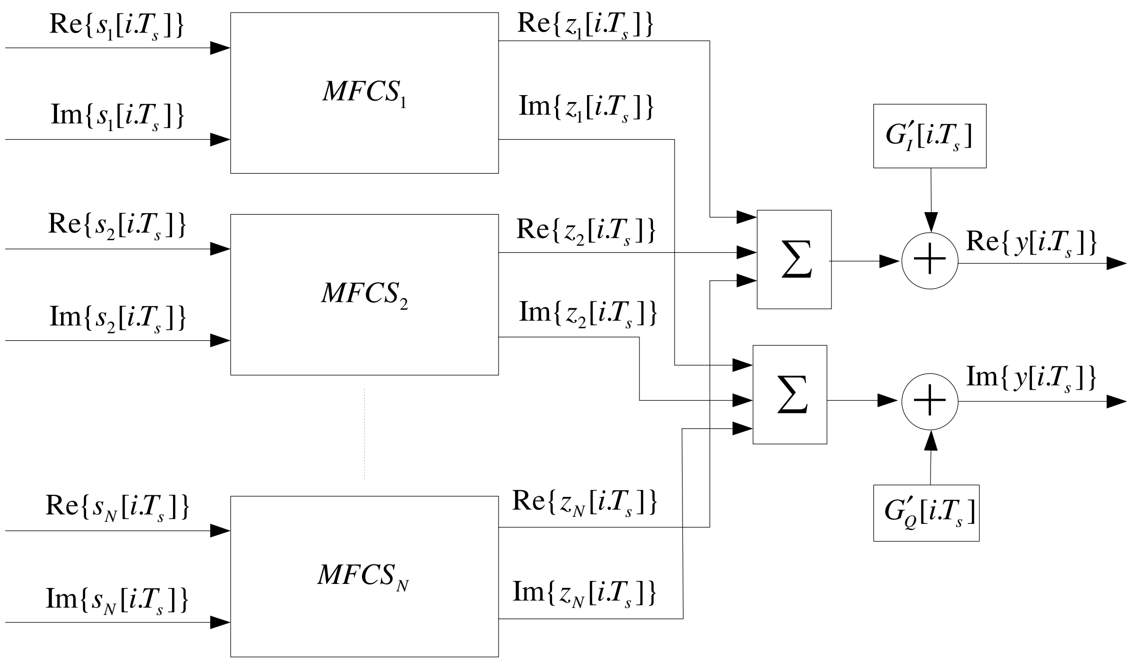

$\text{Re}\{s_1[i.T_s]\}$
$\text{Im}\{s_1[i.T_s]\}$
$MFCS_1$
$\text{Re}\{z_1[i.T_s]\}$
$\text{Im}\{z_1[i.T_s]\}$
$\text{Re}\{s_2[i.T_s]\}$
$\text{Im}\{s_2[i.T_s]\}$
$MFCS_2$
$\text{Re}\{z_2[i.T_s]\}$
$\text{Im}\{z_2[i.T_s]\}$
$\text{Re}\{s_N[i.T_s]\}$
$\text{Im}\{s_N[i.T_s]\}$
$MFCS_N$
$\text{Re}\{z_N[i.T_s]\}$
$\text{Im}\{z_N[i.T_s]\}$
Σ
Σ
$G'_I[i.T_s]$
$G'_Q[i.T_s]$
$\text{Re}\{y[i.T_s]\}$
$\text{Im}\{y[i.T_s]\}$

Figure 2.5: Multi-User Multipath Fading Channel Simulator Structure.

Each of the I- and Q-channel bit streams of *N*-users in the system is processed by a unique complex MFCS (Fig. 2.4), represented by $MFCS_1$ to $MFCS_N$, each with unique multipath parameters. Next, all of the I- and Q-channel output signals from each of the *N*-MFCSs are linearly combined into common incoming I- and Q-channel symbol streams (see Fig. 2.5). Finally, two statistically independent AWGN generators, denoted by $G_I'[i.T_s]$ and $G_Q'[i.T_s]$ add noise to the I- and Q-channel symbol streams, respectively, producing the final complex output $y[i.T_s]$. The generators firstly generate zero-mean Gaussian noise samples, $G_I'[i.T_s]$ and $G_Q'[i.T_s]$, with noise variance $\sigma^2_{G_I'[i.T_s]} = \sigma^2_{G_Q'[i.T_s]} = 1$. In order to obtain noise samples with the required noise variance, as dictated by E_b / N_o, we scale the initially generated samples by a power scaling factor $\sqrt{\alpha_G}$, where α_G is defined by [3]:

$$\alpha_G = \frac{\sigma^2_{s[i.T_s]} \cdot f_s}{2 f_b . 10^{\left(\frac{E_b}{10.N_o}\right)}} . \tag{2.23}$$

In *Eq.* (2.23), $\sigma^2_{s[i.T_s]}$ is the variance (power) of the transmitted signal, f_s is the sampling rate, f_b is the uncoded bit rate and E_b / N_o the required *signal-to-noise ratio* (SNR) per bit (in dB).

2.4 COMPLEX DS/SSMA QPSK RECEIVER STRUCTURE

The complex MD DS/SSMA QPSK receiver associated with the transmitter structure of *section* 2.2 for user-*n* in the CDMA system is shown in Fig. 2.6. The goal of this receiver is to optimally recover the data from the channel, as well as exploiting the diversity introduced by the transmitter. The receiver operates as follows: The complex transmitted sample stream $s^n[i.T_s]$ is received at the input of the receiver as $y^n[i.T_s]$. Each sample value $y^{n,I}[i.T_s]$ and $y^{n,Q}[i.T_s]$ has been affected by fading, AWGN and *Multi-User Interference* (MUI). This received signal is defined as [3]:

$$\begin{aligned} y^n[i.T_s] &= \text{Re}\{y^n[i.T_s]\} + j.\text{Im}\{y^n[i.T_s]\} \\ &= \psi[i.T_s].\left[\text{Re}\{s^n[i.T_s]\} + j.\text{Im}\{s^n[i.T_s]\}\right] + G[i.T_s] + Mui[i.T_s] \end{aligned} , \tag{2.24}$$

with $\psi[i.T_s]$ a complex fading process, defined by *Eq.* (2.9), $G[i.T_s]$ a complex noise process, defined by:

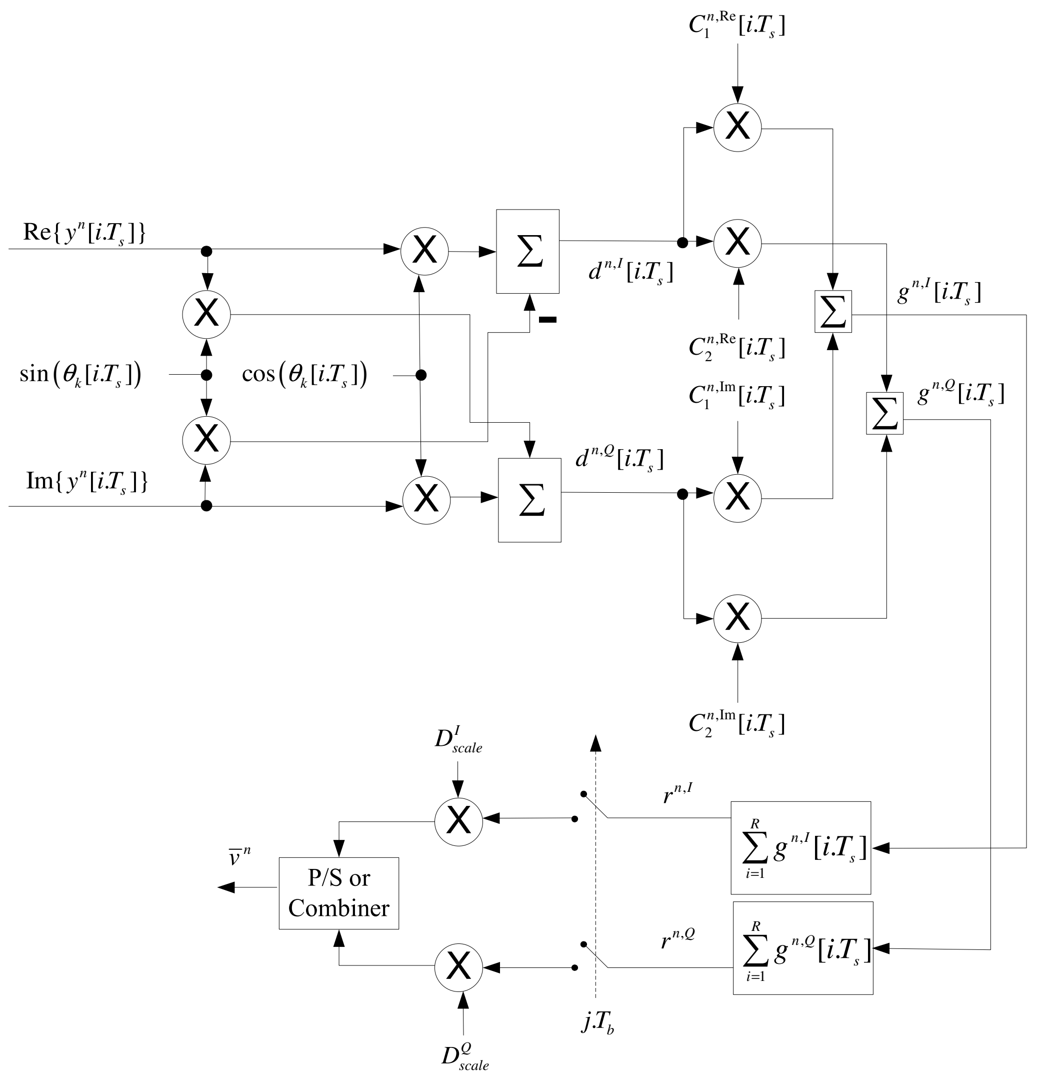

$C_1^{n,\mathrm{Re}}[i.T_s]$
$\mathrm{Re}\{y^n[i.T_s]\}$
$d^{n,I}[i.T_s]$
$g^{n,I}[i.T_s]$
$\sin(\theta_k[i.T_s])$
$\cos(\theta_k[i.T_s])$
$C_2^{n,\mathrm{Re}}[i.T_s]$
$C_1^{n,\mathrm{Im}}[i.T_s]$
$g^{n,Q}[i.T_s]$
$d^{n,Q}[i.T_s]$
$\mathrm{Im}\{y^n[i.T_s]\}$
$C_2^{n,\mathrm{Im}}[i.T_s]$
D_{scale}^{I}
$r^{n,I}$
$\bar{v}^n$
P/S or Combiner
$\sum_{i=1}^{R} g^{n,I}[i.T_s]$
$r^{n,Q}$
$\sum_{i=1}^{R} g^{n,Q}[i.T_s]$
$j.T_b$
D_{scale}^{Q}

Figure 2.6: Complex MD DS/SSMA QPSK Receiver Structure for User-*n*

$$\begin{aligned} G[i.T_s] &= \mathrm{Re}\{G[i.T_s]\} + j.\mathrm{Im}\{G[i.T_s]\} \\ &= G_I'[i.T_s] + j.G_Q'[i.T_s], \end{aligned} \tag{2.25}$$

and $Mui[i.T_s]$ a complex MUI process term. Assume in this event that the spreading codes being used are mutually orthogonal, i.e. having perfect cross correlation properties, makingthe term $Mui[i.T_s]=0$. The sample values $y^{n,I}[i.T_s]$ and $y^{n,Q}[i.T_s]$ of the complex input signal $y^n[i.T_s]$ is firstly phase corrected with the use of the proper CSI, obtained from the associated channel of user-*n*, defined by *Eq.* (2.17), (see *section* 2.3.1), to yield the sample values:

$$d^{n,I}[i.T_s] = y^{n,I}[i.T_s].\cos\left(\theta[i.T_s]\right) - y^{n,Q}[i.T_s].\sin\left(\theta[i.T_s]\right), \tag{2.26}$$

and

$$d^{n,Q}[i.T_s] = y^{n,I}[i.T_s].\sin\left(\theta[i.T_s]\right) + y^{n,Q}[i.T_s].\cos\left(\theta[i.T_s]\right). \tag{2.27}$$

Next, the samples $d^{n,I}[i.T_s]$ and $d^{n,Q}[i.T_s]$ are coherently de-spread by using the same CSSs samples $C_1^{n,\mathrm{Re}}[i.T_s]$, $C_1^{n,\mathrm{Im}}[i.T_s]$, $C_2^{n,\mathrm{Re}}[i.T_s]$ and $C_2^{n,\mathrm{Im}}[i.T_s]$ on the individual I- and Q-channel symbol streams used to spread the data at the associated transmitter of user-*n* (see *section* 2.2). The respective I- and Q-channel resultant signals (after de-spreading) are then summed together in order to increase the Euclidean distance between the signals in the signal space (see Fig. 2.6), which yields:

$$g^{n,I}[i.T_s] = d^{n,I}[i.T_s].C_1^{n,\mathrm{Re}}[i.T_s] + d^{n,Q}[i.T_s].C_1^{n,\mathrm{Im}}[i.T_s], \tag{2.28}$$

and:

$$g^{n,Q}[i.T_s] = d^{n,I}[i.T_s].C_2^{n,\mathrm{Re}}[i.T_s] + d^{n,Q}[i.T_s].C_2^{n,\mathrm{Im}}[i.T_s]\ . \tag{2.29}$$

In order to obtain the original bit value that was transmitted within bit period T_b, the de-spread samples $g^{n,I}[i.T_s]$ and $g^{n,Q}[i.T_s]$ are integrated over T_b using a *Riemann* summation [27] to yield:

$$r^{n,I} = \sum_{i=1}^{R} g^{n,I}[i.T_s], \tag{2.30}$$

and

$$r^{n,Q} = \sum_{i=1}^{R} g^{n,Q}[i.T_s], \tag{2.31}$$

with *R* defined by *Eq*.(2.3). Next, the values $r^{n,I}$ and $r^{n,Q}$ are firstly sampled at multiples of T_b and then scaled by factors D^I_{scale} and D^Q_{scale}, respectively, defined by:

$$D^I_{scale} = \left(\sum_{i=1}^{R} b^{n,I}[i.T_s] . \left[\left(\sum_{i=1}^{R} C_1^{n,\mathrm{Re}}[i.T_s] \right)^2 + \left(\sum_{i=1}^{R} C_1^{n,\mathrm{Im}}[i.T_s] \right)^2 \right] \right)^{-1} , \tag{2.32}$$

and

$$D^Q_{scale} = \left(\sum_{i=1}^{R} b^{n,Q}[i.T_s] . \left[\left(\sum_{i=1}^{R} C_2^{n,\mathrm{Re}}[i.T_s] \right)^2 + \left(\sum_{i=1}^{R} C_2^{n,\mathrm{Im}}[i.T_s] \right)^2 \right] \right)^{-1} , \tag{2.33}$$

to ensure that $-1 < r^{n,I} < 1$ and $-1 < r^{n,Q} < 1$. The scaled values of $r^{n,I}$ and $r^{n,Q}$ are then finally combined back into a vector $\overline{v}^n = \{r^{n,I}, r^{n,Q}\}$, using a parallel-to-serial device. This is the final estimate of the original bit vector $\overline{b}^n = \{b_1, b_2\}$ of user-*n* (see *section* 2.2).

2.5 CONCLUDING REMARKS

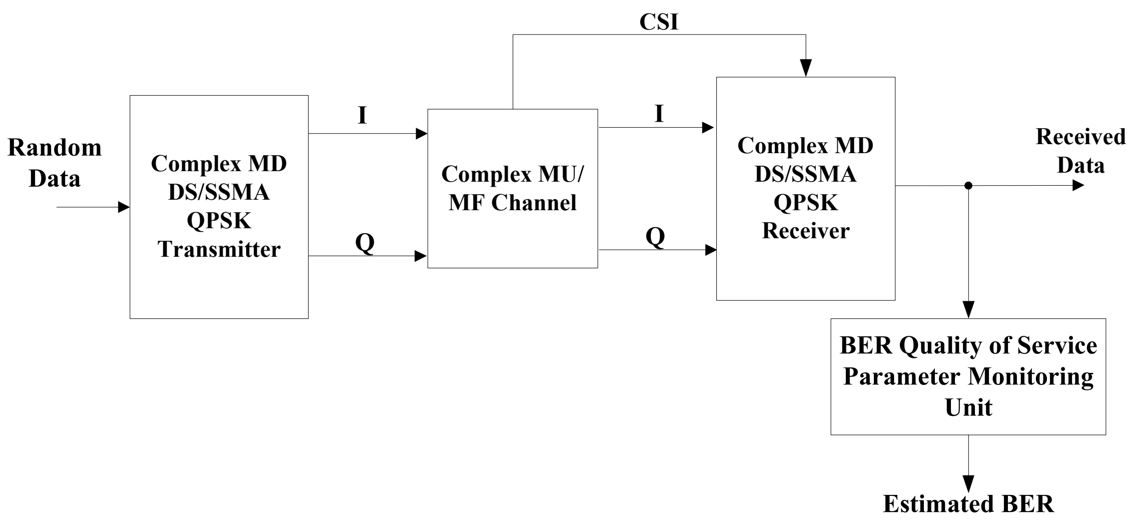
The goal of this chapter was to describe and discuss the individual functional units in the MD DS/SSMA QPSK-based communication system used throughout this study. The first part of this chapter considered the general structure of the proposed MD DS/SSMA QPSK transmitter used that introduces diversity. This was followed by a mathematical overview of the multipath fading channel model and simulator proposed by *Staphorst* [3]. The chapter concludes with the structure and discussion of the matching MD DS/SSMA QPSK receiver that exploits the diversity introduced by the transmitter.

CHAPTER THREE

REAL-TIME BER MEASUREMENT

3.1 OVERVIEW

This chapter is dedicated to the description of a *Quality-of-Service Monitoring Unit* (QoSMU) structure that will monitor the BER performance of a system in real-time by estimating the true uncoded probability of error through measurement. Fig. 3.1 illustrates the position of this QoSMU within the system structure described thus far.



CSI
Random Data
Complex MD DS/SSMA QPSK Transmitter
I
Q
Complex MU/ MF Channel
I
Q
Complex MD DS/SSMA QPSK Receiver
Received Data
BER Quality of Service Parameter Monitoring Unit
Estimated BER

Figure 3.1: System Illustrating the Position of the BER QoSMU

The functional details of a QoSMU that is capable of measuring real-time BER were first investigated by *Gooding* in 1965 [12], where he proposed the estimation of BER through the extrapolation of *pseudo-error rate* (PER) in order to estimate the true probability of error of a system under predetermined channel conditions. It is also the intent of this chapter to review this technique and create a clear basis for the extended research and simulations that will be done on this technique later in this study.

This chapter starts with a description of the general structure of the QoSMU, as well as an explanation of the technique used for pseudo-error generation and PER calculation. This is followed by an investigation into the characteristics of the true pseudo-error rate versus threshold value in classic AWGN conditions. Next, a mathematical description and simple derivation of the *classic linear extrapolation* algorithm, as suggested by *Gooding* [12], will be given, as well as a short sub-section that suggests a technique for improving the linearity of the pseudo-error rate versus modified threshold characteristic, that yields an improvement to the classic linear extrapolation technique. The following section describes a higher-order extrapolation algorithm, namely that of *quadratic-extrapolation*, which will further improve the accuracy of the BER estimate. The last section is dedicated to comparing the speed and accuracy of these three techniques in classic AWGN conditions.

3.2 GENERAL MONITORING UNIT STRUCTURE

Fig. 3.2 illustrates the general structure of the QoSMU, as described by *Gooding* [12].

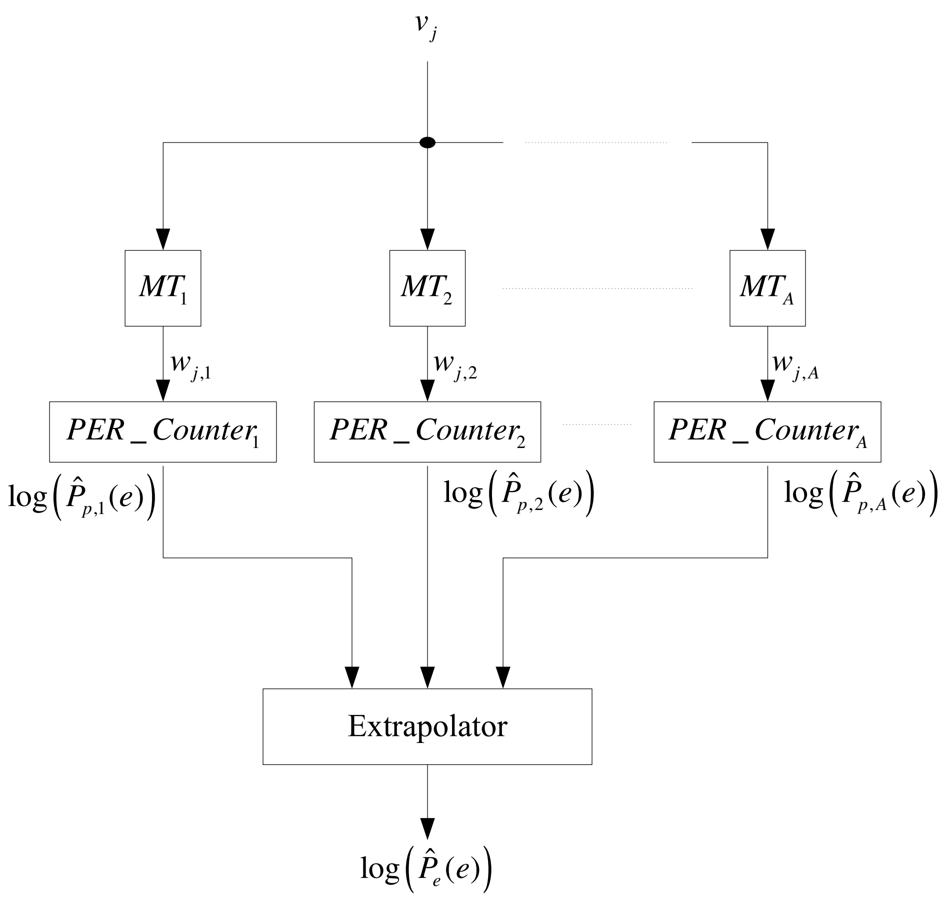

v_j
MT_1
MT_2
MT_A
$w_{j,1}$
$w_{j,2}$
$w_{j,A}$
$PER\_Counter_1$
$PER\_Counter_2$
$PER\_Counter_A$
$\log\left(\hat{P}_{p,1}(e)\right)$
$\log\left(\hat{P}_{p,2}(e)\right)$
$\log\left(\hat{P}_{p,A}(e)\right)$
Extrapolator
$\log\left(\hat{P}_e(e)\right)$

Figure 3.2: General Structure of the QoSMU

The QoSMU functions as follows: The QoSMU consists of A-tapped secondary paths, parallel to the main data recovery path. Each path accepts the j^{th} soft-output bit value v_j and passes it through its own modified threshold (MT) decision device (MT_1 to MT_A) that generates a pseudo-error output value, $w_{j,a}$ according to the following rule:

$$w_{j,a} = \begin{Bmatrix} 1 & (-d_a < v_j < d_a) \\ 0 & elsewhere \end{Bmatrix}, \tag{3.1}$$

where v_j is the j^{th} output bit value and d_a is the threshold value of the a^{th} path, with $a = 1, 2, 3, ..., A$. Thus, a pseudo-error value '1' is generated if the received bit value lies within some predetermined range. The A outputs of the modified threshold decision devices are now connected to A individual counters ($PER\_Counter_1$ to $PER\_Counter_A$) that counts these pseudo-errors and calculates a PER output estimate $\hat{P}_{p,a}(e)$ as follows:

$$\hat{P}_{p,a}(e) = \frac{1}{X}\sum_{j=1}^{X} w_{j,a} , \tag{3.2}$$

where X is the total number of bit values received. This value is an estimate of the true probability of pseudo-error $P_{p,a}(e)$. It is important to state that the calculation of Eq. (3.2) is only performed as soon as the number of pseudo-errors reaches some predetermined value. This process is also referred to as a *Monte-Carlo* process for obtaining a pseudo-error rate. Furthermore:

$$\lim_{X\to\infty} \hat{P}_{p,a}(e) = \lim_{X\to\infty} \frac{1}{X}\sum_{j=1}^{X} w_{j,a} = P_{p,a}(e) . \tag{3.3}$$

The A PER values are finally fed into an extrapolator that calculates an estimate of the true probability of error by extrapolating these A PER values to obtain a hypothetical intersection point:

$$\log\left(\hat{P}_e(e)\right) = \log\left(P_p(e)\right)\Big|_{d_a=0} , \tag{3.4}$$

where d_a is the threshold value. It should be noted that because of *Eq.* (3.1):

$$\lim_{d_a \to 0} \log\left(P_p(e)\right) = -\infty, \tag{3.5}$$

therefore, we define $\hat{P}_e(e)$ of *Eq.* (3.4) as being a hypothetical value. This is also verified in the next section when the general expression for the true probability of pseudo-error is revealed. This technique, whereby the receiver decision threshold is modified in order to obtain pseudo-errors, is known as the *threshold modification* technique [17] and has been exhaustively researched and published by many authors [12-19] since its inception by *Gooding* [12] in 1965. This technique has proven to be the most robust technique to use for pseudo-error generation [17] and BER estimation. This technique also eliminates the need for known transmitted data, which makes it a very attractive technique for real-time BER estimation.

We will now investigate the characteristics of the true probability of pseudo-error $P_p(e)$ versus threshold d under AWGN conditions in order to fully understand the motivation behind the three extrapolation algorithms considered in this study. This is done in the next section.

3.3 TRUE PROBABILITY OF PSEUDO-ERROR VERSUS THRESHOLD IN AWGN

This section investigates the characteristics of the true probability of pseudo-error $P_p(e)$ versus threshold d under classic AWGN conditions. To this end, we need to find the expression for the true probability of pseudo-error $P_p(e)$ as a function of the threshold d and the *signal-to-noise ratio* (SNR) per bit E_b / N_o , where E_b is the signal energy per bit and $N_o / 2$ is the power spectral density of Gaussian noise. The expression for the true probability of pseudo-error $P_p(e)$ for modified threshold under AWGN conditions is given by [15]:

$$P_p(e) = Q\left[(1+d).\sqrt{\frac{2.E_b}{N_o}}\right] - Q\left[(1-d).\sqrt{\frac{2.E_b}{N_o}}\right], \tag{3.6}$$

with:

$$Q[x] = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt\,, \tag{3.7}$$

and d is chosen such that $0 < d < 1$. *Eq*. (3.5) can now be verified with *Eq*. (3.6) by letting $d \to 0$. Fig. 3.3 shows a plot of the log probability of pseudo-error $P_p(e)$ versus threshold d for different values of SNR per bit using Eq. (3.6).

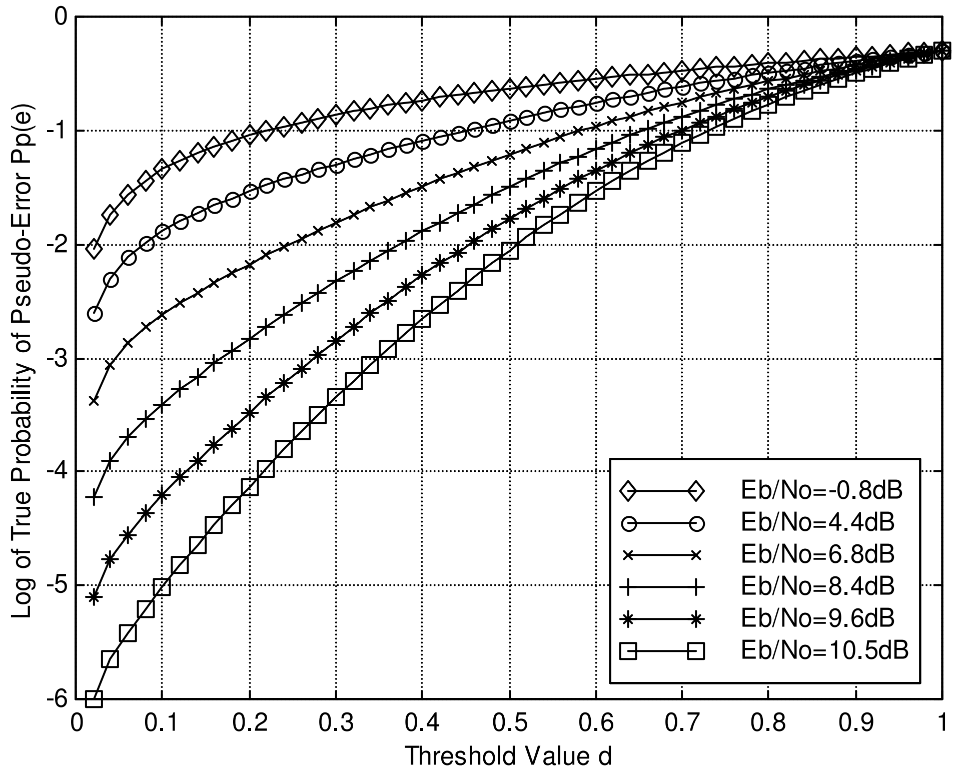


Figure 3.3: Log of True Probability of Pseudo-Error vs. Threshold in AWGN for different E_b / N_o.

Fig. 3.3 reveals an approximately linear relationship between the log of $P_p(e)$ and the threshold value d for d > 0.2. We can now find the log estimate of the true probability of error at a specific SNR value by first finding two or more values of $P_p(e)$ at a threshold $d > 0.2$. These values are then extrapolated to determine the hypothetical y-intersect value $\log\left(P_p(e)\right)\Big|_{d=0}$, which is a log estimate of the true probability of error, and is denoted by

$\log\left(\hat{P}_e(e)\right)$. The minimum number of pseudo-error values $P_{p,A}(e)$ needed for extrapolation is A = 2. In this case, the extrapolation technique is referred to as linear extrapolation and the details thereof are described in the next section.

3.4 LINEAR EXTRAPOLATION

3.4.1 Classic Linear Extrapolation in AWGN

As described in section 3.3, we need A = 2 pseudo-error values in order to perform linear extrapolation. We thus define two threshold values and denote them as d_1 and d_2. We then define the two corresponding pseudo-error values as:

$$P_{p,1}(e) = P_p(e)\Big|_{d=d_1}, \tag{3.8}$$

and

$$P_{p,2}(e) = P_p(e)\Big|_{d=d_2}. \tag{3.9}$$

We now calculate the log estimate of the true probability of error denoted as $\log\left(\hat{P}_e(e)\right)$ through linear extrapolation as follows [12]:

$$\log\left(\hat{P}_e(e)\right) = \frac{d_2.\log\left(P_{p,1}(e)\right) - d_1.\log\left(P_{p,2}(e)\right)}{d_2 - d_1}. \tag{3.10}$$

As an example, we define two values of the threshold $d_1 = 0.3$ and $d_2 = 0.7$ at $E_b / N_o = 8.4dB$. By using Eq. (3.6) we calculate $\log\left(P_{p,1}(e)\right) = -2.33$ and $\log\left(P_{p,2}(e)\right) = -0.877$. Finally, we use Eq. (3.10) and calculate $\log\left(\hat{P}_e(e)\right) = -3.419$. In order to calculate the true probability of error for comparison, we firstly define the expression for the true probability of error for QPSK signals in AWGN. This expression is given by [6]:

$$P_e(e) = Q\left[\sqrt{\frac{2.E_b}{N_o}}\right], \tag{3.11}$$

where $Q[x]$ is defined by Eq. (3.7), and E_b / N_o is the specific SNR per bit. Fig. 3.4 firstly illustrates the plot of the true log probability of pseudo-error vs. threshold in AWGN by use of Eq. (3.6) for the example above. Superimposed onto this plot is the graph of the straight line that originates from the linear extrapolation of the above example, which estimates the value of the true log probability of error $\log\left(P_e(e)\right)$. Indicated on the y-intersect of the graph are the values for the estimated log probability of error $\log\left(\hat{P}_e(e)\right)$, calculated from Eq. (3.10), as well as the true log probability of error $\log\left(P_e(e)\right)$ calculated using Eq. (3.11).

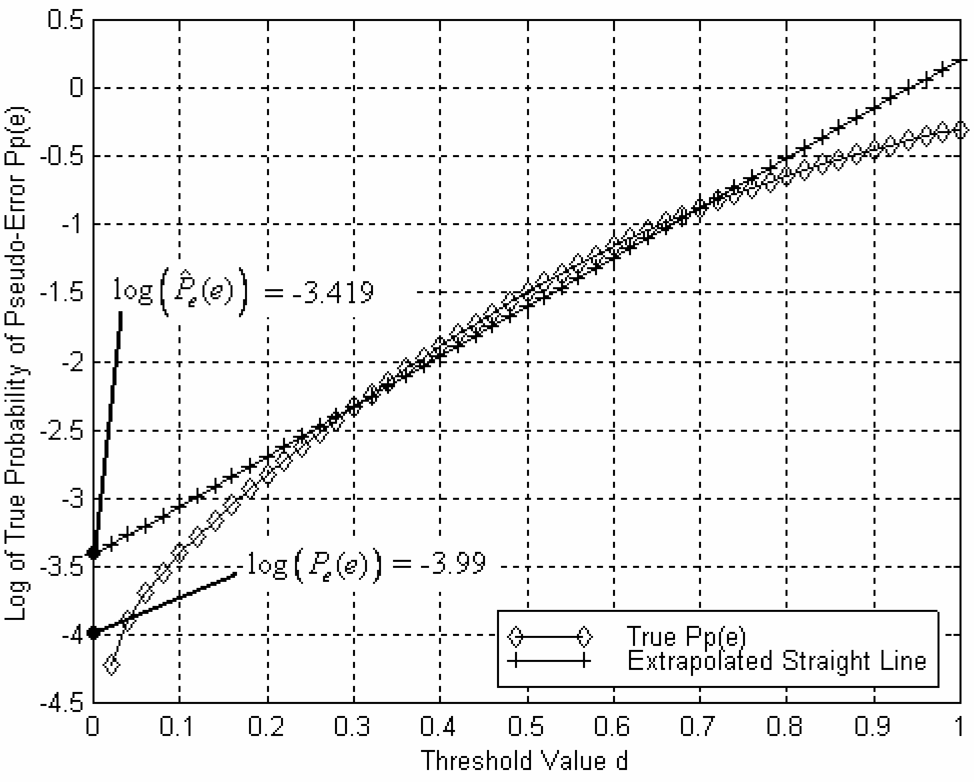


Figure 3.4: Superimposed Graph of the Linear Extrapolated Straight Line that Calculates $\log\left(\hat{P}_e(e)\right)$, and the True Log Probability of Pseudo-Error vs. Threshold in AWGN for $E_b / N_o = 8.4dB$.

During a real-time BER estimation process, each of the counters ($PER\_Counter_1$ to $PER\_Counter_A$) calculates an estimate of $P_{p,1}(e)$ and $P_{p,2}(e)$ denoted as $\hat{P}_{p,1}(e)$ and $\hat{P}_{p,2}(e)$ by using the *Monte-Carlo* process described in *section* 3.2 and by Eq. (3.1) and (3.2). Conclusively, Eq. (3.10) can thus now be redefined as:

$$\log\left(\hat{P}_e(e)\right) = \frac{d_2.\log\left(\hat{P}_{p,1}(e)\right) - d_1.\log\left(\hat{P}_{p,2}(e)\right)}{d_2 - d_1}. \tag{3.12}$$

3.4.2 Improved Linear Extrapolation in AWGN

This section will give a brief description of a technique for improving the linearity of the probability of pseudo-error $P_p(e)$ vs. threshold d characteristic under AWGN conditions, described in *section* 3.3. This technique will then be used to improve the results of classic linear extrapolation described in *section* 3.4.1.

During their investigations into the characteristics of the threshold modification and linear extrapolation techniques, Newcombe and Pasupathy [17] suggested a technique for improving the linearity of the probability of pseudo-error $P_p(e)$ vs. threshold d characteristic under AWGN conditions by linearly transforming the threshold d into a new value V with the following equation [17]:

$$V = d.(2-d) \tag{3.13}$$

Eq. (3.12) can now be redefined as:

$$\log\left(\hat{P}_e(e)\right) = \frac{V_2.\log\left(\hat{P}_{p,1}(e)\right) - V_1.\log\left(\hat{P}_{p,2}(e)\right)}{V_2 - V_1}. \tag{3.14}$$

Fig. 3.5 shows a plot of the true log probability of pseudo-error $P_p(e)$ versus adapted threshold V for different values of SNR per bit and 0 < d < 1. Fig. 3.5 reveals an improvement in the linear relationship between $\log\left(P_p(e)\right)$ and the adapted threshold value V for $V > 0.36$ $(d > 0.2)$.

Once again, we use the two threshold values of the previous example, i.e. $d_1 = 0.3$ and $d_2 = 0.7$, at $E_b / N_o = 8.4dB$. We now convert these two threshold values by using Eq. (3.13) and obtain $V_1 = 0.51$ and $V_2 = 0.91$. Now, by using Eq. (3.6), we calculate $\log\left(P_{p,1}(e)\right) = \text{-}2.331$ and $\log\left(P_{p,2}(e)\right) = \text{-}0.877$. Finally we use Eq. (3.14) and calculate $\log\left(\hat{P}_e(e)\right) = -4.184$. Fig. 3.6 illustrates the plot of the true log probability of pseudo-error vs. threshold value using Eq. (3.6) for the example above. Superimposed onto

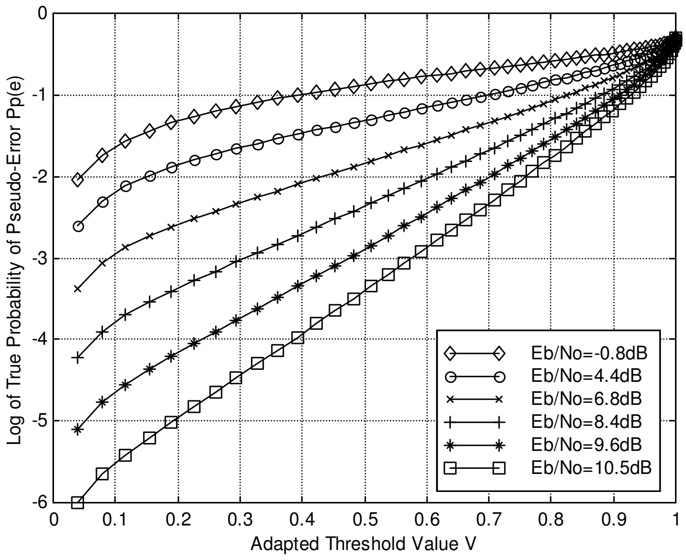


Figure 3.5: Log of True Probability of Pseudo-Error vs. Adapted Threshold in AWGN.

this plot is the graph of the straight line that originates from the linear extrapolation of the above example, which estimates the value of the true log probability of error $\log\left(P_e(e)\right)$. Indicated on the y-intersect of the graph are the values for the estimated log probability of error $\log\left(\hat{P}_e(e)\right)$, as well as the true log probability of error $\log\left(P_e(e)\right)$ calculated with Eq. (3.11). Fig. 3.6 reveals that there is a significant improvement in the estimated probability of error when the adaptation of the threshold is performed employing Eq. (3.13).

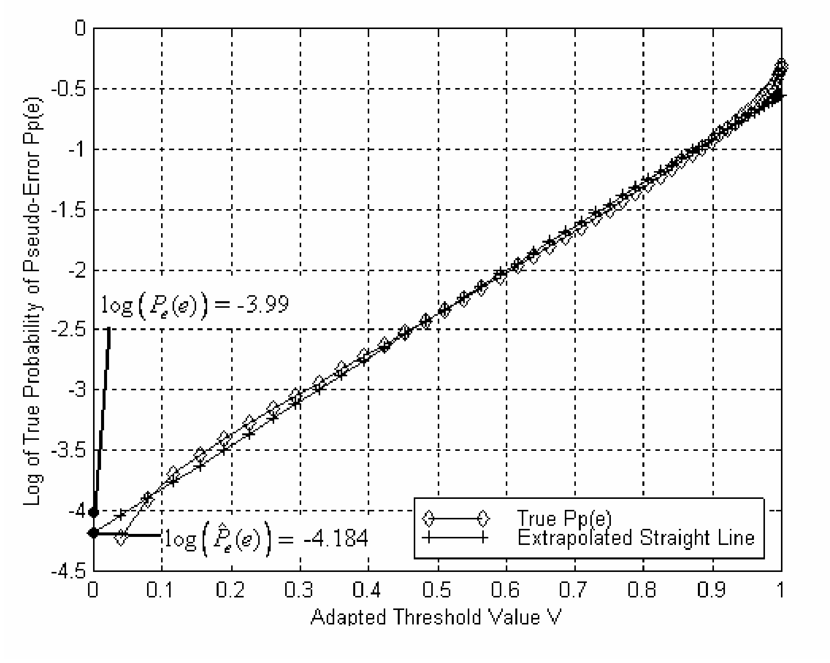


Figure 3.6: Superimposed Graph of the Improved Linear Extrapolated Straight Line that Calculates $\log\left(\hat{P}_e(e)\right)$, and the Log Probability of Pseudo-Error vs. Adapted Threshold at $E_b / N_o = 8.4dB$.

3.5 QUADRATIC EXTRAPOLATION IN AWGN

Quadratic extrapolation will attempt to further improve the accuracy of the estimated BER by using a third pseudo-error value to perform a higher-order extrapolation. By doing this, the ‘arch-like’ characteristic of the curves of Fig.3.3, caused by the non-linearity in the pseudo-error vs. threshold characteristic, is compensated for and will thus improve the accuracy of the final BER estimate. We will firstly derive the general expression for the quadratic extrapolation algorithm as follows:

Assume the function that describes the relationship between the log of the true probability of pseudo-error rate $P_p(e)$ vs. the threshold d is defined by the quadratic polynomial:

$$\log\left(P_p(e)\right) = \alpha.d^2 + \beta.d + \chi \tag{3.15}$$

In order to solve this general expression, we need to solve the constants α, β and χ in Eq. (3.15). We thus need *A*=3 known pseudo-error values, denoted as $P_{p,1}(e) = P_p(e)\big|_{d=d_1}$,

$P_{p,2}(e) = P_p(e)\big|_{d=d_2}$ and $P_{p,3}(e) = P_p(e)\big|_{d=d_3}$. Assume that each of these values satisfy Eq. (3.15). We now define the three pseudo-error values as:

$$P_p(e)\big|_{d=d1} = P_{p,1}(e) = \alpha.(d_1)^2 + \beta.d_1 + \chi\,, \tag{3.16}$$

$$P_p(e)\big|_{d=d2} = P_{p,2}(e) = \alpha.(d_2)^2 + \beta.d_2 + \chi\,, \tag{3.17}$$

and

$$P_p(e)\big|_{d=d3} = P_{p,3}(e) = \alpha.(d_3)^2 + \beta.d_3 + \chi\,. \tag{3.18}$$

Now, with $P_{p,1}(e)$, $P_{p,2}(e)$, $P_{p,3}(e)$ and d_1, d_2, d_3 known, we observe that Eq.'s (3.16)-(3.18) reduce to three linear equations, which we can define in matrix notation as:

$$\bar{d}.\bar{U} = \bar{P}_p\,, \tag{3.19}$$

with

$$\bar{P}_p = \begin{bmatrix} P_{p,1}(e) \\ P_{p,2}(e) \\ P_{p,3}(e) \end{bmatrix}, \tag{3.20}$$

$$\bar{d} = \begin{bmatrix} (d_1)^2 & d_1 & 1 \\ (d_2)^2 & d_2 & 1 \\ (d_3)^2 & d_3 & 1 \end{bmatrix}, \tag{3.21}$$

and

$$\bar{U} = \begin{bmatrix} \alpha \\ \beta \\ \chi \end{bmatrix}. \tag{3.22}$$

In order to solve the general expression of Eq. (3.15), we thus need to solve $\bar{U}$ from Eq. (3.19) as follows:

$$\bar{U} = \left[\bar{d}\right]^{-1}.\left[\bar{P}_p\right]. \tag{3.23}$$

In Eq. (3.23) $\left[\bar{d}\right]^{-1}$ is known as the inverse of the matrix $\bar{d}$ and is defined by [28]:

$$\left[\bar{d}\right]^{-1} = \frac{1}{\det\left[\bar{d}\right]}.adj\left[\bar{d}\right]. \tag{3.24}$$

In Eq. (3.24), $\det\left[\bar{d}\right]$ is known as the determinant of the matrix $\bar{d}$ and is defined as [28]:

$$\det\left[\bar{d}\right] = d_2.(d_1)^2 + d_1.(d_3)^2 + d_3.(d_2)^2 - d_2.(d_3)^2 - d_3.(d_1)^2 - d_1.(d_2)^2, \tag{3.25}$$

and $adj\left[\bar{d}\right]$ is known as the *adjoint* of the matrix $\bar{d}$ [28]. More explicitly, let $adj\left[\bar{d}\right]$ be as defined in [28]:

$$adj\left[\bar{d}\right] = \begin{bmatrix} d'_{00} & d'_{10} & d'_{20} \\ d'_{01} & d'_{11} & d'_{21} \\ d'_{02} & d'_{12} & d'_{22} \end{bmatrix}. \tag{3.26}$$

Then:

$$\left[\bar{d}\right]^{-1} = \frac{1}{\det\left[\bar{d}\right]}.\begin{bmatrix} d'_{00} & d'_{10} & d'_{20} \\ d'_{01} & d'_{11} & d'_{21} \\ d'_{02} & d'_{12} & d'_{22} \end{bmatrix}. \tag{3.27}$$

Now, in order to find $\log\left(\hat{P}_e(e)\right)$, we only need to find the constant χ of Eq. (3.15), which is the hypothetical y-axis intersection point $\log\left(P_p(e)\right)\Big|_{d=0}$. Therefore, in order to solve χ we only need the elements d'_{02}, d'_{12} and d'_{22} from the adjoint matrix of Eq. (3.26). These elements may be shown to be [28]:

$$d'_{02} = d_3 (d_2)^2 - d_2 (d_3)^2, \tag{3.28}$$

$$d'_{12} = d_1 (d_3)^2 - d_3 (d_1)^2, \tag{3.29}$$

and

$$d'_{22} = d_2 (d_1)^2 - d_1 (d_2)^2. \tag{3.30}$$

Finally, $\log\left(\hat{P}_e(e)\right)$ is defined as follows:

$$\begin{aligned}\log\left(\hat{P}_e(e)\right) = \chi &= \frac{1}{\det\left[\bar{d}\right]}.\left[d'_{02} \quad d'_{12} \quad d'_{22}\right].\begin{bmatrix} P_{p,1}(e) \\ P_{p,2}(e) \\ P_{p,3}(e) \end{bmatrix} \\ &= \frac{1}{\det\left[\bar{d}\right]}.\left[d'_{02}.P_{p,1}(e) + d'_{12}.P_{p,2}(e) + d'_{22}.P_{p,3}(e)\right]. \end{aligned} \tag{3.31}$$

Once again, for illustration purposes, define three arbitrary values for the threshold as $d_1 = 0.3$, $d_2 = 0.5$ and $d_3 = 0.7$ at $E_b / N_o = 8.4dB$. By using Eq. (3.6) we then calculate $\log\left(P_{p,1}(e)\right) = \text{-2.331}$, $\log\left(P_{p,2}(e)\right) = -1.499$ and $\log\left(P_{p,3}(e)\right) = \text{-0.877}$. Finally we use Eq. (3.31) and calculate $\log\left(\hat{P}_e(e)\right) = -3.971$. Fig. 3.7 illustrates the plot of the true log probability of pseudo-error vs. threshold value through the use of Eq. (3.6) for the example above. Superimposed onto this plot is the graph of the parabola that originates from the quadratic extrapolation of the above example that estimates the value of the true log probability of error $\log\left(P_e(e)\right)$. Indicated on the y-intersect of the graph are the values for the estimated log probability of error $\log\left(\hat{P}_e(e)\right)$, as well as the true log probability of error $\log\left(P_e(e)\right)$, calculated with Eq. (3.11). It is very clear, by comparing Fig.'s 3.4 and 3.6 to Fig. 3.7, that quadratic extrapolation renders a great improvement in accuracy over both the classic linear extrapolation technique of section 3.4.1 and the improved linear extrapolation technique of section 3.4.2. for the corresponding chosen threshold values.

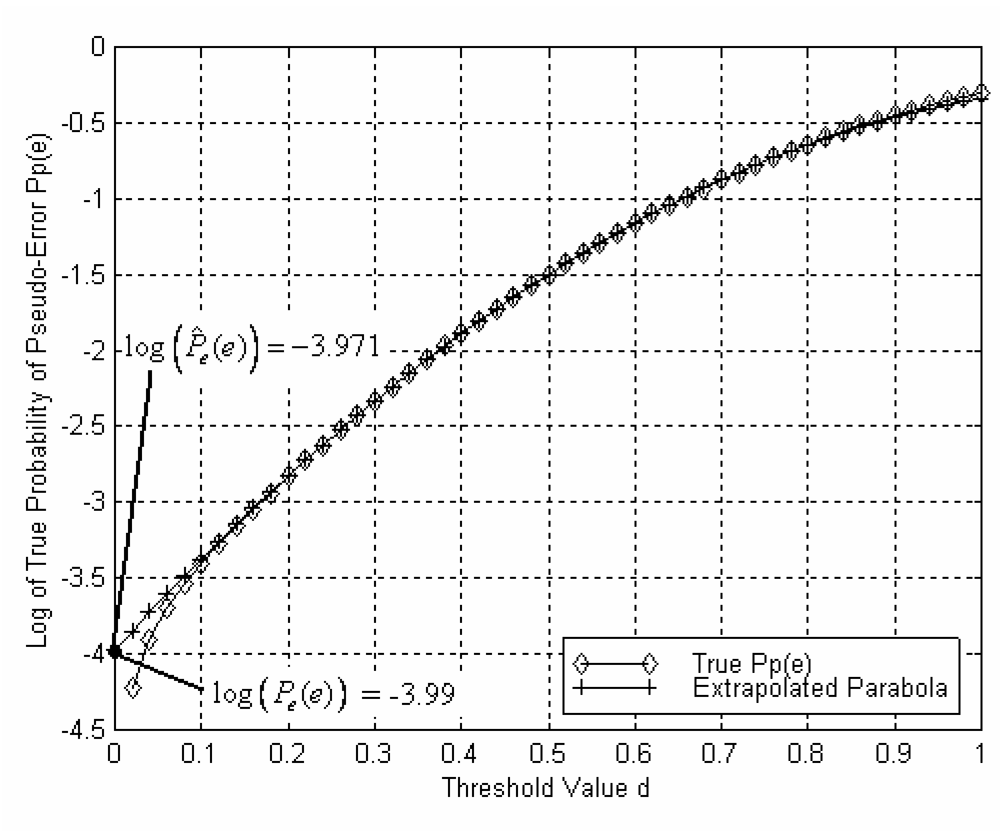


Figure 3.7: Superimposed Graph of the Quadratic Extrapolated Parabola that Calculates $\log\left(\hat{P}_e(e)\right)$, and the True Log Probability of Pseudo-Error vs. Threshold at $E_b / N_o = 8.4dB$.

3.6 CALCULATION ACCURACY AND SPEED OF LINEAR AND QUADRATIC EXTRAPOLATION IN AWGN

In order to properly evaluate the advantages that the above extrapolation techniques have over classic Monte-Carlo analysis, we need to investigate two important attributes that will determine the strength of these techniques, namely: calculation accuracy and calculation speed. It is, however, very important to firstly note the following:

The calculation accuracy and speed attributes of the considered techniques are greatly affected by the choice of the threshold values $(d_1, d_2, ...d_A)$ used at the corresponding modified threshold decision devices (MT_1 to MT_A) of the QoSMU. For the case of linear extrapolation, the following rule applies: The larger the two threshold values d_1 and d_2 (within the range $0 < d_1 < 1$ and $0 < d_2 < 1$), the faster the technique is able to estimate the BER, but at a cost of reduced accuracy. Also, the further apart or the smaller they are chosen to be, the more accurate the estimation technique becomes, but at the cost of increased calculation time. With quadratic extrapolation, the details of these tradeoffs become even more complex to anticipate or predict, because of the addition of the third threshold value needed for extrapolation. Because a detailed investigation into the optimal choice of threshold values for different estimation techniques falls beyond the scope and purpose of this study, it will not be included. Instead, the following assumptions are made:

- When utilizing the classic linear and/or improved linear extrapolation techniques, the values of the two thresholds are always chosen as $d_1 = 0.3$ and $d_2 = 0.7$.
- When utilizing the quadratic extrapolation technique, the values of the three thresholds are always chosen as $d_1 = 0.3$, $d_2 = 0.5$ and $d_3 = 0.7$.

3.6.1 Calculation Accuracy of Linear and Quadratic Extrapolation in AWGN.

Sections 3.4 and 3.5 gave a short insight into the estimation accuracy and improvement of the different extrapolation techniques. However, in order to properly evaluate the accuracy of these techniques, we need to investigate the calculation performance of these techniques in AWGN over a wide range of E_b / N_o values. We will firstly look at the accuracy of the classic linear, improved linear and quadratic extrapolation techniques by comparing their estimation performance to that of the true probability of error for QPSK signals in classic AWGN, as defined by Eq. (3.11). Fig. 3.8 firstly illustrates the plot of the true log probability of error $\log\left(P_e(e)\right)$ vs. E_b / N_o (dB) for $-5dB < E_b / N_o < 15dB$, calculated by Eq. (3.11). Superimposed onto this plot are the graphs of the estimated log probability of error vs. E_b / N_o (dB) resulting from the three extrapolation techniques of sections 3.4 and 3.5, for the same range of E_b / N_o values.

In order to properly investigate the calculation accuracy performance of these three techniques we will have a closer look at the degree of accuracy by calculating the difference between the technique's estimated value and the value of the true probability of

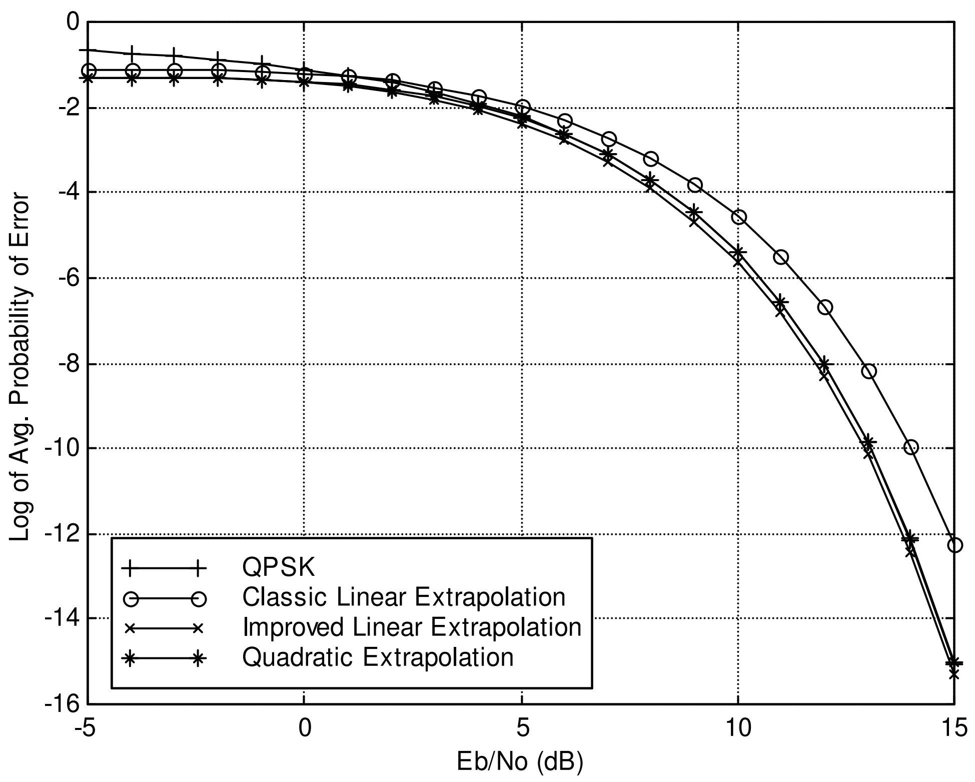


Figure 3.8: Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to True Probability of Error of QPSK in AWGN.

error as depicted by Eq. (3.11) at a specific value of E_b / N_o. We therefore denote this difference value as the *error performance* ε_p of the technique, and define it as:

$$\varepsilon_p = \log\left(\hat{P}_e(e)\right) - \log\left(P_e(e)\right) \tag{3.32}$$

Note that this error value is also a logarithm because of the subtraction of two logarithmic values. Fig. 3.9 illustrates a superimposed plot of the error performance of each of the estimation techniques in AWGN over the range $-5dB < E_b / N_o < 15dB$ as calculated by Eq. (3.32). This figure shows that improved linear extrapolation and quadratic extrapolation are superior in error performance over that of classic linear extrapolation. From the graph it can be seen that improved linear and quadratic extrapolation has an error within approximately $\sqrt{10}$ of the true probability of error for a relatively large range of

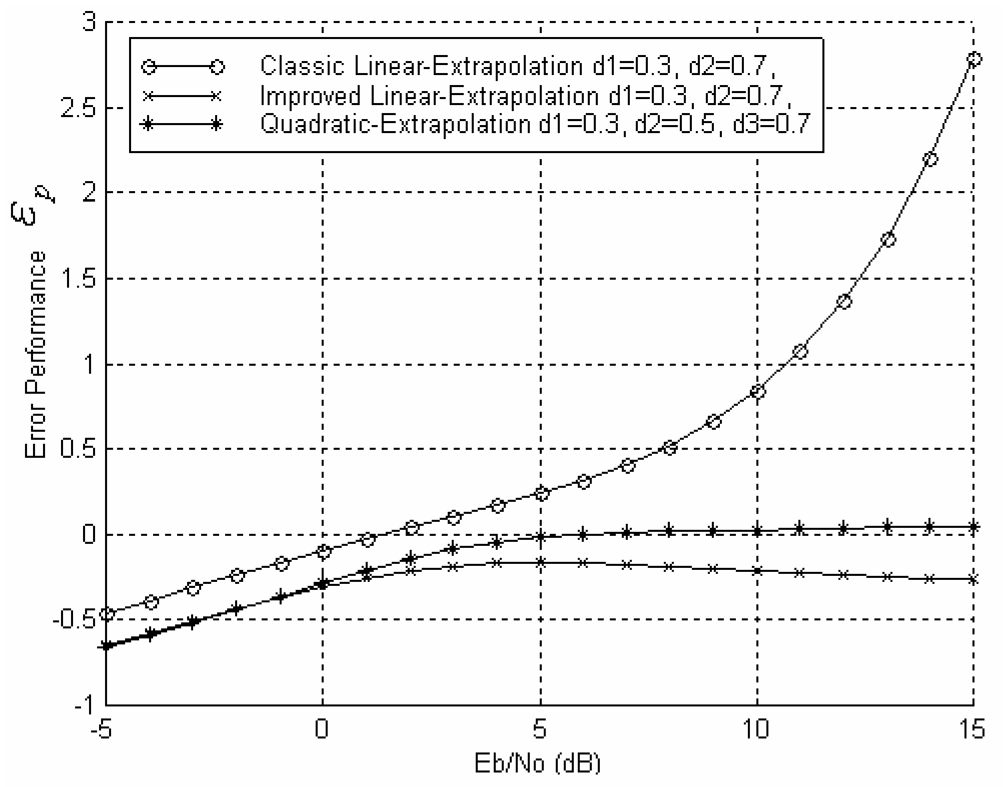


Figure 3.9: Error Performance ε_p vs. E_b/N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation in AWGN

E_b/N_o, whilst classic linear extrapolation has somewhat extreme deviations within this range.

3.6.2 Calculation Speed of Linear and Quadratic Extrapolation in AWGN.

During a real-time BER measurement process, the speed at which a technique is able to estimate or measure a particular BER value is an important attribute that determines how frequently an adaptive system is able to sense and adapt to changes. Determining the BER through a classic Monte-Carlo process, described in section 3.2, requires that the number of errors (or pseudo-errors) be counted until a statistically significant number of errors is reached [29, 30]. This value is then divided by the number of bits received, yielding the required BER or PER value. If the received soft bit values are firstly passed through a classic decision device of which the threshold is set at zero (d = 0), the final calculated

BER value is an estimate of the true probability of error of the system, denoted as $\hat{P}_e(e)$. For the sake of clarity, we name this process the *classic Monte-Carlo* method.
Because a proper error rate value can only be calculated by the classic Monte-Carlo method as soon as the number of errors reaches a predetermined value, the first approach to evaluating the speed of the estimation technique will be to calculate the average time needed for the technique to reach this specific number of errors and compare it to that of the classic Monte-Carlo method. It is also important to note the assumption that the number of errors that are counted is the same for both the classic Monte-Carlo and the relevant extrapolation technique at a specific E_b / N_o. However, because this average time value is (amongst other variables) a function of the bit rate f_b of the system, a better approach would be to evaluate the calculation speed of an extrapolation technique in terms of the average number of received bits needed to obtain a significant number of errors. Also, because this significant number is always a constant value, this approach reduces to simply evaluating the average number of bits needed to obtain one error, which, in essence, is the inverse of the average error rate or simply probability-of-error. For example (from Fig. 3.7), the classic Monte-Carlo method needs an average of approximately $10^4 = 10000$ received bits in order to obtain one error at $E_b / No = 8.4dB$ in a QPSK system.

We now firstly assume that during the estimation process of the three mentioned extrapolation techniques of sections 3.4 and 3.5, an estimate can only be calculated when all of the pseudo-error counters ($PER\_Counter_1$ to $PER\_Counter_A$) in the QoSMU reach a predetermined value. In section 3.2 it was also observed that the log of the true probability of pseudo-error rate $\log\left(P_p(e)\right)$ increases with an increase in the threshold value *d*. Now, in the light of these assumptions and observations, it is logical to conclude that the pseudo-error counter that uses the output from the lowest threshold decision device will be the slowest to reach this predetermined constant error value. Thus, the speed at which an extrapolation technique is able to calculate the final estimated BER value is bounded by the calculation speed of the lowest pseudo-error value $P_{p,1}(e) = P_p(e)\big|_{d=d_1}$. It is thus logical to conclude that the three extrapolation techniques under observation are able to calculate the required BER value at a faster rate than the classic Monte-Carlo method, because of their inherent technique of pseudo-error generation that accelerates the rate at which errors occur (see section 3.2), if it is assumed that the number of errors that are counted is the same for both the classic Monte-Carlo and the relevant extrapolation technique at a specific E_b / N_o.

In order to properly evaluate the calculation speed performance of an extrapolation technique, we will have to evaluate its speed of calculation relative to that of the classic Monte-Carlo technique. Thus, we will determine 'how much faster' the technique is to that of the classic Monte-Carlo technique. A good approach to evaluating the speed of an extrapolation technique would thus be to calculate its speed relative to that of the classic Monte-Carlo by means of a ratio. In light of all that was explained in this section thus far, we define the calculation speed performance $\hat{s}_p$ of an extrapolation technique as the ratio:

$$s_p = \frac{P_{p,1}(e)}{P_e(e)}. \tag{3.33}$$

Note that s_p is a dimensionless number since it only represents the factor by which the lowest pseudo-error rate $P_{p,1}(e)$ is larger than the true error-rate $P_e(e)$. Note that, whenever the true probabilities are not available, estimates thereof are calculated trough a Monte-Carlo process, and Eq. (3.33) changes to:

$$s_p = \frac{\hat{P}_{p,1}(e)}{\hat{P}_e(e)} \tag{3.34}$$

Now, since the pseudo-error value $P_{p,1}(e)$ is the same for all three extrapolation techniques, due to the equal lower threshold choice $d_1 = 0.3$, we can assume that all three techniques will have the same speed performance. As an example, we observe from Fig. 3.7 that, to obtain an error in AWGN at $E_b / N_o = 8.4dB$ at the lowest threshold value, $d_1 = 0.3$, of the quadratic extrapolation technique, we need an average of approximately $10^{(2.3)} \approx 200$ received bits, whereas with the classic Monte-Carlo method we need approximately $10^4 = 10000$ bits. The extrapolation technique is therefore (according to Eq. (3.33)) $s_p = 50$ times faster than the classic Monte-Carlo method. However, this is not constant for all values of E_b / N_o. We therefore present Fig. 3.10, which illustrates a plot of the log calculation speed performance $\log(s_p)$ as a function of E_b / N_o over the range $-5dB < E_b / N_o < 15dB$ for all three extrapolation techniques.

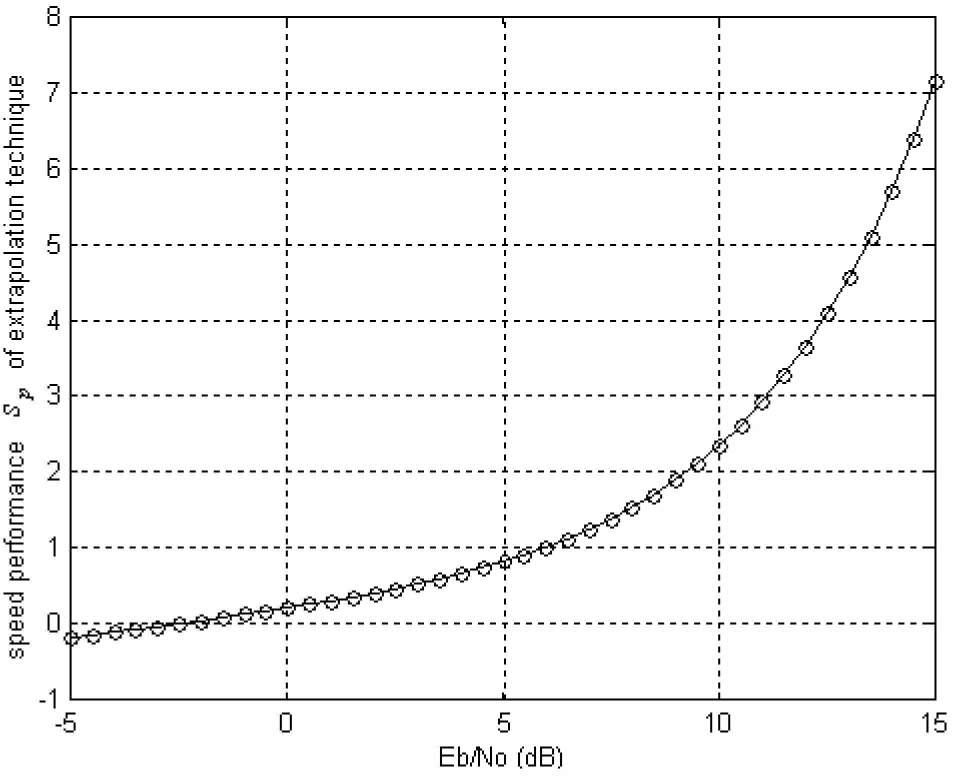


Figure 3.10: Log of Calculation Speed Performance s_p vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation in AWGN.

Fig. 3.10 illustrates that there is an exponential increase in the log calculation speed performance as E_b / N_o increases. It is also interesting to note from Fig. 3.10 that in the range $-5dB < E_b / N_o < -2.5dB$ the log calculation speed is smaller than zero, revealing that the extrapolation techniques are slower than the classic Monte-Carlo method within this range.

3.7 CONCLUDING REMARKS

This chapter presented the general structure of a QoSMU that is used to monitor the uncoded BER performance of a system without affecting the main data throughput. This chapter also reviewed the linear extrapolation technique proposed by *Gooding* [12] that accelerates the speed of monitoring BER performance. Improved extrapolation techniques were also derived and presented in this chapter. Finally, the two most important performance characteristics, namely accuracy and speed of the proposed extrapolation techniques were derived, presented and evaluated under classic AWGN conditions. Novel contributions made in this chapter are:

1. A proper derivation of the general expression for the quadratic extrapolation technique was presented in section 3.5.
2. Relative to all given references, a unique standard for evaluating the speed performance of a PER-extrapolation technique was proposed in section 3.6.2.

CHAPTER FOUR

ADAPTIVE CHANNEL CODING

4.1 OVERVIEW

The main objective of channel coding is to protect transmitted data against the undesirable degrading effects of a non-ideal transmission channel through added redundancy in the transmitted signal. This is one of the few alternatives to improving the error performance in a digital communication system without having to increase the signal power. The existences of such codes were postulated by Shannon's famous channel capacity coding theorem [1]. The channel capacity theorem flows directly from this theorem, and states that if a channel has a finite capacity, and a source generates information at a rate less than the channel capacity, then there exists a coding technique that enables an arbitrarily low probability of symbol error at the receiver.

There are mainly three categories of channel codes: block codes, convolutional codes and coded modulation. From here, there are endless subdivisions, including linear and non-linear, binary and non-binary codes as well as systematic and non-systematic codes. The taxonomy of these subgroups is not important to this study, so only the relevant codes to this study are mentioned. These are: the binary Hamming (7,4,3) block code, the (15,7,5) binary *Bose-Chadhuri-Hocquenghem* (BCH) block code and a rate 1/3 *Non-Systematic* (NS) binary convolutional code.

Encoder Structures: There are unique encoder structures for each type of channel code and a multitude of literature is available on each. Therefore, only an introductory mathematical description on the encoder-implementation of each of the relevant codes used in this study will be given.

Decoder Structures: As mentioned earlier, decoding of channel codes can be very processing–intensive and many current adaptive coding systems suffer from the fact that large blocks of decoders have to be readily available for each type of block code used in the system. This is not cost-effective and, in some cases impossible, because of the complexity involved. Recent work by *Staphorst* [3] suggests a generic *Maximum Likelihood* (ML) decoder structure for all types of codes, including binary and non-binary block codes. An introductory mathematical description of this decoder structure is given in this study. This decoder structure will be the only one employed throughout this study. Decoders using soft-decision input to operate, prove to have an average of 3dB improvement on the uncoded BER performance of the system [6] and because of the structure of the QoSMU (see Chapter 1, section 1.1.2 and Chapter 3), this study will include an application of a *Soft-Output Viterbi Algorithm* (SOVA) to the decoder deliver a soft output.

This chapter starts off with a concise description of the three channel coding schemes used in this study, namely: Binary Hamming (7,4,3) block code, (15,7,5) *Bose-Chadhuri-Hocquenghem* (BCH) block code and a Rate 1/3 *Non-Systematic* (NS) binary convolutional code. This discussion will include all relevant and important channel coding parameters, as well as illustrations of the encoder structures. This section will then be followed by a short description of the generic ML decoder structure for block and convolutional codes, as proposed by *Staphorst* [3], used in the decoding process. This is followed by a simplified explanation and graphical illustration of the SOVA by *Hagenauer* [39], employed by the decoder, that will ensure a soft output, which is needed for the proper functionality of the QoSMU, described in Chapter 3. Lastly, this chapter will discuss the structure of a *Code Decision Device* (CDD) that will decide which coding scheme to implement, based on the real-time estimated BER measurement results of the QoSMU, creating a system that is adaptive to changes in channel conditions. Fig. 4.1 is a simplified illustration of the position of all the relevant functional units of the system described thus far, as well as the additional units that will make the system adaptive. Note that the QoSMU described in Chapter 3 is now placed after the decoder unit, i.e. it functions on the soft decoder outputs.

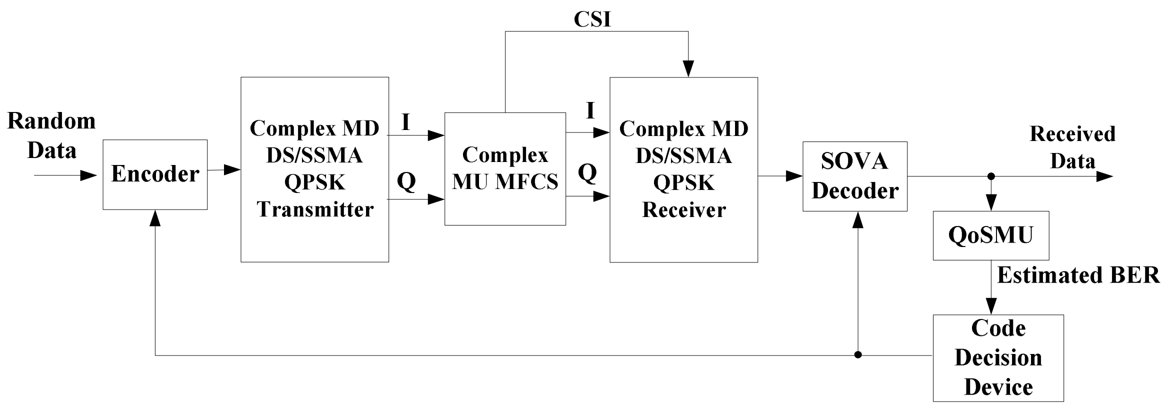


Figure 4.1 Functional Composition of the Adaptive Channel Coded Communication System.

4.2 RELEVANT CODES USED IN THIS STUDY

4.2.1 (15,7,5) Binary BCH Code

Binary *Bose-Chaudhuri-Hocquenghem* (BCH) codes are part of the main group of *Forward Error Correction* (FEC) codes known as *linear block codes*. Furthermore, BCH codes are part of a special subgroup of linear block codes known as *cyclic codes*. Cyclic codes satisfy the following property [6]: if $[c_{\varsigma-1}, c_{\varsigma-2}, ..., c_1, c_0]$ is a code word from a cyclic code, then $[c_{\varsigma-2}, c_{\varsigma-3}, ..., c_0, c_{\varsigma-1}]$, obtained through the cyclic shift of the elements of the code word, is also a code word, where *c* is the binary bit value and ς is the total number of code word bits. Thus, all cyclic shifts of code words are also code words. Because of this, these codes exhibit a simple structure that can be exploited during encoding and decoding processes. A code word from a *systematic* cyclic block code is comprised of two parts: δ information bits and $\varsigma-\delta$ parity check bits. Such a code word can be constructed by firstly passing the δ information bits, followed by the $\varsigma-\delta$ parity check bits. We can define the generation of a codeword in terms of polynomials. We first define a generator polynomial as [6]:

$$g(p) = g_{\varsigma-\delta} p^{\varsigma-\delta} + g_{\varsigma-\delta-1} p^{\varsigma-\delta-1} + ... + g_1 p + g_0 . \tag{4.1}$$

For briefness, we will not discuss the details of the origin of this polynomial, but simply accept this fact and supply the relevant polynomial to the code later in this section. We also define the message or information sequence in terms of a degree $\delta - 1$ polynomial defined by [6], namely:

$$Z(p) = z_{\delta-1}.p^{\delta-1} + z_{\delta-2}.p^{\delta-2} + ... + z_0 . \qquad (4.2)$$

The process of obtaining a codeword polynomial $c(p)$ defined by [6]:

$$c(p) = c_{\varsigma-1}.p^{\varsigma-1} + c_{\varsigma-2}.p^{\varsigma-2} + ... + c_0 , \qquad (4.3)$$

is executed as follows [6]:

a.) Multiply the message polynomial $Z(p)$ by $p^{\varsigma-\delta}$.

b.) Obtain the remainder polynomial $u(p)$ from the division of $p^{\varsigma-\delta}.Z(p)$ by $g(p)$.

c.) Add $u(p)$ to $p^{\varsigma-\delta}.Z(p)$ to obtain [3]:

$$c(p) = p^{\varsigma-\delta}.Z(p) + u(p) . \qquad (4.4)$$

The above process can, however, be simplified and realized through the use of feedback registers. The $\varsigma - \delta$ parity check bits of a systematic cyclic linear block code can be generated by a feedback shift register, based on the generator polynomial of Eq. (4.1). Fig. 4.2 illustrates the general structure for the encoder that generates the $\varsigma - \delta$ parity check bits, based on the generator polynomial [6]. The encoder operates as follows: Initially the shift register contains all zeros. The δ message bits are clocked into the shift register one bit at a time, beginning with the least significant bit. For the first δ shifts, switch 1 is in the closed position. After the δ information bits are clocked into the register as well as into the channel. The two switches then change their position, and at that instant, the shift register contains the needed $\varsigma - \delta$ parity check bits. These $\varsigma - \delta$ parity bits are then clocked out of the shift register and passed to the modulator. The details on the derivation of the structure of this encoder are trivial and will not be expanded on further (see [6] for details). It is however, important to note that for binary codes, $g_0 = g_{\varsigma-\delta} = 1$ and all the summing processes are performed in modulo-2 arithmetic.

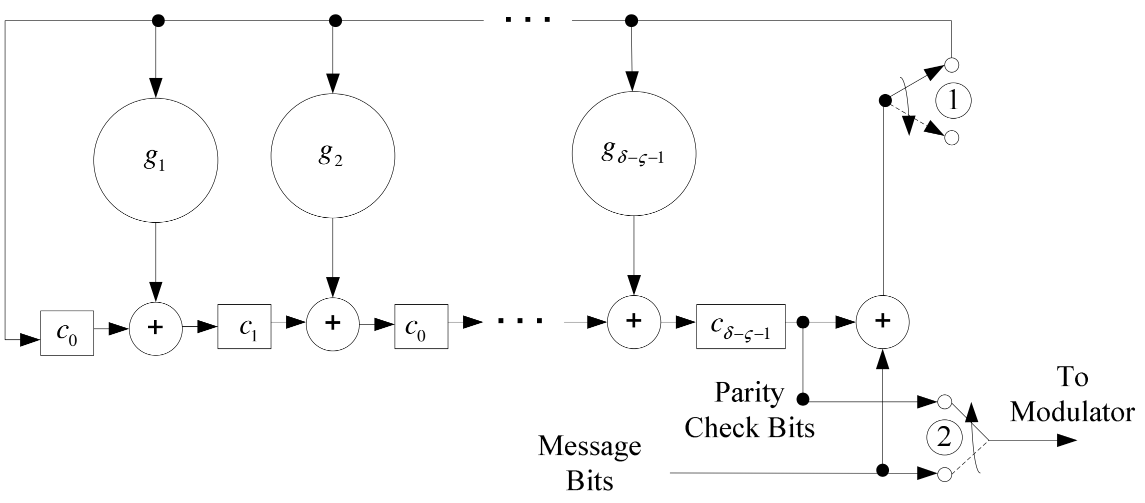


Figure 4.2 General Encoder Structure for the Cyclic Code Based on the Generator Polynomial.

Generally, a specific BCH code is denoted as a $(\varsigma, \delta, d_{\min})$ BCH code with ς and δ as described above, and $d_{\min}$ denotes the minimum distance of the code and is defined by [6]:

$$d_{\min} = 2c_{correct} + 1, \tag{4.5}$$

with $c_{correct}$ the number of correctable errors per code word. Also, the number of parity bits added to an information word during encoding is defined by [6]:

$$\varsigma - \delta \leq m.c_{correct}, \tag{4.6}$$

with $m \geq 3$ an arbitrary integer value. The block length or total number of bits per code word ς is defined by [6]:

$$\varsigma = 2^m - 1. \tag{4.7}$$

For the (15,7,5) BCH code of this study, the generator polynomial is given by [6]:

$$g(p) = p^8 + p^7 + p^6 + p^4 + 1, \tag{4.8}$$

and from this we construct the encoder structure as illustrated in Fig. 4.3.

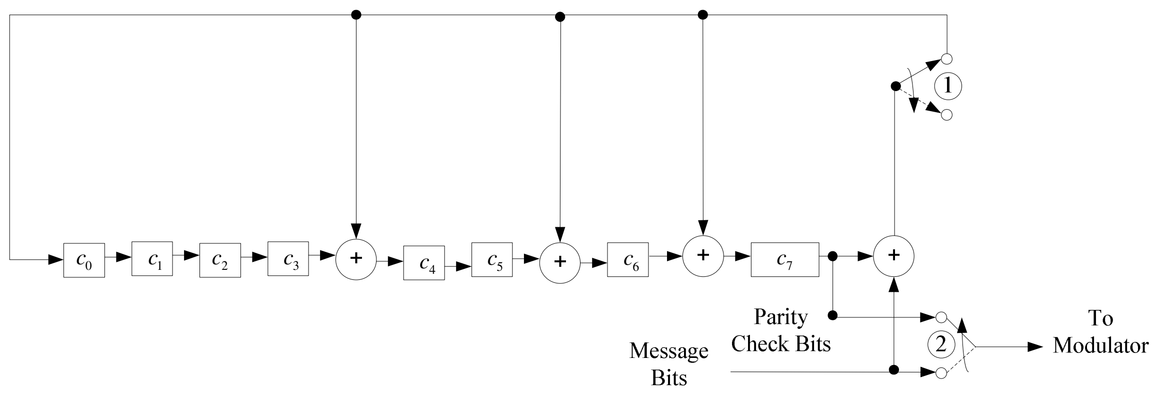


Figure 4.3 Encoder Structure for a (15,7,5) BCH Code.

For the above mentioned code, the relevant parameters are calculated by Eq.'s (4.5) to (4.7) as: $d_{\min} = 5$, $m = 4$, $\varsigma = 15$, $\delta = 7$ and $c_{correct} = t = 2$.

4.2.2 (7,4,3) Binary Hamming Code

Hamming codes can also be classified under the subclass of linear block codes, namely *cyclic codes*. A $(\varsigma, \delta, d_{\min})$ Hamming code has the following special properties:

Block length: $\varsigma = 2^m - 1$ (4.9)

Number of message bits: $\delta = 2^m - m - 1$ (4.10)

Number of parity bits: $\varsigma - \delta = m$ (4.11)

with $m \geq 3$ an arbitrary integer value. Similar to the BCH code of section 4.2.1, a Hamming code generator can be realized through the use of a feedback register and is constructed using the generator polynomial. The generator polynomial for the (7,4,3) Hamming code of this study is given by:

$$g(p) = p^3 + p + 1, \tag{4.12}$$

and from this we construct the encoder structure illustrated by Fig. 4.4.

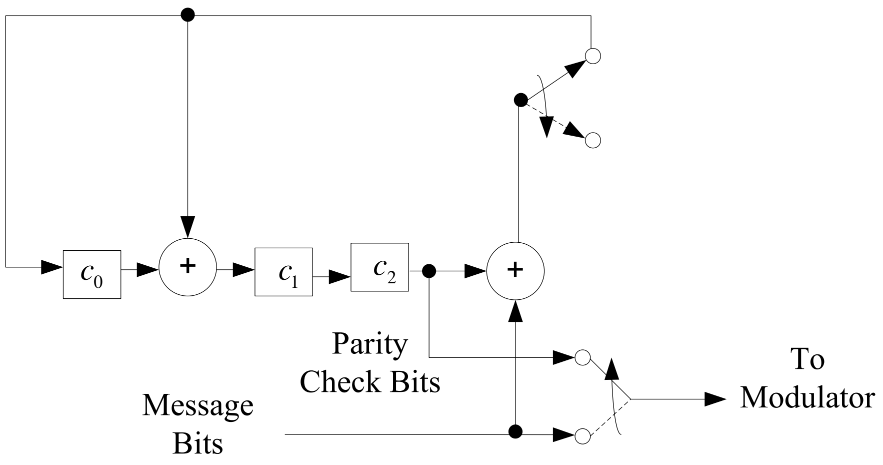


Figure 4.4 Encoder Structure for a (7,4,3) Hamming Code

4.2.3 Rate 1/3 Binary Convolutional Code

A rate δ / ς binary convolutional code is a code where δ binary information bits are passed through a series of linear finite-state shift registers that perform a series of modulo-2 convolutions, to create a binary ς-bit output with $\varsigma > \delta$. The shift registers consist of S_T (δ-bit) stages and ς algebraic function generators that perform modulo-2 addition, where S_T is known as the constraint length of the code. Fig. 4.5 illustrates the encoder structure for the rate 1/3 convolutional code used in this study.

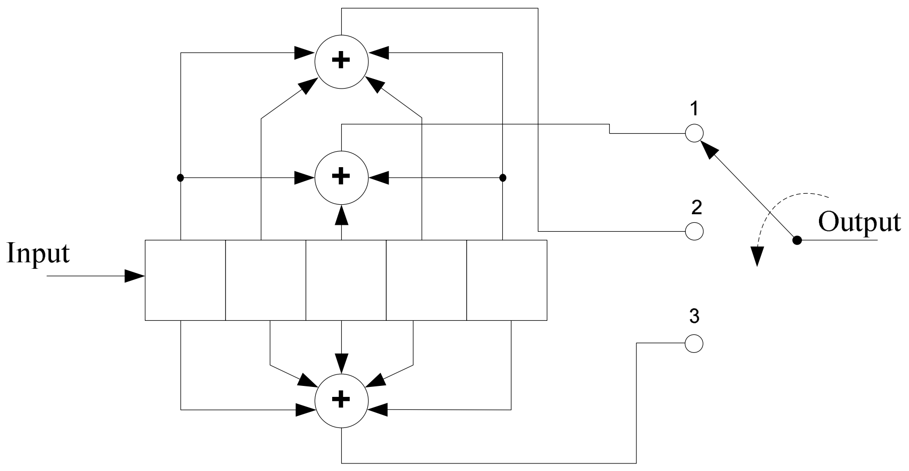


Figure 4.5 Encoder Structure for a Rate 1/3 Convolutional Code with $\delta = 1$, $S_T = 5$ and $\varsigma = 3$ used in this Study.

Note that for this code, $\delta = 1$, $S_T = 5$ and $\varsigma = 3$. The generators for this code are defined by [6] $S_1 = [10101]$, $S_2 = [11011]$ and $S_3 = [11111]$

4.3 GENERIC ML DECODER USED IN THIS STUDY

This section will present a short description of the decoder used in this study. This decoder is based on the generic ML decoder suggested by *Staphorst* [3] which uses the *Viterbi Algorithm* (VA) for decoding binary block codes as well as binary convolutional codes. It is important to note that because VA decoding of binary convolutional codes is such a well established field of study, no attention will be given to it. This section will thus only provide a summarized explanation of the procedures involved in the block-wise VA decoding of linear binary block codes, first suggested by *Wolf* [41] and investigated by many authors [42-50]. It will then suggest an adaptation to ensure that this decoder produces a soft-output value for the proper operation of the QoSMU, based on the SOVA proposed by *Hagenauer* and *Hoeher* in [39].

4.3.1 Construction of a Linear Block Code Trellis

4.3.1.1 Unexpurgated Linear Block Code Trellis Construction

One technique for the decoding of linear binary block codes requires the use of the code's parity check matrix. The parity check matrix of a code is a $(\varsigma - \delta)$-by-ς matrix defined by [20]:

$$\mathbf{H} = \left[\mathbf{I}_{\varsigma-\delta} \vdots \mathbf{P}^T \right], \tag{4.13}$$

with $\mathbf{I}_{\varsigma-\delta}$ the $(\varsigma - \delta)$-by-$(\varsigma - \delta)$ identity matrix, and $\mathbf{P}^T$ a $(\varsigma - \delta)$-by-δ coefficient matrix that is unique to the specific code. The VA decoding of received binary code words require that the code's trellis be constructed. The process of constructing the unexpurgated trellis of a binary block code was presented in detail by *Staphorst* in [3], and also requires the code's parity check matrix, defined by Eq. (4.13). This process is a tedious task whereby all paths (valid and invalid) have to be constructed on a $(2^{\varsigma-\delta})$-by-$(\varsigma+1)$ grid of nodes. A detailed description of this process can be found in [3]. For illustration purposes, it will suffice to provide an example of an already constructed trellis of the VA decoding algorithm in section 4.3.2. Fig. 4.6 illustrates the trellis of the binary Hamming (7, 4, 3) code used in this study, that was constructed through the trellis construction process described by *Staphorst* in [3]. The parity check matrix of the binary Hamming (7, 4, 3) code is given by [20]:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{4.14}$$

Note that the trellis-grid contains $(2^{\varsigma-\delta}) \times (\varsigma+1)$ nodes and the branch bit values are bipolar.

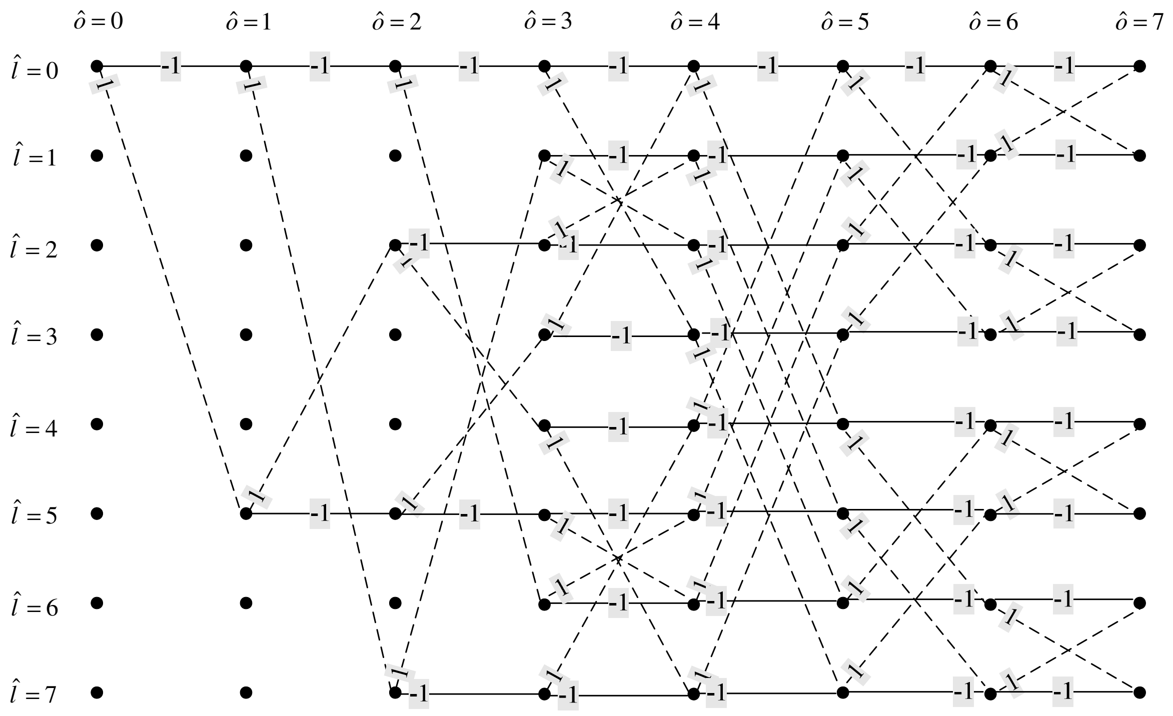


Figure 4.6 Unexpurgated Trellis of the Binary Hamming (7, 4, 3) Code

Each node is indexed by $(\hat{l}, \hat{o})$, with $\hat{o} = 0,1,2,3,...,(\varsigma-1),\varsigma$ and $\hat{l} = 0,1,2,3,...,(2^{\varsigma-\delta}-1)$, with index $\hat{l}$ the *row index*, and $\hat{o}$ the *column index*. From Fig.4.6, if we define a code word path as all possible combinations of branches between nodes $(0,0)$ and $(\hat{l}, \varsigma)$, then the trellis resulting from this procedure will contain more paths than valid code words. The number of paths generated by this procedure is much larger than 2^{δ} when in fact there are only 2^{δ} valid code words. For example, the number of possible paths contained in the trellis of Fig. 4.6 is larger than the $2^{4}=16$ valid code words in the binary Hamming (7, 4, 3) code. Also note that the input value of each branch between two nodes is indicated on each branch, whilst the output value of that branch is indicated by a solid line (when the output value is -1), or a dotted line (when the output value is 1).

4.3.1.2 Expurgation of a Linear Block Code Trellis

Trellis expurgation is a process whereby all non-code word paths of the previously constructed trellis, with code word symbols from the extended Galois field $GF(2^{\xi})$, with $\xi > 1$, are removed [3]. This process simply involves the discarding of all paths that do not end in node $(0,\varsigma)$. In the resulting trellis there are only $2^{\xi.\delta}$ valid code word paths. *Staphorst*, *Buttner* and *Linde* also presented a simplified expurgation algorithm for removing all branches from the nodes in set $\hat{o}-1$ entering the node $(\hat{l},\hat{o})$ for $\hat{o}=\delta+1,\delta+2,\delta+3,...,\varsigma$ and $\hat{l}=2^{\xi.(\varsigma-\hat{o})},2^{\xi.(\varsigma-\hat{o})}+1,...,2^{\xi.(\varsigma-\delta)}$ for binary and non-binary block codes in [31] and [32], respectively. *Staphorst* [3] also presented a method for reducing an already expurgated trellis. Fig. 4.7 illustrates the expurgated trellis of the previously constructed binary Hamming (7, 4, 3) code trellis example of section 4.3.1.

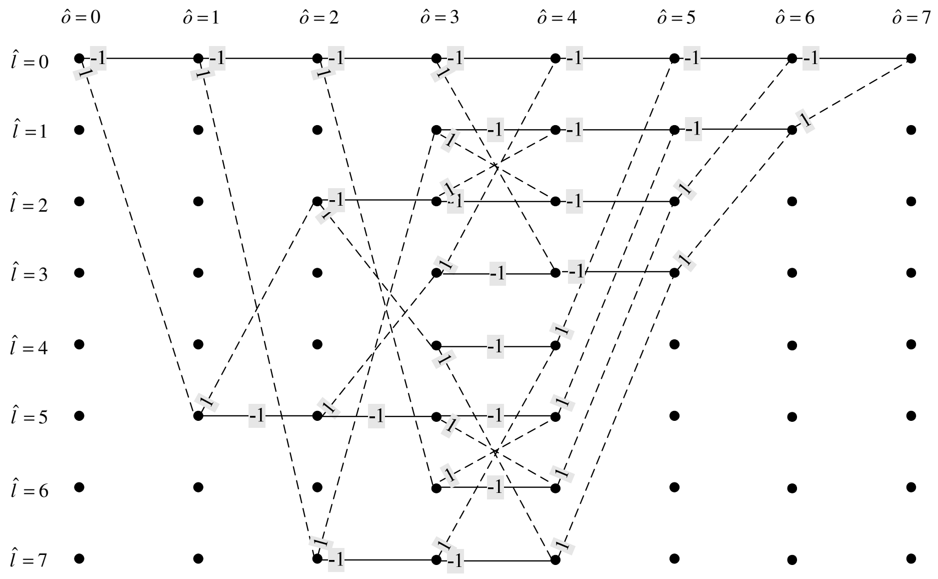


Figure 4.7 Expurgated Trellis of the Binary Hamming (7, 4, 3) Code

4.3.2 Block-Wise Viterbi Decoding Algorithm

As stated in [3], if $P(\overline{v}|\overline{c})$ is defined as the probability of receiving the code word $\overline{v}$, conditioned on the transmitted codeword $\overline{c}$, then the probability of decoding error is minimized if the log-likelihood function $\ln\left(P(\overline{v}|\overline{c})\right)$ is maximized. The VA moves from trellis depth 0 to ς and systematically discards paths that do not maximize the overall log-likelihood function, $\ln\left(P(\overline{v}|\overline{c})\right)$. This section will give a concise explanation of the process of block-wise VA decoding of linear block codes and will use the expurgated trellis of the binary Hamming (7, 4, 3) code of the previous section as an example. However, before the process can be properly explained, a few concepts have to be introduced:

Definition 1

Branch Weight: A branch weight is defined as the binary input bit value, (indicated on each trellis path) between two nodes, and is denoted by $W_{(\hat{l}',\hat{o}-1),(\hat{l},\hat{o})}$, with $(\hat{l}',\hat{o}-1)$ the node of origin (previous node), and $(\hat{l},\hat{o})$ the destination node (current node). For example, in Fig. 4.7 the branch weight $W_{(0,0),(5,1)}$ between nodes (0,0) and (5,1) is 1, whilst the branch weight $W_{(0,0),(0,1)}$ between nodes (0,0) and (0,1) is -1.

Definition 2

Branch Output: A branch output is defined as the hard-decision output value on the branch between two nodes, and is denoted by $O_{(\hat{l}',\hat{o}-1),(\hat{l},\hat{o})}$, with $(\hat{l}',\hat{o}-1)$ the node of origin, and $(\hat{l},\hat{o})$ the destination node. For example, in Fig. 4.7 the branch output $O_{(0,0),(5,1)}$ between nodes (0,0) and (5,1) is 1 (indicated by a dotted line), whilst the branch weight $O_{(0,0),(0,1)}$ between nodes (0,0) and (0,1) is -1 (indicated by a solid line).

Definition 3

Branch Metric: The branch metric is defined as the received code word bit value $\overline{v}_{\hat{o}}$ multiplied by the branch weight as follows:

$$BM_{(\hat{l}',\hat{o}-1),(\hat{l},\hat{o})} = (\overline{v}_{\hat{o}}).\left(W_{(\hat{l}',\hat{o}-1),(\hat{l},\hat{o})}\right), \tag{4.15}$$

with $(\hat{l}',\hat{o}-1)$, the index for the node of origin, and $(\hat{l},\hat{o})$ the index for the destination node.

Definition 4

Cumulative Metric: A cumulative metric is defined as the sum of the previous cumulative metric and the current branch metric between the node of origin $(\hat{l}',\hat{o}-1)$, and the destination node $(\hat{l},\hat{o})$ as follows:

$$CM_{(\hat{l}',\hat{o}-1),(\hat{l},\hat{o})} = CM_{(\hat{l}',\hat{o}-1)} + BM_{(\hat{l}',\hat{o}-1),(\hat{l},\hat{o})} \tag{4.16}$$

Definition 5

Survivor Path: A survivor path is the path from a node of origin $(\hat{l}',\hat{o}-1)$, to a destination node $(\hat{l},\hat{o})$ that results in the largest cumulative metric.

The VA decoding process of a linear block code can now be summarized as follows:

1. Start at node (0,0) and assign to this node a cumulative metric of $CM_{(0,0)} = [0]$.
2. Start at the leftmost column depth of the trellis $(\hat{o}=0)$, and move through the trellis stepwise in single increments $(\hat{o}=0,1,2,3...,(\varsigma-1),\varsigma)$. At each column of the trellis, perform the following at each node:
   a.) For each node within the current column depth (i.e. for each node with row index $\hat{l}=0,1,2,3,...,(2^{\varsigma-\delta}-1)$) that contains incoming paths, calculate the branch metric for all incoming paths using Eq. (4.15).
   b.) Using the calculated branch metrics of step (a), calculate the cumulative metrics using Eq. (4.16) for each node within the current column depth, and decide on the resulting survivor path for each.
3. When the rightmost node $(0,\varsigma)$ is reached, and all metric calculations and decisions have been performed, find the *Maximum Likelihood* (ML) path by combining all the survivor paths, starting from the rightmost node, and moving stepwise through to the leftmost node of the trellis. The final output code word is

now given by the branch output value on each of the branches of the ML path combined from left to right.

Fig. 4.8 illustrates an example of a hard-decision output decoding process, using the expurgated trellis of Fig. 4.7 and a soft-decision received code word vector $\overline{v} = [-1.3 \quad -0.4 \quad 0.3 \quad -1.1 \quad -0.5 \quad -0.9 \quad -1.6]$.

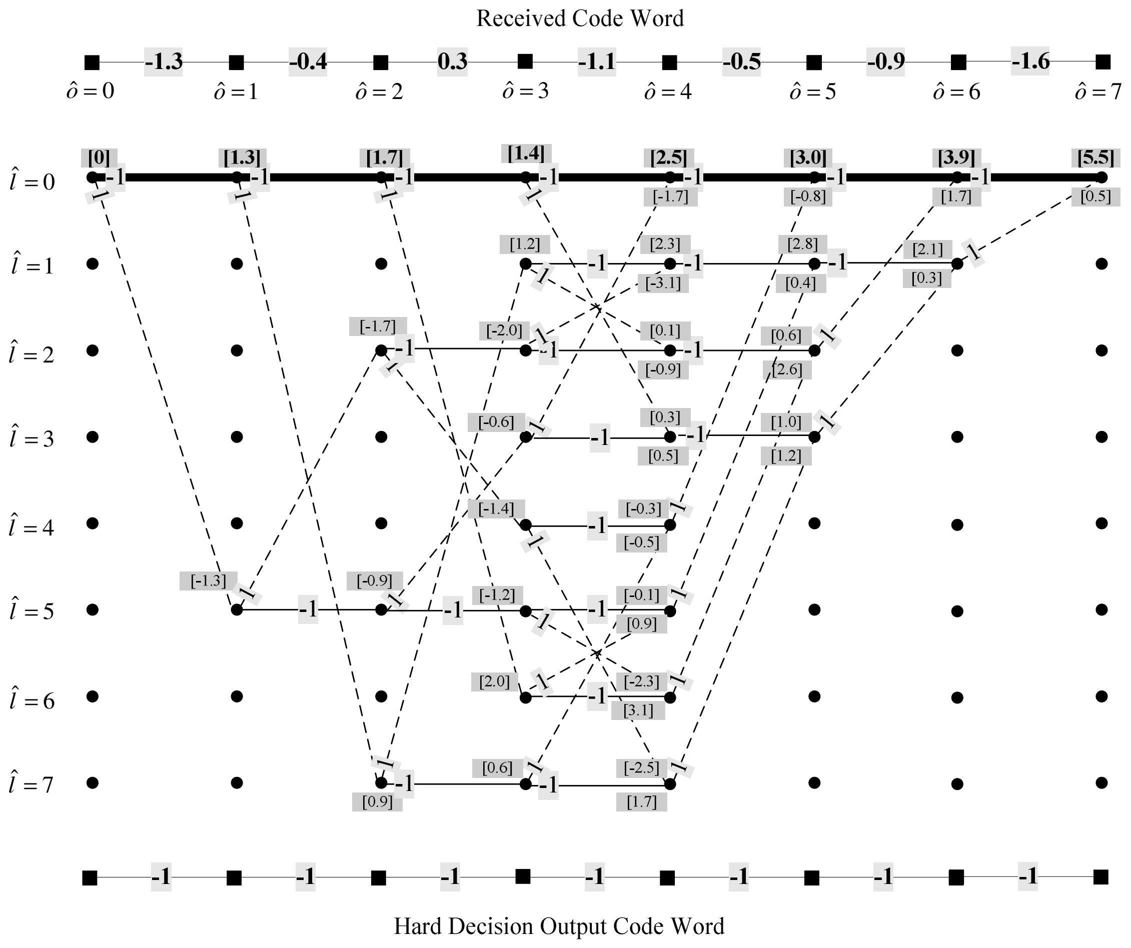


Figure 4.8 Hard-Decision Output VA Decoding Example for the Binary Hamming (7, 4, 3) Code.

Note that this received code word corresponds to the transmitted all-zero code word $\overline{c} = [-1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1]$, with a potential bit-error in position 3. In Fig. 4.8, all the cumulative metrics for each node are shown. The cumulative metric is displayed in bold if it is part of the final ML path. The final ML path containing the correct code word bit between each branch is also shown in bold. The correct bipolar VA decoded code word can be seen at the bottom of the figure. Note that when this block has been decoded, the next decoding process is repeated for a new received block of data. Therefore the VA for linear block codes operates in a block-wise fashion, in contrast to the traditional sliding window approach for the case of convolutional codes.

4.3.3 Application of a Soft-Output Viterbi Algorithm to Block Codes.

We recall from Chapter 3, section 3.2, that the QoSMU requires a soft-input value in order to operate properly. We will therefore need to use an adapted algorithm inside the decoder that ensures the calculation of soft-output valued code words. This section will use the well-known SOVA for soft-decision VA decoding of convolutional codes, as proposed in a ground-breaking paper by *Hagenauer* and *Hoeher* in [39]. This algorithm is based on calculating a reliability measure for each code word bit and using this as the soft-output value. The theory behind this algorithm is also properly summarized and published in a book by *Hanzo* in [40]. The principles of this algorithm will be applied to *Staphorst's* VA decoding of block codes [3]. Because the derivation of the SOVA principles falls beyond the scope of this study, only a simplified explanation of the operation of the SOVA will be given. A more detailed explanation of the theory behind the SOVA can be found in [39] and [40]. The application of the SOVA on binary block codes will now be graphically illustrated by extending the VA decoding example of section 4.3.2. The SOVA decoding of block codes can be summarized as follows:

1. Using the relevant expurgated trellis (see section 4.3.1), follow the steps of section 4.3.2 and find the ML path through the trellis, purging all survivor paths that terminate before $\hat{o} = \varsigma$, and retaining those survivor paths that merge with the ML path. For sake of clarity, we will name the paths that merge with the ML path the *contender paths*. Fig. 4.9 illustrates the decoding example of section 4.3.2 for the VA decoding of a binary Hamming (7, 4, 3) block code, containing the ML path and also the necessary retained paths. From this point onward, all nodes that are situated on the ML path will be indexed by $(\hat{l}_{\hat{o}}, \hat{o})^{ML}$, with $\hat{o} = 0,1,2,3,\ldots,(\varsigma-1),\varsigma$ the column index and $\hat{l} = 0,1,2,3,\ldots,(2^{\varsigma-\delta}-1)$, the row index (as defined in section 4.3.1.1).

2. Starting at node $(\hat{l}_1, 1)^{ML}$ on the ML path, step through each consecutive node on the ML path until $(\hat{l}_\varsigma, \varsigma)^{ML}$ is reached. At each node on the ML path that has two incoming paths, calculate a cumulative metric difference $CMD_{(\hat{l}_{\hat{o}},\hat{o})^{ML}}$ as follows:

$$CMD_{(\hat{l}_{\hat{o}},\hat{o})^{ML}} = CM_{(\hat{l}_{\hat{o}-1},\hat{o}-1)^{ML},(\hat{l}_{\hat{o}},\hat{o})^{ML}} - CM_{(\hat{l}',\hat{o}-1),(\hat{l}_{\hat{o}},\hat{o})^{ML}}\,, \tag{4.17}$$

where $CM_{(\hat{l}_{\hat{o}-1},\hat{o}-1)^{ML},(\hat{l}_{\hat{o}},\hat{o})^{ML}}$ is the cumulative metric of the incoming branch of the ML path. Conversely, $CM_{(\hat{l}',\hat{o}-1),(\hat{l}_{\hat{o}},\hat{o})^{ML}}$ is the cumulative metric of the incoming branch of the contender path. Note that the subscripted indexing is consistent with the fact that both incoming branches have the same destination node, situated on the ML path, but distinctively different nodes of origin. For example, in Fig. 4.9, the cumulative metric difference at node $(0,4)^{ML}$, $CMD_{(0,4)^{ML}}$ is calculated as follows:

$$CMD_{(0,4)^{ML}} = CM_{(0,3)^{ML},(0,4)^{ML}} - CM_{(3,3),(0,4)^{ML}} = 2.5-(-1.7) = 4.2\,. \tag{4.18}$$

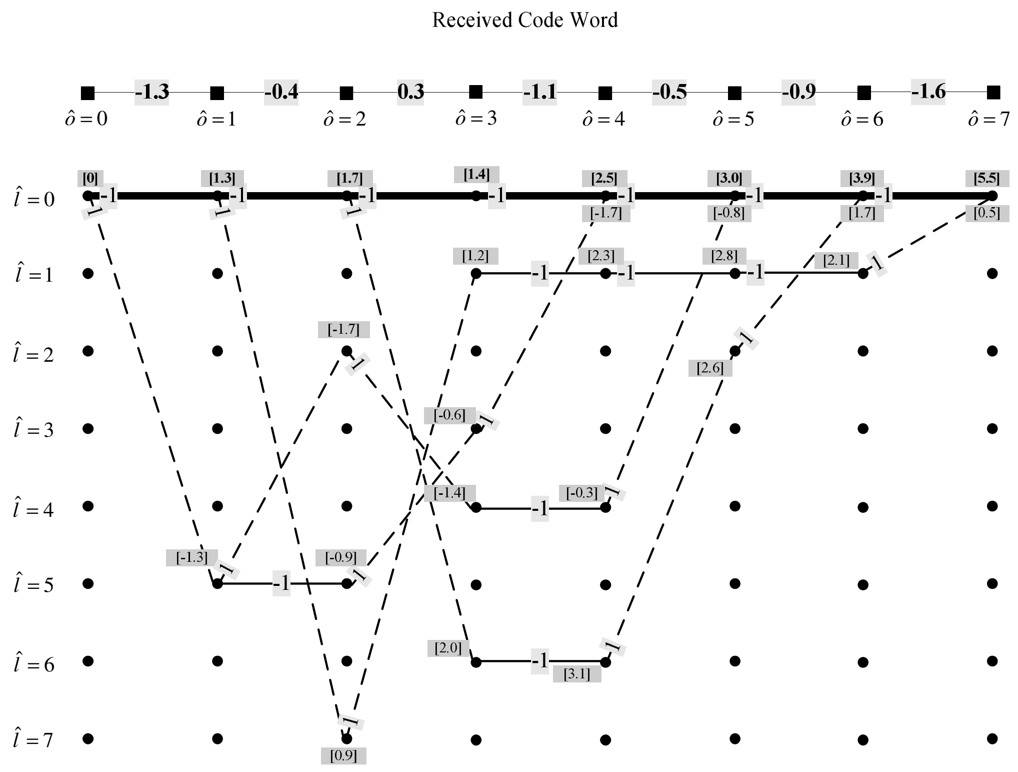


Figure 4.9 VA Decoding Example of Section 4.3.2 for the Binary Hamming (7, 4, 3) Code With All Necessary Paths Purged Except Those that Merge With the ML Path.

This calculation will also ensure that $CMD_{(\hat{l}_{\hat{o}},\hat{o})^{ML}} > 0$. According to the log-likelihood function, if only a single cumulative metric exists (i.e. there is only one incoming path) at the node, the cumulative metric difference is taken as $CMD_{(\hat{l}_{\hat{o}},\hat{o})^{ML}} = \infty$, because the reliability measure for that branch is said to be infinite. Details on the origin and derivation of this log-likelihood function can be found in [40]. Fig. 4.10 shows an extension of Fig. 4.9, and contains the necessary cumulative metric differences at each node on the ML path. These values are indicated in the darkened blocks to the left of the two corresponding cumulative metrics on the ML path. For sake of clarity, from this point forward we will index all ML branches with the index $(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}$. Thus all ML branches are indexed with the same index as the respective destination node $(\hat{l}_{\hat{o}},\hat{o})^{ML}$. For example, using Fig. 4.10, the index of the first branch of the ML path is $(1,1)^{ML}_{Branch}$, whilst the index of

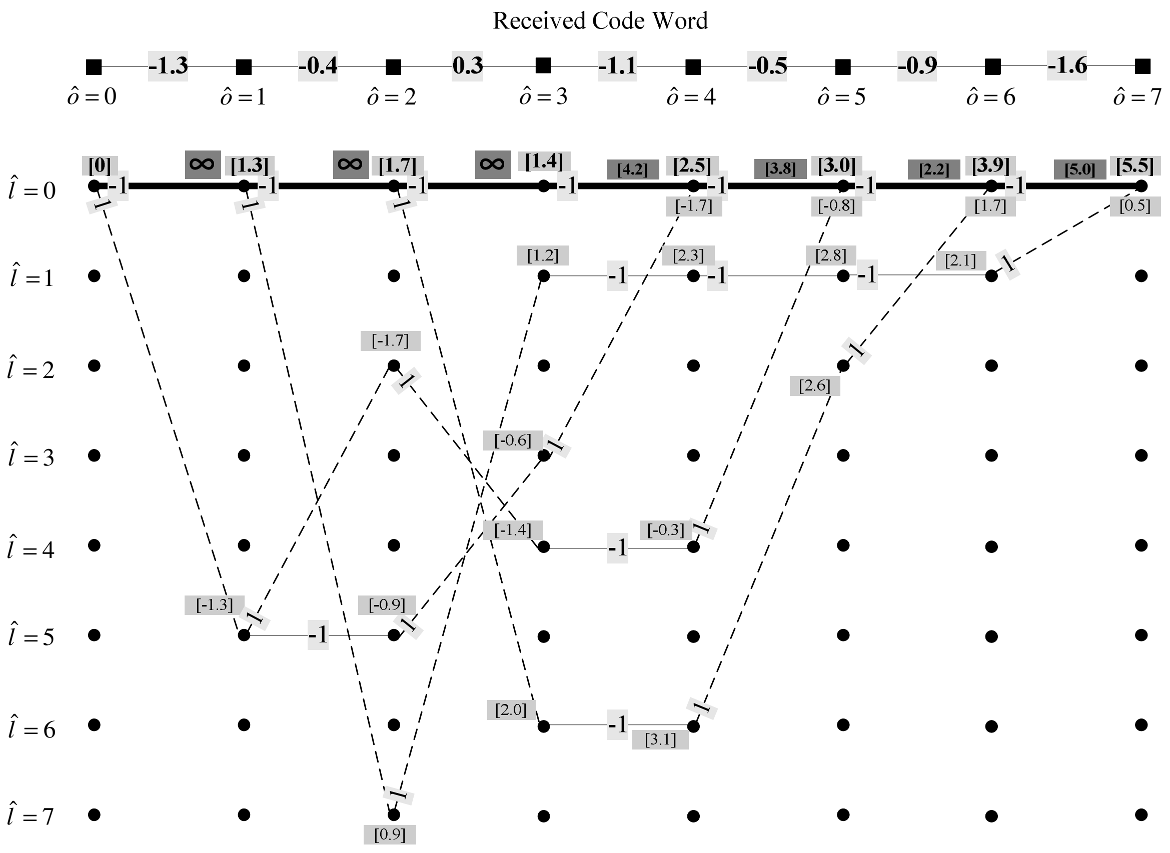


Figure 4.10 Extension of Fig. 4.9 Containing the Calculated Cumulative Metric Differences

the second branch is $(2,2)^{ML}_{Branch}$ etc. Similarly, all cumulative differences will be subscripted with their respective branch indexes.

3. Starting at the branch containing the first cumulative metric difference $CMD_{(\hat{l}_1,1)^{ML}_{Branch}}$ on the first branch of the ML path, moving along the ML path through the trellis in stepwise increments, i.e. $\hat{o}=1,2,3,\ldots,(\varsigma-1),\varsigma$, perform the following:

   a) Including the current branch, determine the branch with the smallest cumulative metric difference on the ML path from the current branch forward through the trellis, i.e find:

$$CMD^{\min}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} = \min\left\{\begin{matrix} CMD_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}, CMD_{(\hat{l}_{\hat{o}+1},\hat{o}+1)^{ML}_{Branch}}, CMD_{(\hat{l}_{\hat{o}+2},\hat{o}+2)^{ML}_{Branch}}, \ldots \\ , CMD_{(\hat{l}_{\varsigma-2},\varsigma-2)^{ML}_{Branch}}, CMD_{(\hat{l}_{\varsigma-1},\varsigma-1)^{ML}_{Branch}}, CMD_{(\hat{l}_{\varsigma},\varsigma)^{ML}_{Branch}} \end{matrix}\right\}, \qquad (4.19)$$

   with $(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}$, the index of the current branch. For example, using Fig. 4.10:

$$CMD^{\min}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} = 2.2 \qquad for \quad \hat{o}=1,2,3,4,5,6 \;, \qquad (4.20)$$

   Whilst

$$CMD^{\min}_{(\hat{l}_7,7)^{ML}_{Branch}} = 5.5\,. \qquad (4.21)$$

   b) Starting at the node where the smallest cumulative metric difference was found (step (a)), trace the alternative path back to the same depth (column index) as that of the current ML branch and compare the current branch output value (indicated by solid or a dotted line) of the alternative path branch with that of the ML path branch. If these two values are equal, repeat step (a), discarding all previously determined $CMD^{\min}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$ values, and using the next smallest cumulative metric difference to determine a new $CMD^{\min}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$. Alternatively, if these values are different, we call this contender path the *optimal contender* *path* for the current ML branch, and we redefine the current $CMD^{\min}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$ as $CMD^{\min,optimal}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$. For example, using Fig. 4.10, we see that $CMD^{\min,optimal}_{(\hat{l}_1,1)^{ML}_{Branch}} \neq 2.2$ because when the alternative path is traced back to the same depth as the current ML branch, it merges with the current ML branch, and the branch output of the equivalent current alternative path branch is therefore equal to that of the ML path branch and the value 2.2 is discarded. However when we repeat this process, using the next smallest cumulative metric difference value of 3.8, we find that this process is successful, resulting in $CMD^{\min,optimal}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} = 3.8$. Finally, if all the above steps were properly followed and all current alternative path branch outputs are equal to that of the ML path branch, set $CMD^{\min,optimal}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} = \infty$, because the reliability measure of the output value on the current branch (in terms of the log likelihood function) is infinite.

c) We finally calculate the soft-output value $SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$ for the current branch as follows:

$$SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} = \left(O_{(\hat{l}_{\hat{o}},\hat{o})^{ML}}\right).\left(CMD^{\min,optimal}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}\right) \quad . \tag{4.22}$$

Fig. 4.11 illustrates an extension of Fig. 4.10, containing the final calculated soft-output values, illustrated at the bottom of the figure.

Finally, to summarize:

Finding the soft-output code word for a binary block code using the SOVA simply involves the following three steps:

1. Find the ML path of the received code word through the expurgated trellis, without purging the contender paths (paths that merge with the ML path).
2. Calculate the metric differences $CMD_{(\hat{l}_{\hat{o}},\hat{o})^{ML}}$ for all branches of the ML path (i.e. for $\hat{o} = 1,2,3,...,(\varsigma-1),\varsigma$).

3. Determine the optimal alternative path and $CMD^{\min,optimal}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$ for all branches of the ML path. Finally, calculate the final soft-output value for each branch of the ML path by using Eq. (4.22).

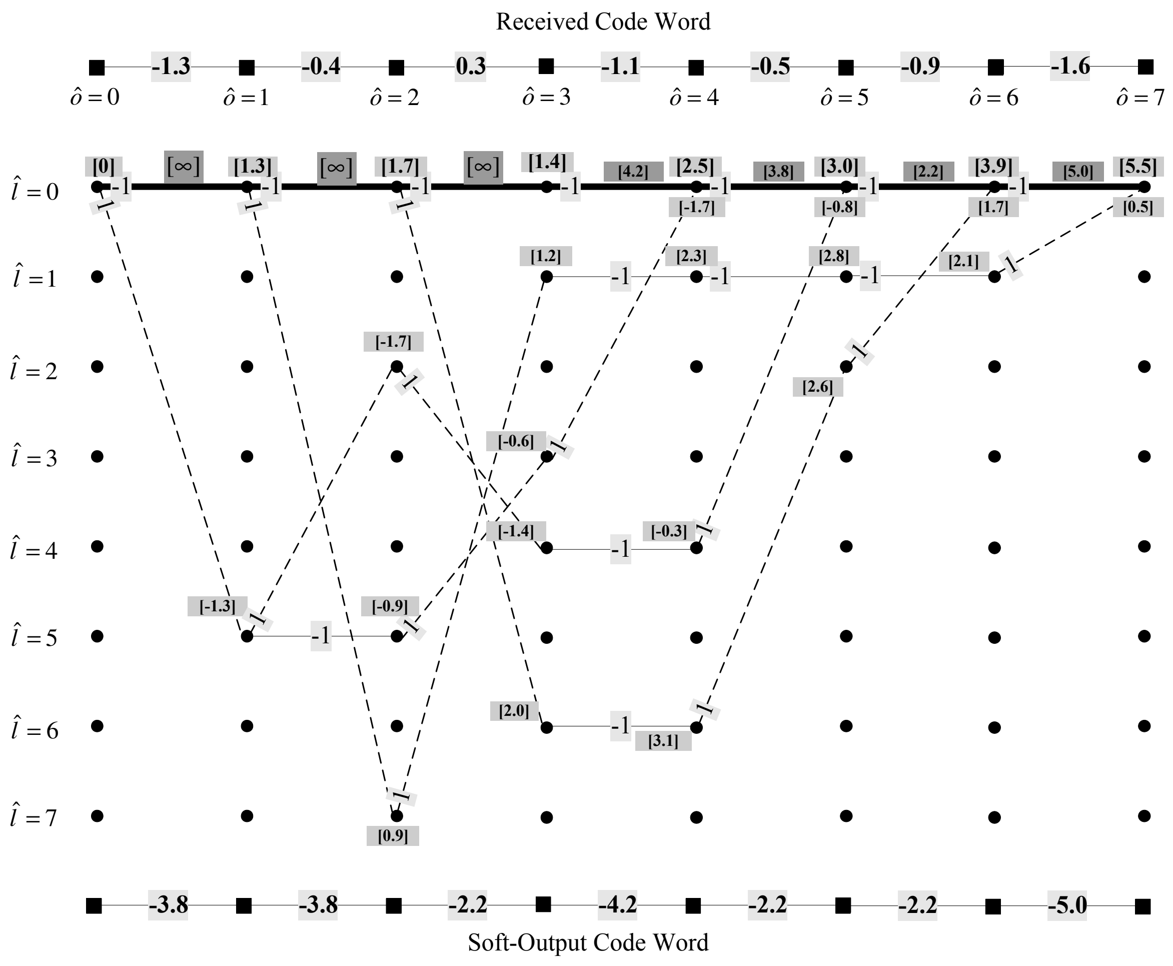


Figure 4.11 Extension of Fig. 4.10 Containing the Calculated Soft Output Code Word

Another requirement for the QoSMU in Fig. 4.1 to function properly, is the fact that it needs the output values from the decoder to satisfy $-1 < SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} < 1$, in order to use the correct threshold values during the threshold modification process (see Chapter 3, section 3.2). We will therefore need to scale the soft-output values $SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$ by some predetermined factor in order to meet this requirement. This process is in actual fact not theoretically possible, because of the possibility of $SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} = \infty$. However, because of the small probability of this condition being satisfied, it can be neglected, resulting in the fact that $\left|SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}\right|$ will have an average (finite) upper bound, depending on the length of the trellis, and type of channel code being implemented. Naturally, we will have to scale the soft-output value $SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$ by this maximum value in order to meet the requirement: $-1 < SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} < 1$. We therefore declare this maximum value, or upper bound, as $SO^{\max}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$, and calculate the final scaled soft-output values (soft-input values to the QoSMU) as:

$$SO^{scaled}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}} = \frac{SO_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}}{SO^{\max}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}} \tag{4.23}$$

Table 4.1 shows the values of $SO^{\max}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$ that are used for the codes of this study.

Table 4.1 Scaling Factor for the Codes Used in this Study

Channel Code	$SO^{\max}_{(\hat{l}_{\hat{o}},\hat{o})^{ML}_{Branch}}$
(7,4,3) Binary Hamming Code	6
(15,7,5) Binary (BCH) Code	10
Rate 1/3 Binary Convolutional Code	24

4.4 CODE DECISION DEVICE

In Fig. 4.1, the CDD is connected after the QoSMU (see Chapter 3). The main function of the CDD is to make a decision as to which code to implement, based on the estimated BER value provided by the QoSMU. The CDD then generates a trigger signal that is passed to the encoder and decoder of the system, simultaneously. In practice, the trigger signal that is fed back to the encoder at the transmitter end of the system occurs through a reverse channel with similar characteristics as the forward channel. In order to simplify the simulation model, this channel was assumed to be perfect, and not included in the model.

Before we explain the algorithm by which the CDD makes its decisions, the region within which the CDD makes these decisions should be explained. Firstly, because we are implementing three codes of different strengths, naturally there are three coding possibilities within the system. However, because we also consider the uncoded state as a possible coding state, we regard it as the fourth coding possibility. Thus, we can now define the four coding possibilities by four individual code modes (0-3):

Code Mode 0 = Uncoded

Code Mode 1 = (7,4,3) Hamming Code

Code Mode 2 = (15,7,5) BCH Code

Code Mode 3 = Rate 1/3 Convolutional Code

We consider code mode 0 as the lowest rated, and code mode 3 as the highest rated code in the proposed adaptive coding scheme.

The aim of an adaptive coded system is to choose an appropriate code mode that will ensure that the overall BER performance of the system is kept within a predetermined region. The choice of this region is governed by the level of BER performance that a specific application is able to endure. This region is also bounded by two thresholds that determine the condition whereby the system should switch to a different code mode. Fig. 4.12 illustrates the optimal BER performance level region, bounded by the two thresholds a and b, with $a < b$, that the system will attempt to maintain throughout operation.

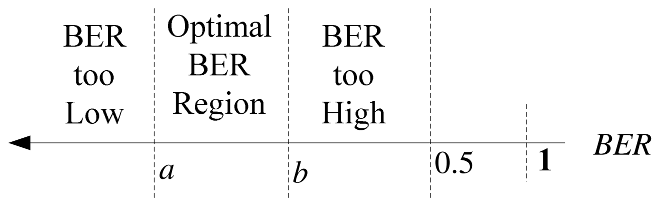


Figure 4.12 Illustration of the Optimal BER Performance Operating Region

Note from Fig. 4.12 that the highest BER is bounded by 0.5. The thresholds a and b should be carefully chosen to avoid system instability, as well as unresponsiveness. For example, if the range $b-a$ is chosen to be too small, the system might enter a situation where the estimated BER value never enters the range $a < BER < b$, which will result in irresolute switching between two code modes. On the other hand, if the range $b-a$ is chosen to be too large, the system will remain in one code mode for too long and will therefore not be able to adapt to the varying channel conditions. Lastly, if the system is already operating at the BER limits of a particular code mode and channel conditions should further degrade (or improve), the system should maintain the current mode. The algorithm that ensures $a < BER < b$ is summarized by the flow chart illustrated in Fig. 4.13.

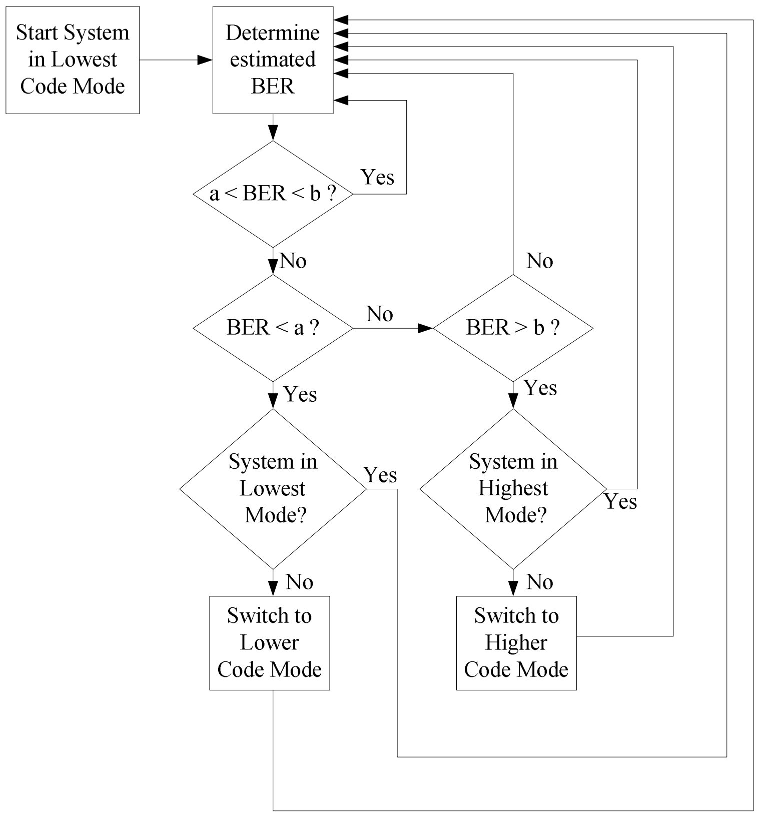


Figure 4.13 Flow Chart of the Algorithm that Ensures Optimal System BER Performance.

4.5 CONCLUDING REMARKS

This chapter presented the general structure of the adaptive communication system. The relevant codes and their encoder structures were summarized, followed by a short section on the generic ML decoder structure for VA decoding of linear block codes, as presented by *Staphorst* [3]. This section was followed by a subsection explaining the application of a SOVA by *Hagenauer* [39] to the VA decoding of linear block codes. Novel contributions made in this chapter are:

1. The principles of the SOVA by Hagenauer [39] was applied to the VA decoding of linear block codes by *Staphorst* [3] and illustrated in section 4.3.3.
2. The structure of a Code Decision Device (CDD) that will ensure proper switching between codes as well as optimal system performance under varying channel conditions was presented in section 4.4.

CHAPTER FIVE

SIMULATION SETUPS AND PARAMETERS

5.1 OVERVIEW

This chapter presents the configurations, as well as all important and relevant system parameters, for the different simulations that will be performed during this study and presented in Chapter 6. The first part of this chapter considers the spectral characteristics of the MD-DS/SSMA transmitter of Chapter 2 and presents the relevant parameters used during the simulation study. This is then followed by a section that considers the simulation setup and relevant parameters for evaluating the spectral characteristics of the frequency-selective fading channel presented in Chapter 2. The next part of this chapter considers the simulation setup for evaluating the coded and uncoded BER performance of the fully functional wideband communication system in a MU-MFC environment. Following this section is the simulation setup and parameters for evaluating the PER vs. threshold characteristics of a coded and uncoded system in a MU-MFC environment, as well as the simulation setup for evaluating the accuracy and speed performance of the three extrapolation techniques of Chapter 3, also in a MU-MFC environment. Lastly, this chapter will present the setup configuration and parameters for the most important simulations of this study, namely the real-time simulation of the fully adaptive coded wideband communication system, whereby the results of the adaptation performance as well as the throughput vs. BER performance of the system, will be presented and discussed in Chapter 6.

5.2 TEMPORAL AND SPECTRAL CHARACTERISTICS OF THE MD-DS/SS TRANSMITTER

Firstly, the temporal output of the transmitter will reveal the temporal characteristics of the signal that will pass through the channel. Secondly, the *Power Spectral Density* (PSD) of the transmitter output is an important measure that reveals how much bandwidth that a particular user's transmitted signal occupies. Chapter 2, Fig. 2.2 illustrated the structure of the MD-DS/SSMA transmitter that use DSB CE-LI-RU filtered GCL sequences for spreading. Table 5.1 contains all relevant parameters and information for the transmitter that will be used during simulation.

Table 5.1 MD-DS/SS Transmitter Parameters

Parameter	**Value / Setting**
CSS Family	DSB CE-LI-RU filtered GCL
Assumed Unspreaded Bit Rate	1 kb/s
Spreading Sequence Length	63 chips
Samples Per Chip	8
Sampling Rate	504 KHz
Simulation RF Carrier	None

It is important to note that, because of the use of the pre-filtered spreading sequences, no additional pulse shaping filters or transmit filters are required by the transmitter.

If we assume that the two sequences from the family being used are perfectly orthogonal, and we assume a QPSK modulation configuration, it can be shown that the base band (BB) bandwidth BW_{sig}^{BB} of the transmitter output signal employing DSB CE-LI-RU filtered GCL sequences can be expressed by [3]:

$$BW_{sig}^{BB} = \frac{1}{2.T_c} = 31500 \quad Hz, \tag{5.1}$$

where T_c is defined as the sequence chip period.

5.3 SPECTRAL CHARACTERISTICS OF THE COMPLEX CHANNEL

This section will firstly present the simulation setups and parameters for the evaluation and verification of the spectral characteristics of the complex FFCS of Chapter 2, section 2.3.1. Thereafter, the simulation setups and parameters for the evaluation and verification of the spectral characteristics of the complex MFCS of Chapter 2, section 2.3.2 will be presented. The last section will present all the simulation setup parameters for when the MFCS is expanded into a MU-MFCS, as presented in Chapter 2, section 2.3.3, after which each user's channel parameters within the system will be properly tabulated.

5.3.1 Spectral Characteristics of the Complex Flat Fading Channel Simulator

Fig. 5.1 illustrates a simplified representation of the simulation setup used for evaluating the spectral characteristic of the complex FFCS. Recall that the functional unit represented by *FFCS* in Fig. 5.1 has been represented in detail in Fig. 2.3 of Chapter 2, section 2.3.1.

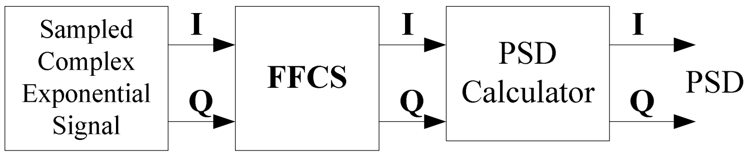


Figure 5.1 Simulation Setup for Evaluating the Spectral Characteristic of the Complex FFCS

Table 5.2 contains all the relevant parameters and information for the simulation setup.

Table 5.2 Simulation Setup Parameters for Evaluating the Spectral Characteristic of the Complex FFCS

Parameter	**Value / Setting**
Input Signal Type	Complex Exponential
Signal Frequency f_c	1 kHz
Sampling Frequency $f_s = \frac{1}{T_s}$	10 kHz
Rician Factor W_k Used	-100 dB
Doppler Spreads $(B_{Doppler})$ Investigated	10 Hz, 50 Hz and 100 Hz

5.3.2 Spectral Characteristics of the Complex Multipath Fading Channel Simulator

Fig. 5.2 illustrates a simplified representation of the simulation setup used for evaluating the spectral characteristic of the complex MFCS. Recall that the functional unit represented by *MFCS* in Fig. 5.2 has been illustrated in detail in Fig. 2.4 of Chapter 2, section 2.3.2. Table 5.3 contains all the relevant parameters and information for the simulation.

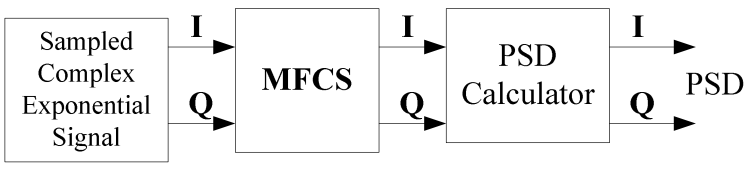


Figure 5.2 Simulation Setup for Evaluating the Spectral Characteristic of the Complex MFCS.

The MFCS of Fig. 5.2 will simulate a three-path model and will thus contain three complex FFCSs ($FFCS_1$ to $FFCS_3$), each with its own individual simulation parameters. Table 5.3 summarizes all the relevant parameters for this simulation setup.

Table 5.3 Simulation Setup Parameters for Evaluating the Spectral Characteristic of the Complex MFCS

Parameter		**Value / Setting**
Input Signal Type		Complex Exponential
Signal Frequency f_c		150 kHz
Sampling Frequency $f_s = \frac{1}{T_s}$		1.5 MHz
Path 1	Path Power $Y(\tau_1)$	-0.479 dB
	Rician Factor W_1	9 dB
	Doppler Spread	100 Hz
	Path Delay τ_1	0.992 μs
Path 2	Path Power $Y(\tau_2)$	-9.845 dB
	Rician Factor W_2	0 dB
	Doppler Spread	50 Hz
	Path Delay τ_2	69.4 μs
Path 3	Path Power $Y(\tau_3)$	-30.479 dB
	Rician Factor W_3	-100 dB
	Doppler Spread	10 Hz
	Path Delay τ_3	220.2 μs

5.3.3 Complex Multi-User Multipath Fading Channel Simulator Parameters

Whenever the MU-MFCS is utilized, each user within the system has its own individual multipath fading channel simulator (MFCS), as illustrated by Fig. 2.5 of Chapter 2, section 2.3.3. The complete multi-user communication system of this study is able to support up to 5 users. Each of the users in the system operates with a MFCS containing 3 paths. Table 5.4 contains the setup parameters for each user's MFCS.

Table 5.4 Simulation Setup Parameters of the Complex MU-MFCS

		User Number				
		1	2	3	4	5
Path 1 Setup	Path Power $Y(\tau_1)$ [dB]	-0.479	-0.4371	-0.3607	-0.2436	-0.1211
	Rician Factor W_1 [dB]	9	9	9	9	9
	Doppler Spread [Hz]	100	50	10	100	50
	Path Delay τ_1 [μs]	0.992	11.9	24.8	32.7	39.7
Path 2 Setup	Path Power $Y(\tau_2)$ [dB]	-9.845	-10.23	-11.03	-12.70	-15.76
	Rician Factor W_2 [dB]	0	0	0	0	0
	Doppler Spread [Hz]	50	100	100	10	10
	Path Delay τ_2 [μs]	69.4	74.40	82.34	91.27	100
Path 3 Setup	Path Power $Y(\tau_3)$ [dB]	-30.48	-30.44	-30.36	-30.24	-30.12
	Rician Factor W_3 [dB]	-100	-100	-100	-100	-100
	Doppler Spread [Hz]	10	10	50	50	100
	Path Delay τ_3 [μs]	220.2	203.4	186.5	173.6	155.8

5.4 UNCODED AND CODED BER PERFORMANCE OF WIDEBAND SYSTEM IN FREQUENCY-SELECTIVE FADING

This section will explain the simulation setup for evaluating the BER performance for the uncoded as well as complete coded, wideband system over a MU-MFCS. The simulation setup of this section will enable the evaluation of the uncoded and coded BER performances of the complete MD-DS/SSMA QPSK communication system over the MU-MFCS presented in Chapter 2, section 2.3.3. Fig. 5.3 illustrates the setup for the simulation.

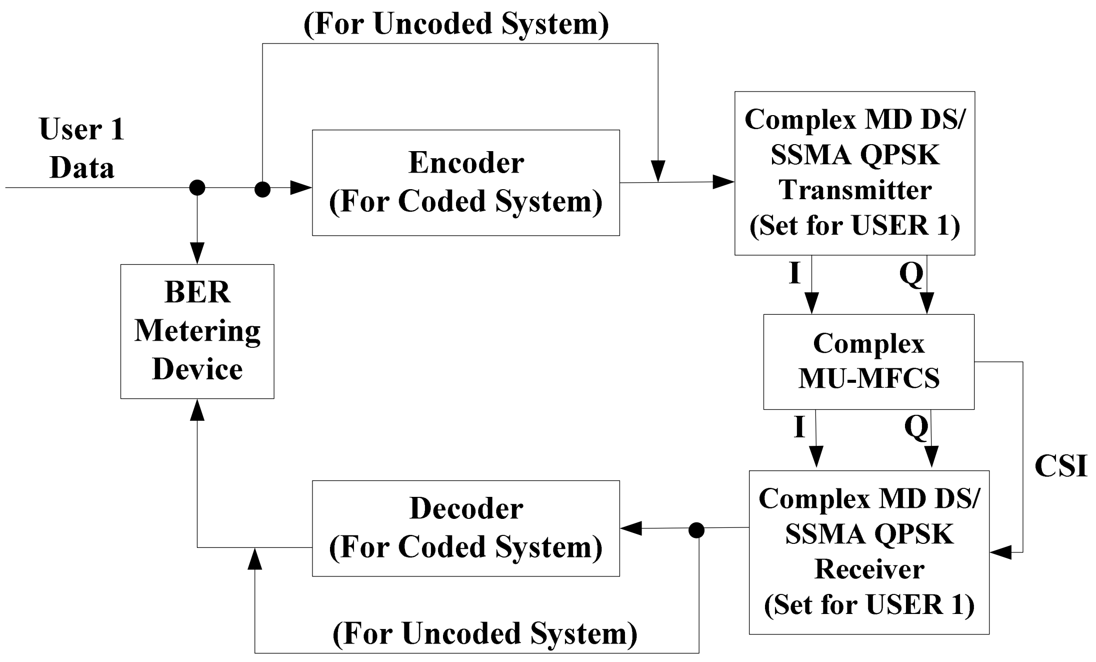

(For Uncoded System)
User 1 Data
Encoder (For Coded System)
Complex MD DS/ SSMA QPSK Transmitter (Set for USER 1)
I
Q
BER Metering Device
Complex MU-MFCS
I
Q
CSI
Decoder (For Coded System)
Complex MD DS/ SSMA QPSK Receiver (Set for USER 1)
(For Uncoded System)

Figure 5.3 Simulation Setup for the Uncoded and Coded BER Performance of the Wideband Communication System

The BER metering device of Fig. 5.3 has the function of determining the BER through the classic Monte-Carlo process described in Chapter 3, section 3.2 and section 3.6.2. Note also that the complex MD-DS/SSMA transmitter in Fig. 5.3 is fully represented by the transmitter structure of Fig. 2.2, explained in Chapter 2, section 2.2, and is set to accept and transmit data from user-1 in the system. Also, the complex QPSK receiver shown in Fig. 5.3 is fully represented by the receiver structure of Fig. 2.6, explained in Chapter 2, section 2.4 and is also configured to receive the data of user-1 in the system that was sent by the corresponding transmitter. Finally, the MU-MFCS shown in Fig. 5.3 is fully represented by the channel model structure of Fig. 2.5, explained in Chapter 2, section 2.3.3 and has the parameter configuration presented in Table 5.4 of section 5.3.3. For the case of a coded system, the encoder and decoder of Fig. 5.3 is fully represented by one of the three encoder and matching decoder structures of Chapter 4, section 4.1, that correspond to one of the three codes considered in this study.

5.5 PER vs. THRESHOLD CHARACTERISTIC FOR THE UNCODED AND CODED WIDEBAND SYSTEM OVER MU-MFCS

This section will explain the simulation setup for evaluating the PER vs. threshold value characteristic for the uncoded as well as complete coded wideband system over a MU-MFCS. The concepts and details of this characteristic was explained in Chapter 3, section 3.3, whereby the uncoded PER vs. threshold characteristic for a narrowband system in AWGN was presented. Fig. 5.4 illustrates the simulation setup for evaluating the PER vs. threshold value characteristic for both the coded and uncoded wideband system over the MU-MFCS of this study.

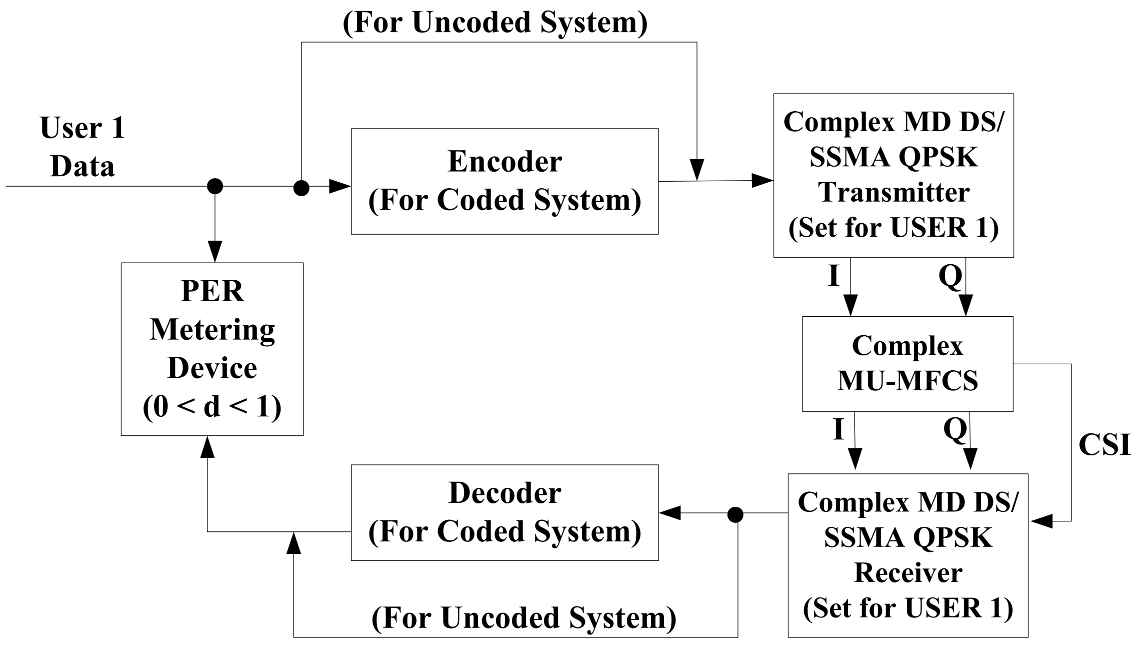


Figure 5.4 Simulation Setup for both the Coded and Uncoded PER vs. Threshold Characteristic of the Wideband Communication System

In this simulation the probability of PER is estimated by the PER metering device shown in Fig. 5.4, by calculating the PER through the Monte-Carlo process described in Chapter 3, section 3.2. For the case of a coded system, we recall that the PER metering device requires a soft output from the decoder in order to be able to modify the threshold and conclusively calculate the PER. The details of how the decoder calculates a soft output value is fully described in Chapter 4, section 4.3.3. The rest of the functional units has the same parameters as described throughout section 5.4.

5.6 CALCULATION ACCURACY AND SPEED OF LINEAR AND QUADRATIC EXTRAPOLATION FOR UNCODED AND CODED WIDEBAND COMMUNICATION SYSTEM OVER MU-MFCS

This section will present the simulation setups for evaluating the calculation accuracy and speed of the linear and quadratic extrapolation techniques of Chapter 3 for both the uncoded and coded wideband systems in a MU-MFCS environment.

5.6.1 Calculation Accuracy of Linear and Quadratic Extrapolation

Fig. 5.5 illustrates the simulation setup for the evaluation of the accuracy performance of linear and quadratic extrapolation techniques for both the uncoded and coded wideband systems in a MU-MFCS environment.

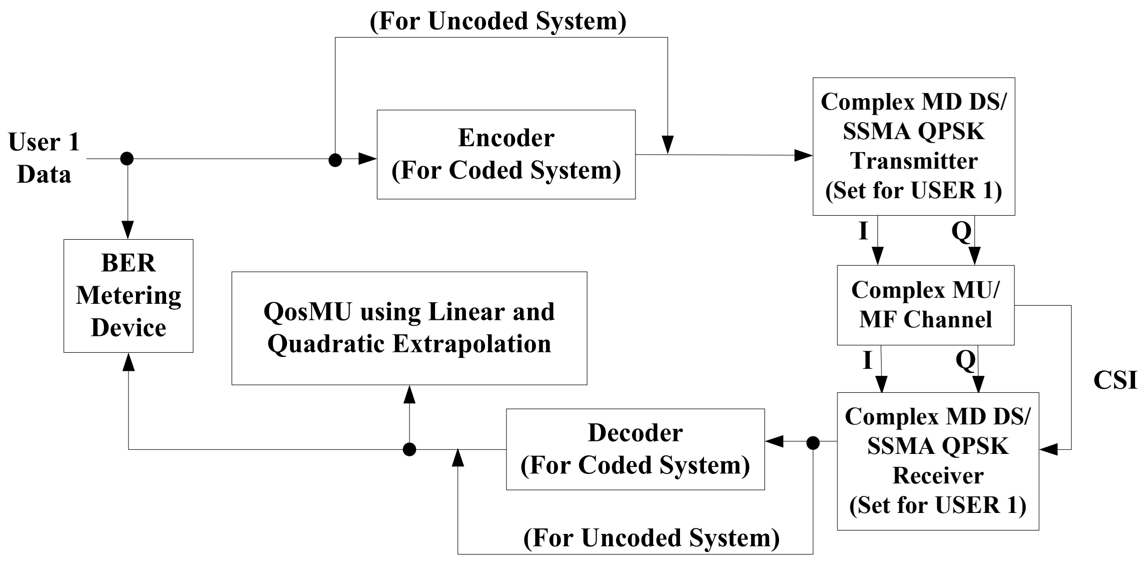

(For Uncoded System)
User 1 Data
Encoder (For Coded System)
Complex MD DS/ SSMA QPSK Transmitter (Set for USER 1)
I
Q
BER Metering Device
QosMU using Linear and Quadratic Extrapolation
Complex MU/ MF Channel
I
Q
CSI
Decoder (For Coded System)
Complex MD DS/ SSMA QPSK Receiver (Set for USER 1)
(For Uncoded System)

Figure 5.5 Simulation Setup for the Evaluation of the Accuracy Performance of the Linear and Quadratic Extrapolation Techniques for both the Uncoded and Coded Wideband Systems in a MU-MFCS Environment..

In this simulation, the QoSMU of Fig. 5.5 is utilizing linear (i.e. classic linear and improved linear), as well as quadratic extrapolation techniques. Recall from Chapter 3, section 3.5 that quadratic extrapolation needs a total of $A = 3$ secondary paths and that the threshold associated with each path was assumed to be (see section 3.6): $d_1 = 0.3$, $d_2 = 0.5$

and $d_3 = 0.7$. For the case of linear extrapolation, the associated thresholds are $d_1 = 0.3$ and $d_2 = 0.7$ (See Chapter 3, section 3.4 and 3.6). The BER metering device of Fig. 5.5 has the function of determining the BER through the classic Monte-Carlo process described in Chapter 3, section 3.2 and section 3.6.2. The accuracy performance is finally determined and calculated through the process described by Chapter 3, section 3.6.1.

5.6.2 Calculation Speed of Quadratic Extrapolation

The setup configuration for this simulation is the same as illustrated by Fig. 5.5, section 5.6.1. Recall from Chapter 3, section 3.6.2 that the speed at which an extrapolation technique is able to calculate the final estimated BER value is bounded by the calculation speed of the lowest pseudo-error value $P_{p,1}(e) = P_p(e)\big|_{d=d_1}$. Therefore, the QosMU of Fig. 5.5 will only evaluate the PER at $d = d_1$, as well as the BER, through a classic Monte-Carlo process, whereupon the speed performance will be determined through the calculation processes described in section 3.6.2.

5.7 REAL-TIME ADAPTIVE CODING SIMULATIONS

This section will describe the simulation setup configuration for evaluating the real-time performance of the fully adaptive wideband MD-DS/SSMA communication system in a MU-MFCS environment. Fig. 5.6 illustrates the setup.

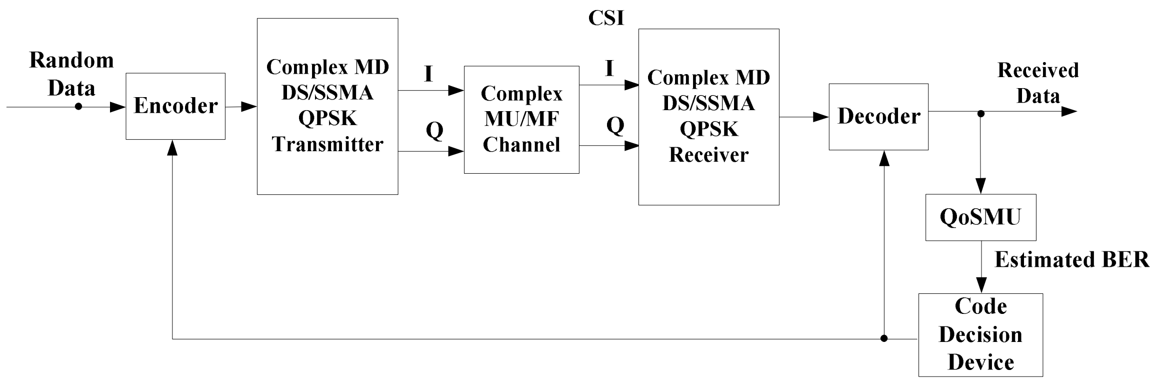


Figure 5.6 Functional Composition of the Adaptive Communication System.

The random data at the input of the system simulates the data of user-1 in the system. Depending on the code being implemented, the encoder will have one of three structures corresponding to one of the three codes of this study, i.e. binary Hamming (7,4,3) block code, (15,7,5) BCH block code and a rate 1/3 NS binary convolutional code as explained in Chapter 4, section 4.2. The setup configuration for the complex MD-DS/SSMA QPSK transmitter is the same as given by Table 5.1 of section 5.2 and the internal structure is given by Fig. 2.2 of Chapter 2, section 2.2. The setup configuration for the complex MU-MFCS is the same as given by Table 5.4 of section 5.3.3 and the internal structure is given by Fig. 2.5 of Chapter 2, section 2.3. The complex QPSK receiver is the same as explained in Chapter 2, section 2.4 and illustrated by Fig. 2.6. It is important to note that because the complex channel contains multiple users, the receiver is set to receive only the data of user-1 in the system. The receiver thus utilizes the same spreading code as the transmitter of user 1. The decoder structure is the generic VA decoder structure explained in Chapter 4, section 4.3, and contains the trellises of the three codes being implemented. The QoSMU monitors the BER performance through the estimation technique explained in Chapter 3, section 3.2 by utilizing only the quadratic extrapolation technique explained in Chapter 3, section 3.5. The code decision device decides which code to implement, based on the output of the QoSMU, and is explained in Chapter 4, section 4.4. Table 5.5 contains additional information on parameter values and simulation configurations.

Table 5.5 Additional Setup Parameters for the Real-Time Simulation of the Complete Adaptive Communication System

Parameter		**Value / Setting**
Max # of Users in System		5
Optimal BER Operating Region	Left Bound (a)	10^{-2}
	Right Bound (b)	10^{-4}
Extrapolation Technique Implemented		Quadratic
Extrapolation Thresholds Used		$d_1 = 0.3$, $d_2 = 0.5$, $d_3 = 0.7$
Constant # of Errors Counted		2000
Assumed Simulated Bit Rate		100 Kbps

During the real-time simulation process, the adaptive performance of the system will be observed by monitoring the current code mode (see Chapter 4, section 4.4), the estimated BER (log value) and the percentage throughput. The SNR per bit will be varied over a wide enough range of E_b / N_o to observe the switching between code modes. The percentage throughput $TP_{\%}$ will be determined by the current code mode, which is determined by the current coding scheme being implemented. This parameter is deterministic because of the constant input/output relation of a channel code and is calculated using the following equation:

$$TP_{\%} = 100.\left(\delta / \varsigma\right), \tag{5.2}$$

where δ represents the number of binary information input bits at the input to the encoder, and ς the number of coded binary output bits at the output of the encoder (see Chapter 4).

5.8 CONCLUDING REMARKS

This chapter was dedicated to the presentation of all setup configurations and parameters that will be performed and discussed in Chapter 6. Novel contribution made in this chapter is:

A simulation setup configuration for evaluating the accuracy and speed performance of the fully coded wideband system that applies the principles of the SOVA to the generic VA decoder for block- and convolutional codes by *Staphorst* [3], over a MU-MFCS was presented in section 5.6.

CHAPTER SIX

SIMULATION RESULTS AND DISCUSSIONS

6.1 OVERVIEW

This chapter is dedicated to the presentations and discussions of the simulation results for the experimental setups presented in Chapter 5. Firstly, the results for the spectral characteristics of the MD-DS/SSMA transmitter is presented and discussed, followed by the results and discussions for the spectral characteristics of the complex flat and multipath fading channel simulators. Next, the uncoded and coded performance of the wideband system in frequency-selective fading is presented and discussed, followed by the PER vs. threshold characteristics of the coded and uncoded wideband system in frequency selective fading. Following this section is the presentation of the results, as well as the discussions on the accuracy and speed performance simulations of the three extrapolation techniques in frequency selective fading. The last section presents the results and discussions of the most important simulations, namely the real-time adaptive coded simulations of the system in a MU-MFC environment.

All of the simulation setups presented in Chapter 5 were developed on a C++ platform and realized through the use of an object orientated programming approach. *Matlab* was also extensively used in the analysis and graphical representation of the resultant data. A 'black box' approach was followed towards the realization of the encoder and decoder structures of Chapter 4 by using existent blocks of C++ source code, obtained from *Staphorst* [3], which were incorporated into the system. All of the simulations were performed through command line driven executable applications, compiled using *Intel's ICC* compiler for *Linux* platforms, and executed on the University of Pretoria's *I-percube* 16-station *Mandrake Linux*-based HPC cluster.

6.2 TEMPORAL AND SPECTRAL CHARACTERISTICS OF THE MD-DS/SSMA TRANSMITTER

6.2.1 Simulation Results

Fig. 6.1 shows the I- and Q-channel BB temporal outputs whilst Fig. 6.2 illustrates the two-dimensional complex envelope of the BB output signal of the MD-DS/SSMA transmitter with 1 user. Finally, Fig. 6.3 illustrates the PSD of the MD-DS/SSMA transmitter output signal

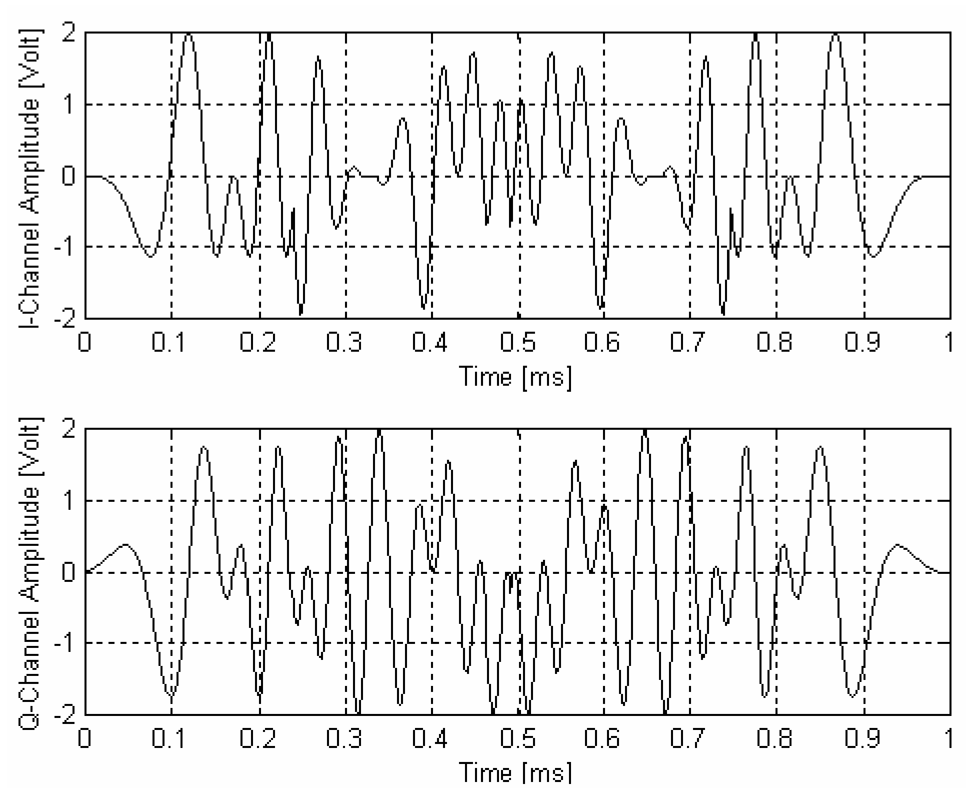


Figure 6.1. MD-DS/SSMA Transmitter I- and Q-Channel Time Signals

6.2.2 Discussion of Simulation Results

- From Fig. 6.1, the chirp-like behavior of the MD-DS/SSMA transmitter output signal employing DSB CE-LI-RU filtered GCL CSSs is clearly illustrated for one bit period.
- Fig. 6.2 illustrates that, because of its multi-dimensional structure, the two-dimensional complex envelope of the MD-DS/SSMA transmitter output signal is

not constant, even though the incorporated DSB CE-LI-RU filtered GCL CSSs have a constant two-dimensional envelope (see Appendix B, Figure B.1 and B.2).

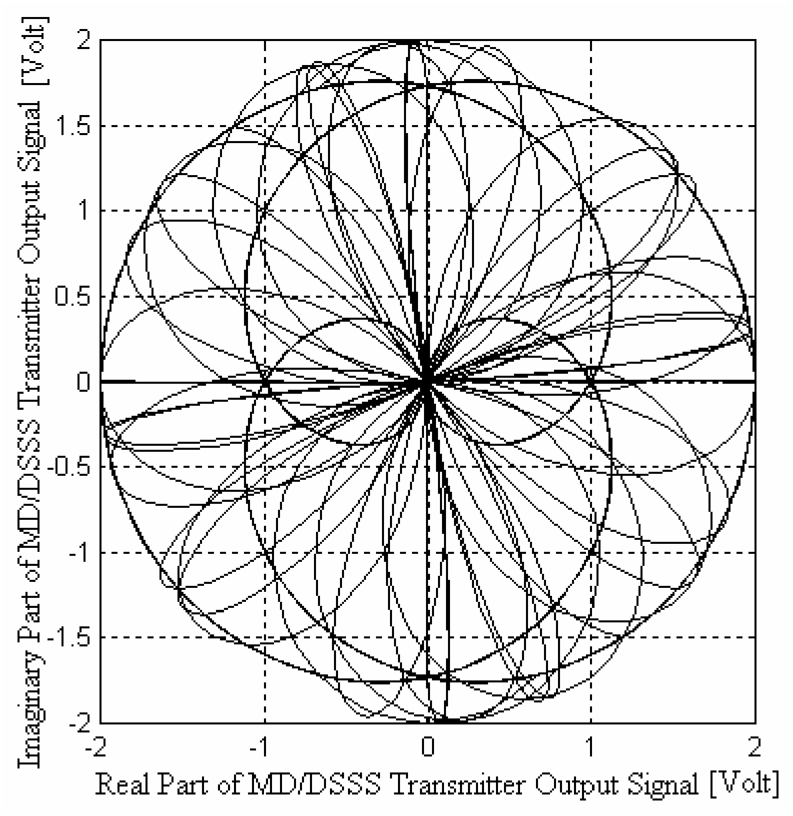


Figure 6.2. Complex Two-Dimensional Envelope of the Four-Dimensional DS/SS Transmitter BB Output Signal

- From Fig 6.3 it can be seen that DSB CE-LI-RU filtered GCL CSSs possess a relatively flat PSD within the respective bandwidth, which can also be attributed to the chirp-like behavior of the sequences. Also, the bandwidth of the signal is true to Eq. (5.1).

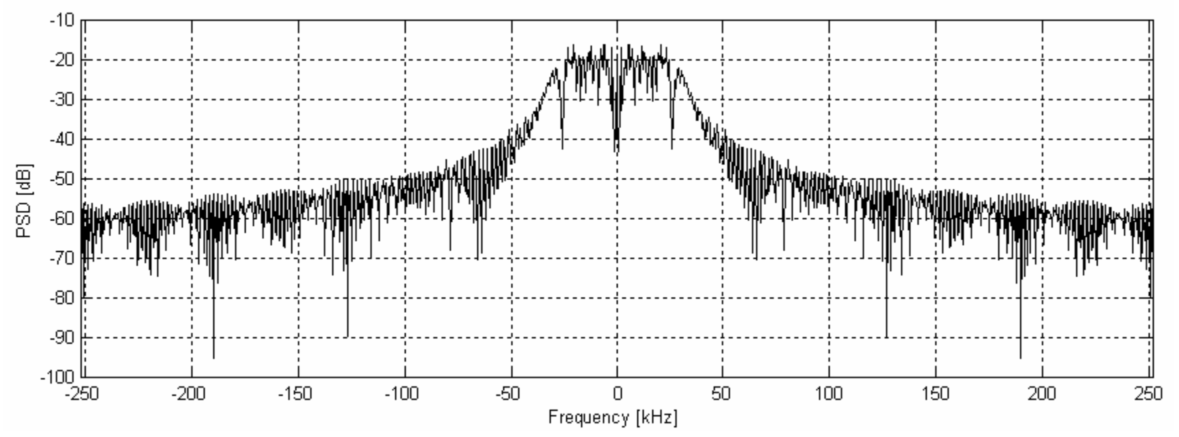


Figure 6.3. PSD of the MD-DS/SS Transmitter I- and Q-Channel Output Signals

6.3 SPECTRAL CHARACTERISTICS OF THE COMPLEX CHANNEL

6.3.1 Spectral Characteristics of the Complex Flat Fading Channel Simulator

Fig. 6.4 illustrates the PSD of the complex FFCS output signal for a complex exponential input signal $e^{j.2.\pi.f_c.n.T_s}$, and Doppler spectra of $B_{Doppler} = 10$ Hz, $B_{Doppler} = 50$ Hz and, $B_{Doppler} = 100$ Hz.

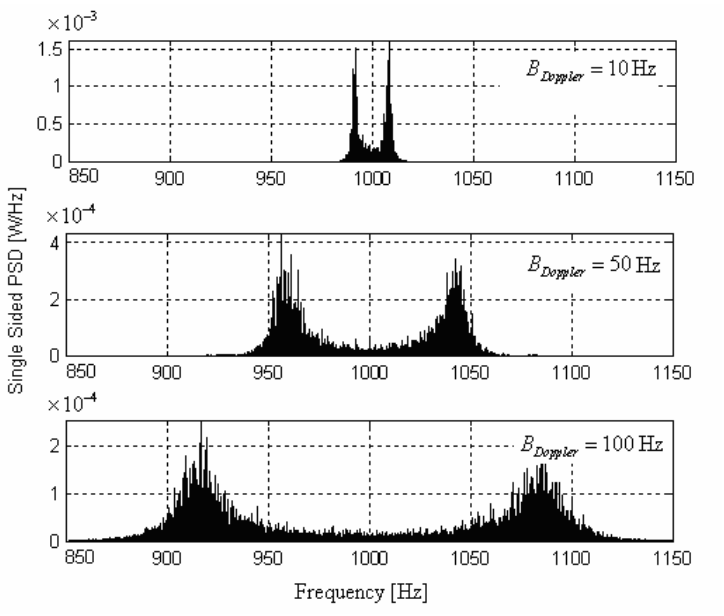


Figure 6.4. Complex FFCS Output Signal PSD for $B_{Doppler} = 10$ Hz, $B_{Doppler} = 50$ Hz, $B_{Doppler} = 100$ Hz and W_k=-100dB.

6.3.2 Spectral Characteristics of the Complex Multipath Fading Channel Simulator

Fig. 6.5 illustrates the results for the three path delays in the channel, with the respective PSD distribution of each path.

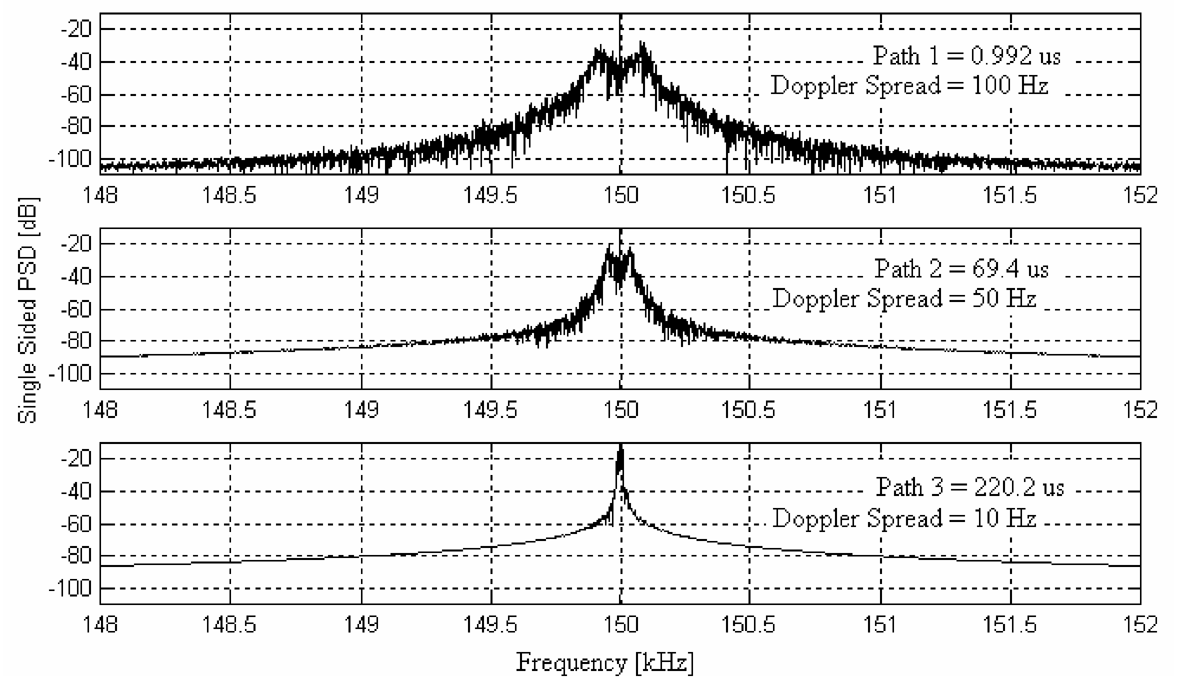


Figure 6.5. PSDs of the Paths from the Complex MFCS.

6.3.3 Discussion of Simulation Results

- From Fig. 6.4 it is clear that the FFCS reproduces realistic Doppler spectra characteristics, close to the theoretical values.
- Fig 6.5 illustrates that the correct Doppler spread for each path is reproduced, i.e. 100 Hz, 50 Hz and 10 Hz. Secondly, the PSDs of each path is unique according to each Rician factor, thus causing different LOS components at the carrier frequency of 150 kHz. Because of the small LOS component present in the third path, it experiences Rayleigh flat fading, whilst paths one and two experience mostly Rician fading.
- In conclusion, we can see that the channel simulator is able to reproduce realistic and user-definable fading environments.

6.4 UNCODED AND CODED PERFORMANCE OF WIDEBAND SYSTEM IN FREQUENCY-SELECTIVE FADING

6.4.1 Single User System

Figure 6.6. illustrates the uncoded and coded BER performance of the complete MD-DS/SS QPSK communication system in frequency selective fading for the case of 1 user in the system. Also shown on the graph is a plot of the theoretical BER performance of an uncoded 1-user system through a Rayleigh flat fading channel, which is given by [6]:

$$P_{Rayleigh}(e) = \frac{1}{2}\left(1 - \sqrt{\frac{E_b / N_o}{1 + E_b / N_o}}\right), \tag{6.1}$$

as well as the plot of the theoretical BER performance of an uncoded 1-user system through a pure Gaussian noise channel, which is given by Eq. (3.7) of Chapter 3:

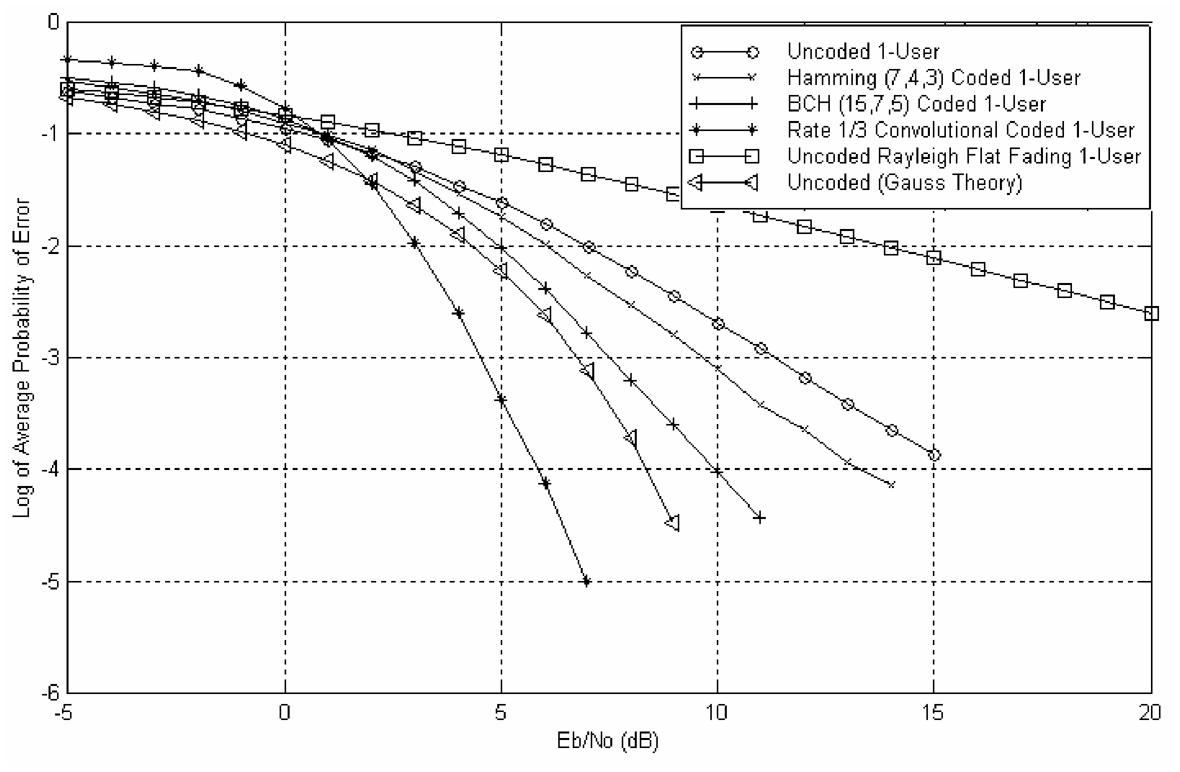


Figure 6.6. Uncoded and Coded BER Performance of the MD-DS/SS QPSK Communication System in Frequency Selective Fading for 1 User in the System.

6.4.2 Five User System

Fig. 6.7 illustrates the uncoded and coded BER performance of the complete MD-DS/SS QPSK communication system over the MU-MFCS for the case of 5 users in the system. Also shown on the graph is a plot of the theoretical BER performance of an uncoded 1-user system through a Rayleigh flat fading channel, which is given by Eq. (6.1).

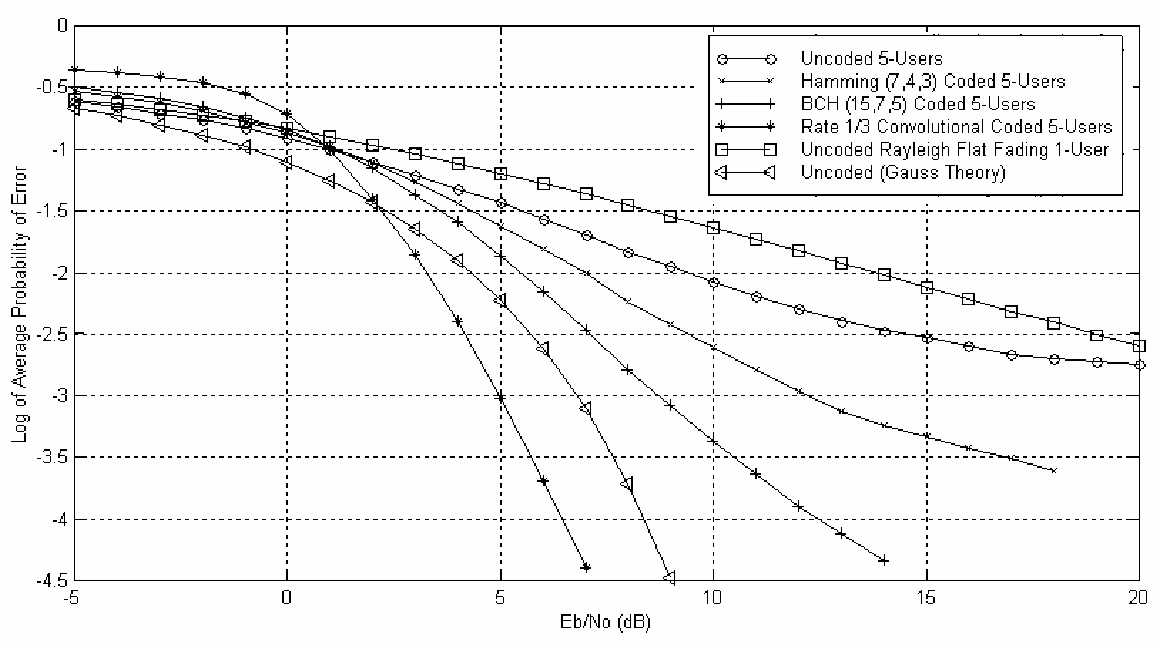


Figure 6.7. Uncoded and Coded BER Performance of the MD-DS/SS QPSK Communication System Over MU-MFCS for 5 Users.

6.4.3 Discussion of Simulation Results

- From Fig. 6.6 it is clear that implementing a higher code mode (stronger code), results in an improved BER vs. SNR performance. It therefore verifies that the binary Hamming (7,4,3) block code is the weakest of the channel codes (represented by code mode 1), whilst the rate 1/3 binary convolutional code is the strongest (represented by code mode 3). It can also be seen that the error-correcting capability of the binary BCH (15,7,5) block code (represented by code mode 2) is somewhere in between that of the aforementioned codes.
- Fig. 6.7 reveals much the same result as that of Fig. 6.6, except for a noticeable degradation in BER vs. SNR performance, because of the addition of multi-user interference caused by the additional 4 users in the system.

6.5 PER vs. THRESHOLD CHARACTERISTIC FOR THE UNCODED AND CODED WIDEBAND SYSTEM OVER MU-MFCS

6.5.1 Uncoded System

Fig. 6.8 and 6.9 respectively illustrates the simulated PER vs. threshold characteristic for the complete uncoded wideband system over the MU-MFCS for 1 and 5 users in the system, at different values of E_b / N_o.

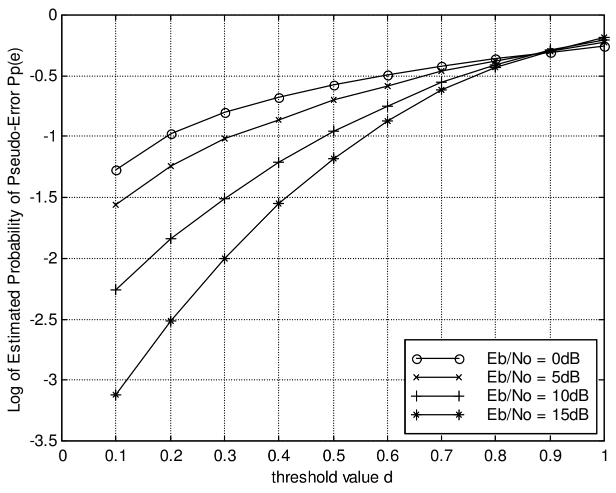


Figure 6.8. Log of Estimated (Simulated) Probability of Pseudo-Error vs. Threshold for 1-User, Uncoded System over MU-MFCS and E_b / N_o = 0, 5, 10 and 15 dB.

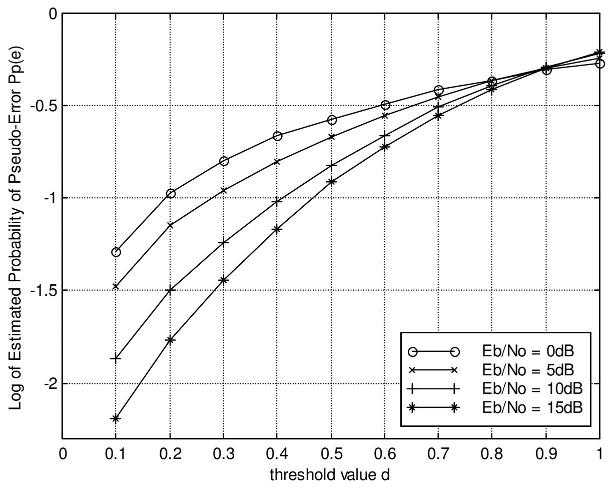


Figure 6.9. Log of Estimated (Simulated) Probability of Pseudo-Error vs. Threshold for 5-User, Uncoded System over MU-MFCS and E_b / N_o = 0, 5, 10 and 15dB.

6.5.2 Coded System

6.5.2.1 Binary Hamming (7,4,3) Block Code

Fig. 6.10 and 6.11 respectively illustrates the simulated PER vs. threshold characteristic for the binary Hamming (7,4,3) block coded wideband system for 1 and 5 users in the system, at different values of E_b / N_o.

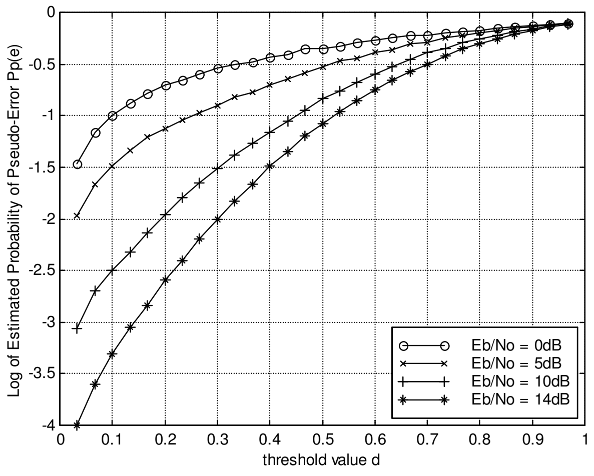


Figure 6.10. Log of Estimated (Simulated) Probability of Pseudo-Error vs. Threshold for 1-User, Binary Hamming (7,4,3) Block Coded System over MU-MFCS at E_b / N_o = 0, 5, 10 and 14 dB.

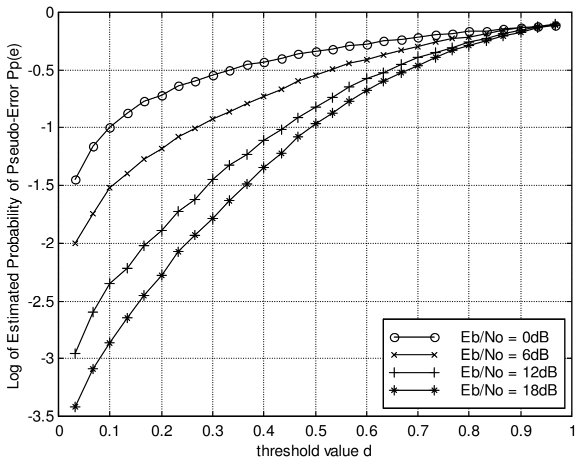


Figure 6.11. Log of Estimated (Simulated) Probability of Pseudo-Error vs. Threshold for 5-User, Binary Hamming (7,4,3) Block Coded System over MU-MFCS at E_b / N_o = 0, 6, 12 and 18 dB.

6.5.2.2 Binary BCH (15,7,5) Block code

Fig. 6.12 and 6.13 respectively illustrates the PER vs. threshold characteristic for the binary BCH (15,7,5) block coded wideband system for 1 and 5 users in the system, at different E_b / N_o.

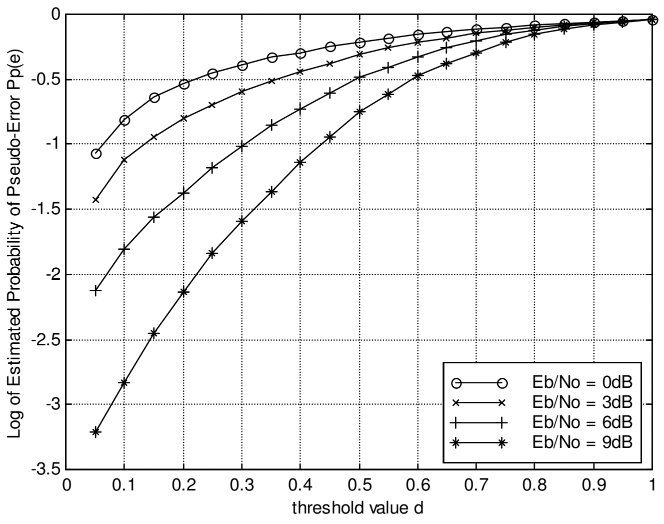


Figure 6.12. Log of Estimated (Simulated) Probability of Pseudo-Error vs. Threshold for 1-User, Binary BCH (15,7,5) Block Coded System over MU-MFCS at $E_b / N_o = 0$, 3, 6 and 9 dB.

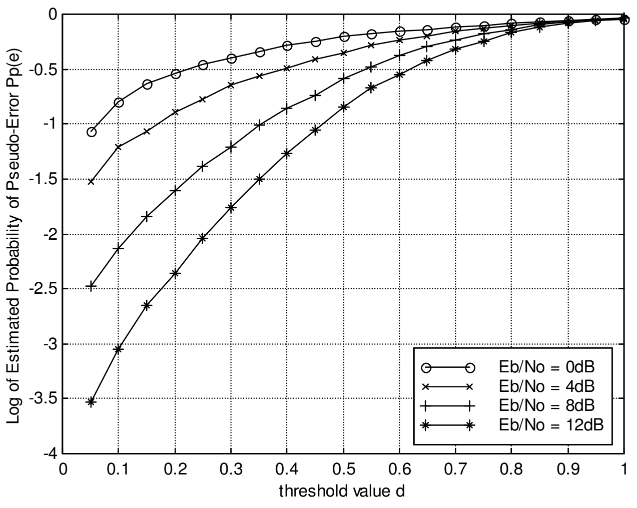


Figure 6.13. Log of Estimated (Simulated) Probability of Pseudo-Error vs. Threshold for 5-Users, Binary BCH (15,7,5) Block Coded System over MU-MFCS at $E_b / N_o = 0$, 4, 8 and 12 dB.

6.5.2.3 Rate 1/3 Binary Convolutional Code

Fig. 6.14 and 6.15 repectively illustrates the PER vs. threshold characteristic for the rate 1/3 binary convolutional coded wideband system for 1 and 5 users in the system, and for different E_b / N_o.

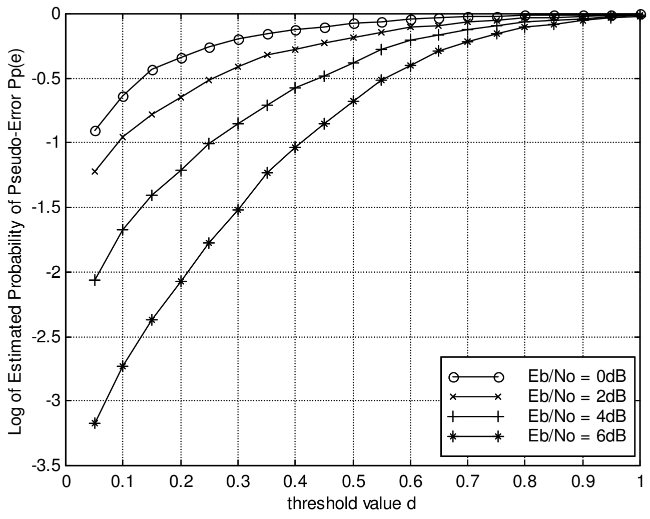


Figure 6.14. Log of Estimated (Simulated) Probability of Pseudo-Error vs. Threshold for 1-User, Rate 1/3 Binary Convolutional Coded System over MU-MFCS at E_b / N_o = 0, 2, 4 and 6 dB.

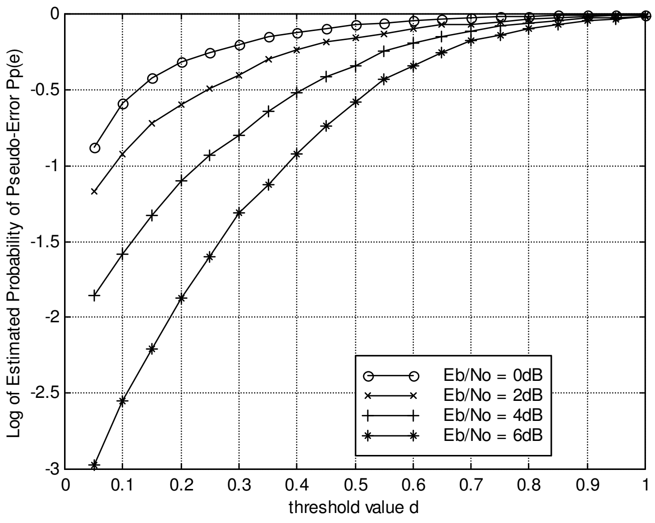


Figure 6.15. Log of Estimated (Simulated) Probability of Pseudo-Error vs. Threshold for 5-Users, Rate 1/3 Binary Convolutional Coded System over MU-MFCS at E_b / N_o = 0, 2, 4 and 6 dB.

6.5.3 Discussion of Simulation Results

- Figs. 6.8 and 6.9 reveal an approximately linear relationship between the log of $\hat{P}_p(e)$ and the threshold value d for d > 0.2 for both 1 and 5-user systems, suggesting that the linear extrapolation technique would have fair performance in the estimation of BER for an uncoded system. There is also a noticeable common crossover point at $\log\left(\hat{P}_p(e)\big|_{d=0.9}\right) = -0.3$, where all the PER vs. threshold curves seem to pass through.
- Figs. 6.10 through 6.15 also reveal an approximately linear relationship between the log of $\hat{P}_p(e)$ and the threshold value d for d > 0.2 for both 1 and 5-user systems, as well as for all the relevant codes of this study, suggesting that the employment of the SOVA [39] in the system is effective for coded systems and that BER estimation will have satisfying results. There is, however a noticeable increase in the curvature the PER vs. threshold characteristic for all of the codes when compared to the uncoded system results. This could prove cumbersome to the linear extrapolation techniques, but conversely, be supportive to that of quadratic extrapolation. Also, there does not seem to be a noticeable common crossover point present in any of the PER vs. threshold curves for the coded cases.

6.6 CALCULATION ACCURACY AND SPEED OF EXTRAPOLATION FOR UNCODED AND CODED WIDEBAND COMMUNICATION SYSTEM OVER MU-MFCS

6.6.1 Calculation Accuracy

6.6.1.1 Uncoded System

Fig. 6.16 firstly illustrates the estimated (simulated) log probability of error $\log\left(\hat{P}_e(e)\right)$ vs. E_b / N_o (dB) obtained via the classic Monte-Carlo process for $-5dB < E_b / N_o < 15dB$ in a 1 system user. Superimposed onto this plot are the graphs of the estimated log probability of error vs. E_b / N_o (dB) resulting from the estimation performance simulations of the three extrapolation techniques presented in this study (see Chapter 3 section 3.6.1), for the same range of E_b / N_o values. Fig. 6.16 shows the same results for a 5-user. system

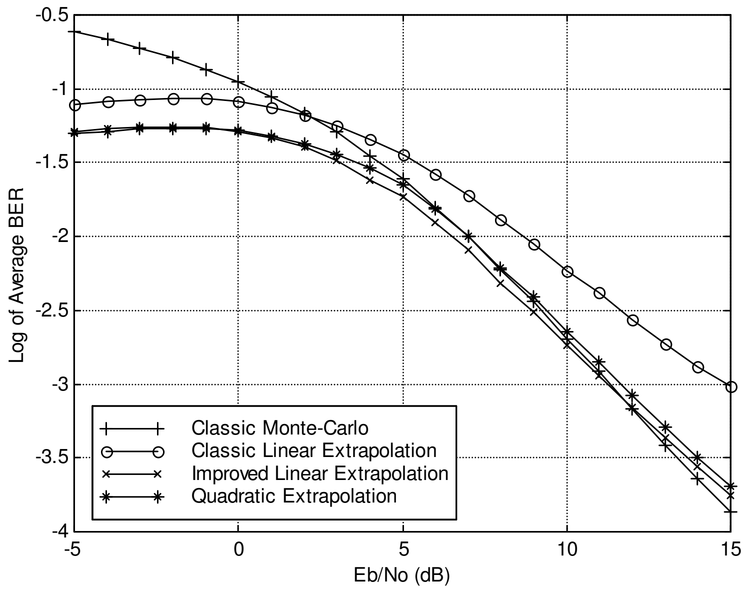


Figure 6.16. Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to the Classic Monte-Carlo Probability of Error vs. E_b / N_o for an Uncoded, 1-User System in MU-MFCS.

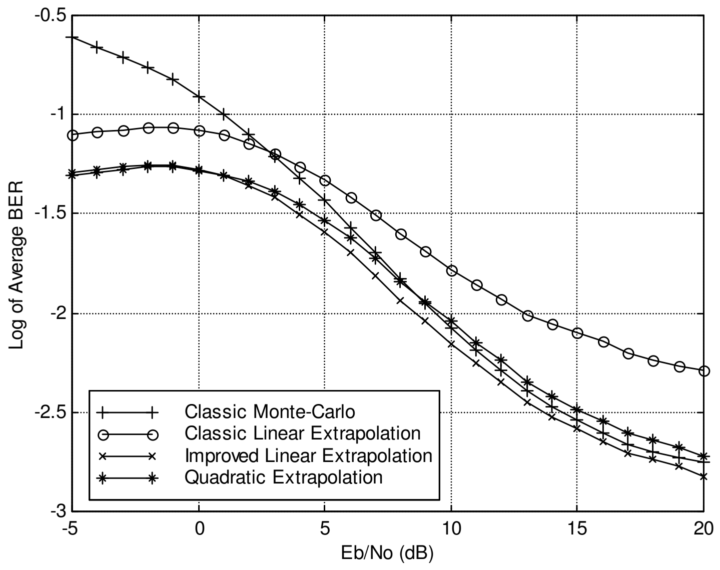


Figure 6.17. Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to the Classic Monte-Carlo Probability of Error vs. E_b / N_o for an Uncoded, 5-User System in MU-MFCS.

As was seen in Chapter 3, section 3.6.1, in order to properly investigate the calculation accuracy performance of these three techniques we will have to take a closer look at the degree of accuracy by calculating the difference between the technique's estimated BER value and the Classic Monte-Carlo estimated BER value, for each value of E_b / N_o. We again denote this difference value as the *error performance* $\varepsilon_p^{Technique}$ of the technique, and redefine Eq. (3.32) as:

$$\varepsilon_p^{Technique} = \log\left(\hat{P}_e^{Technique}(e)\right) - \log\left(P_e^{Monte-Carlo}(e)\right), \qquad (6.2)$$

where the superscript '*Technique*' points to the specific extrapolation technique investigated, and the superscript '*Monte-Carlo*' points to the classic Monte-Carlo results. Note that this error value is also a logarithm because of the subtraction of two logarithmic values. Fig. 6.18 illustrates a superimposed plot of the error performance of each of the estimation techniques for a single-user, uncoded system in MU-MFCS, and for the range $-5dB < E_b / N_o < 15dB$, as calculated by Eq.(6.2). Fig. 6.19 illustrates the same result for a 5-user system.

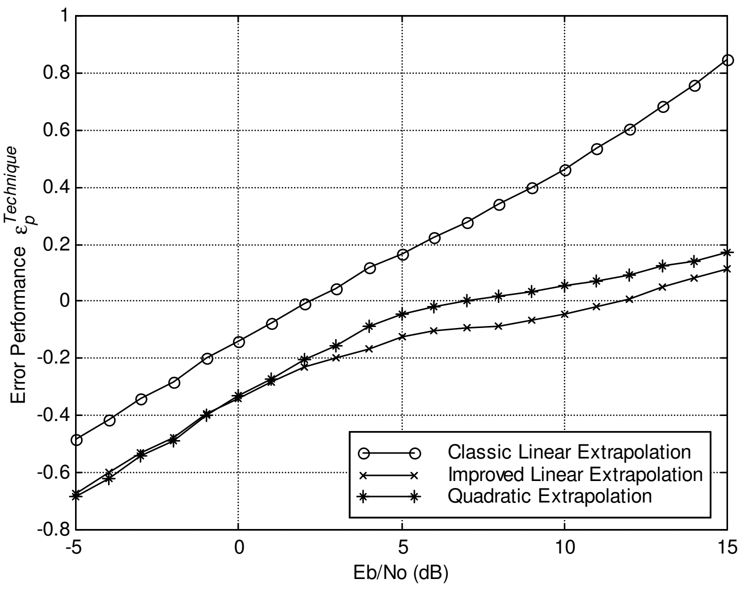


Figure 6.18. Error Performance vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a 1-User, Uncoded System in MU-MFCS.

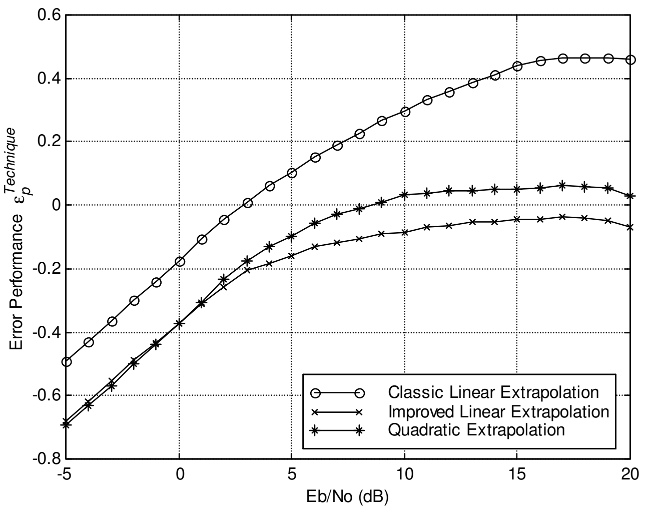


Figure 6.19. Error Performance vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a 5-User, Uncoded System in MU-MFCS.

6.6.1.2 Binary Hamming (7,4,3) Coded System

Fig. 6.20 illustrates the estimation performances of the extrapolation techniques for a binary Hamming (7,4,3) coded, 1-user system in MU-MFCS. Fig. 6.21 shows the same results for a 5-user system.

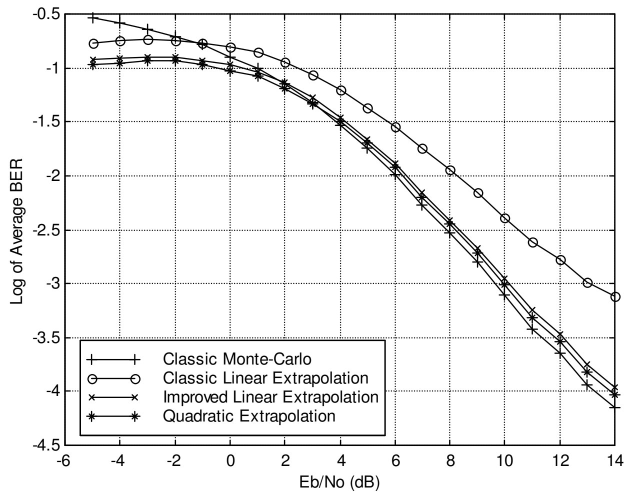


Figure 6.20. Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to the Classic Monte-Carlo Probability of Error vs. E_b / N_o for a binary Hamming (7,4,3) coded, 1-User System in MU-MFCS.

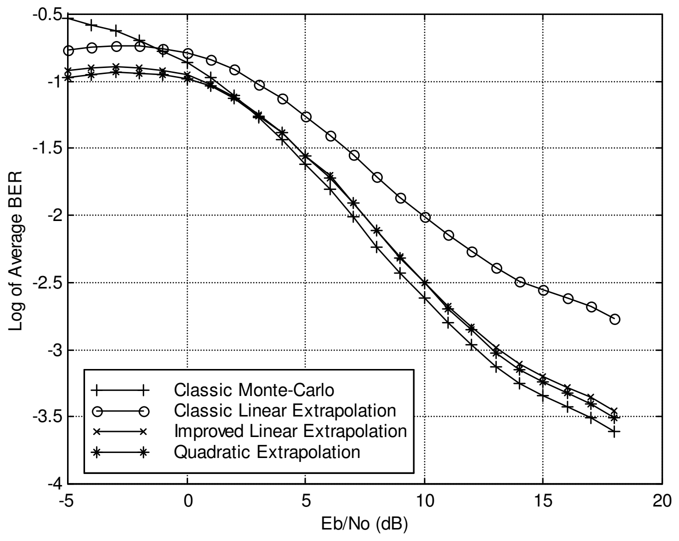


Figure 6.21. Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to the Classic Monte-Carlo Probability of Error vs. E_b / N_o for a binary Hamming (7,4,3) coded, 5-User System in MU-MFCS.

Fig. 6.22 illustrates the error-performances of the extrapolation techniques, for a binary Hamming (7,4,3) coded, 1-user system over MU-MFCS. Fig. 6.23 shows the same results for a 5-user system.

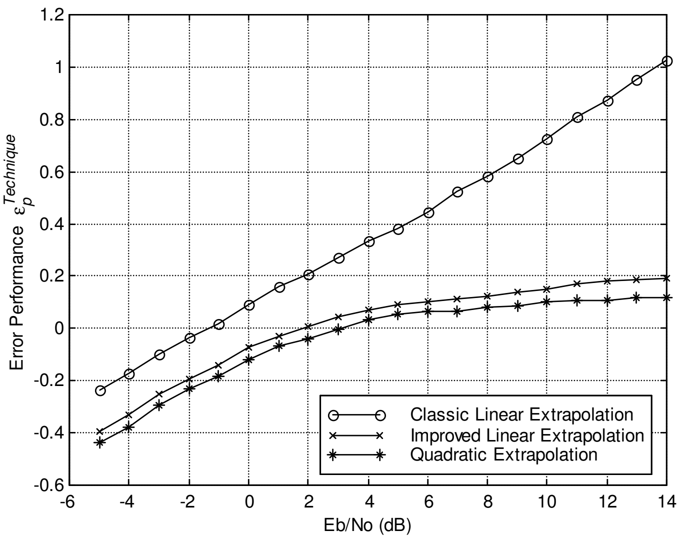


Figure 6.22. Error Performance vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a 1-User, Binary Hamming (7,4,3) coded System in MU-MFCS.

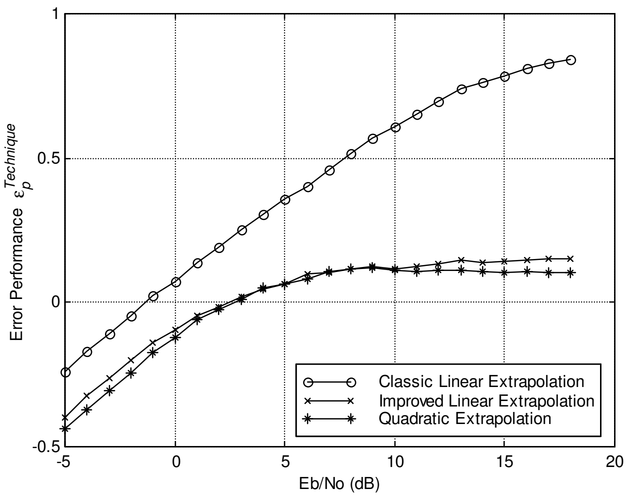


Figure 6.23. Error Performance vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a 5-User, Binary Hamming (7,4,3) coded System in MU-MFCS.

6.6.1.3 Binary BCH (15,7,5) Coded System

Fig. 6.24 illustrates the estimation performances of the extrapolation techniques for a binary BCH (15,7,5) coded, 1-user system in MU-MFCS. Fig. 6.25 shows the same results for a 5-user system.

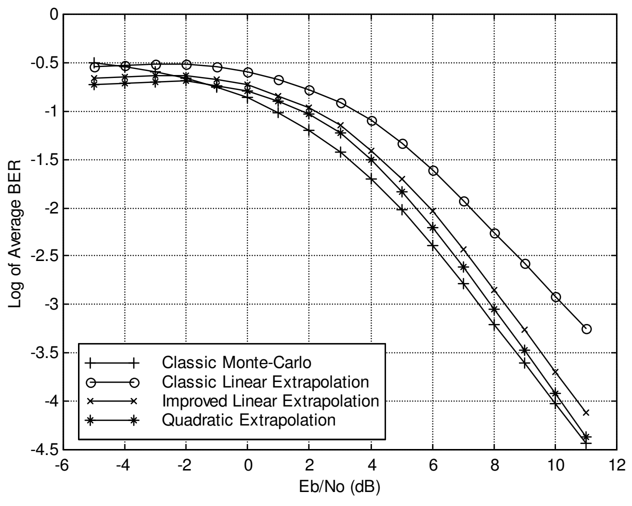


Figure 6.24. Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to the Classic Monte-Carlo Probability of Error vs. E_b / N_o for a Binary BCH (15,7,5) Coded, 1-User System in MU-MFCS.

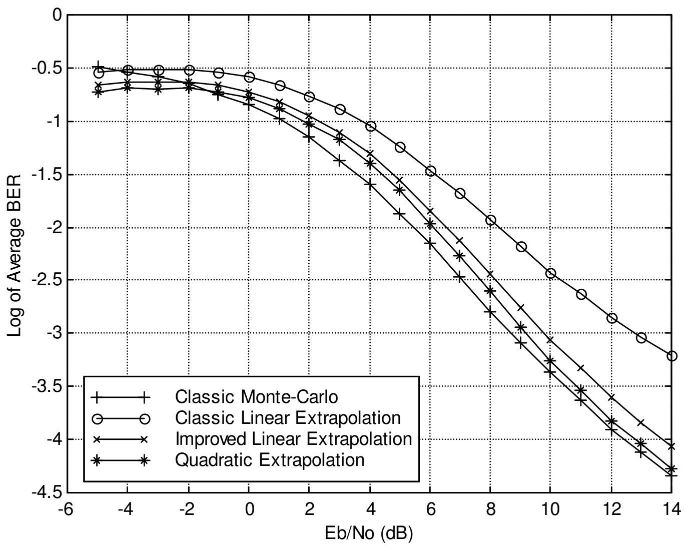


Figure 6.25. Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to the Classic Monte-Carlo Probability of Error vs. E_b / N_o for a Binary BCH (15,7,5) Coded, 5-User System in MU-MFCS.

Fig. 6.26 illustrates the error-performances of the extrapolation techniques, for a binary BCH (15,7,5) coded, 1-user system in MU-MFCS. Fig. 6.27 shows the same results for a 5-user system.

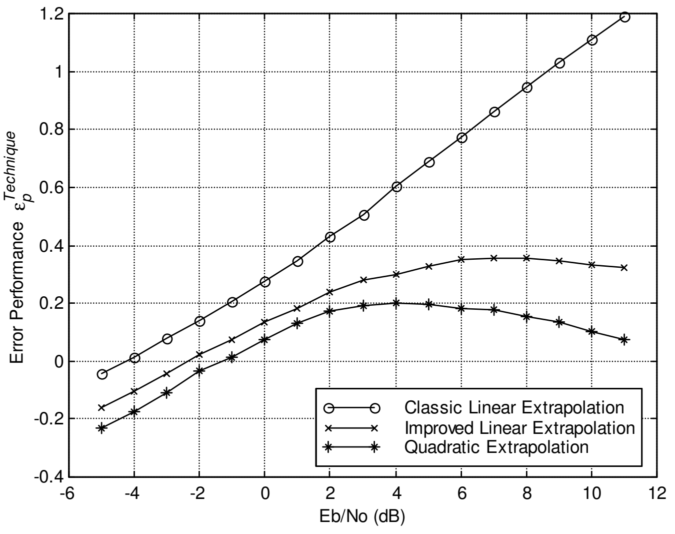


Figure 6.26. Error Performance vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a 1-User, Binary BCH (15,7,5) Coded System in MU-MFCS.

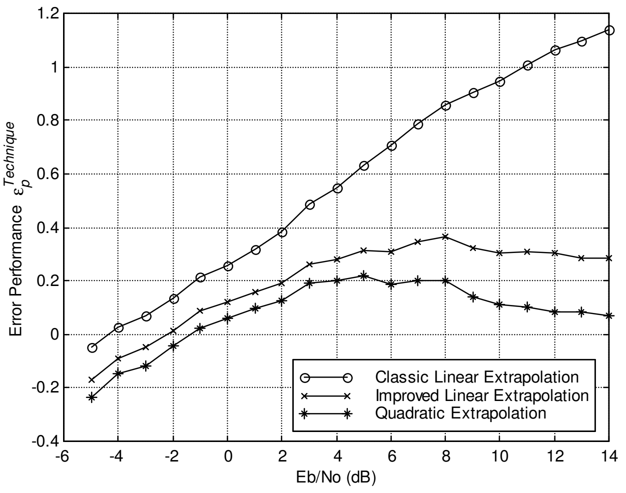


Figure 6.27. Error Performance vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a 5-User, Binary BCH (15,7,5) Coded System in MU-MFCS.

6.6.1.4 Rate 1/3 Binary Convolutional Coded System

Fig. 6.28 illustrates the estimation performances of the extrapolation techniques for a rate 1/3 binary convolutional coded, 1-user system over MU-MFCS. Fig. 6.29 shows the same results for a 5-user system.

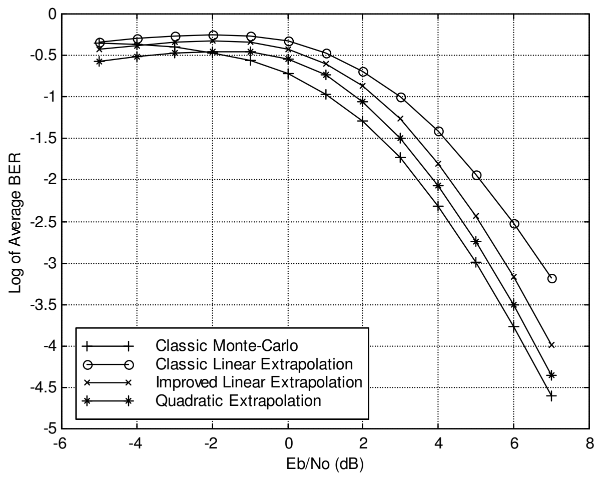


Figure 6.28. Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to the Classic Monte-Carlo Probability of Error vs. E_b / N_o for a rate 1/3 binary Convolutional Coded, 1-User System in MU-MFCS.

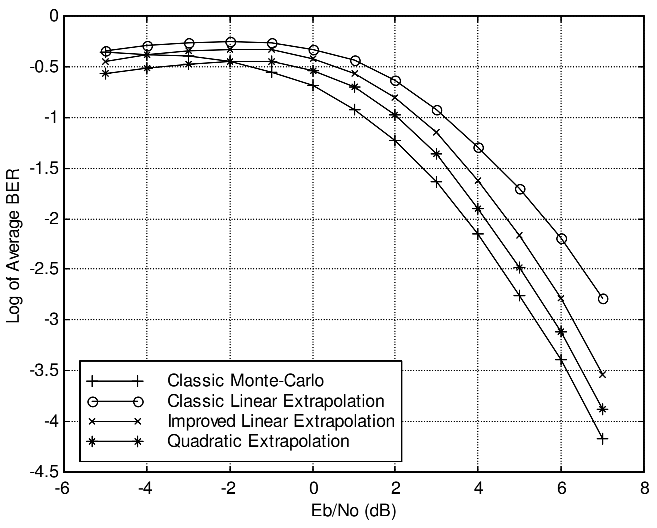


Figure 6.29. Estimation Performance of Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation Compared to the Classic Monte-Carlo Probability of Error vs. E_b / N_o for a rate 1/3 binary Convolutional Coded, 5-User System in MU-MFCS.

Fig. 6.30 illustrates the error-performances of the extrapolation techniques, for a rate 1/3 binary Convolutional coded, 1-user system in MU-MFCS. Fig. 6.31 shows the same results for a 5-user system.

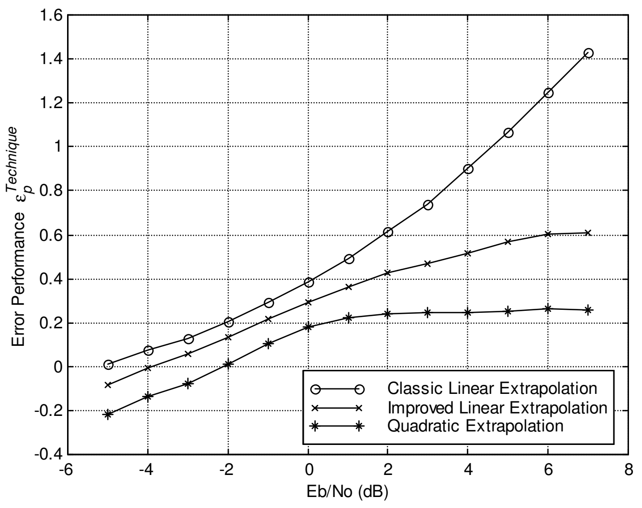


Figure 6.30. Error Performance vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a 1-User, Rate 1/3 Binary Convolutional Coded System in MU-MFCS.

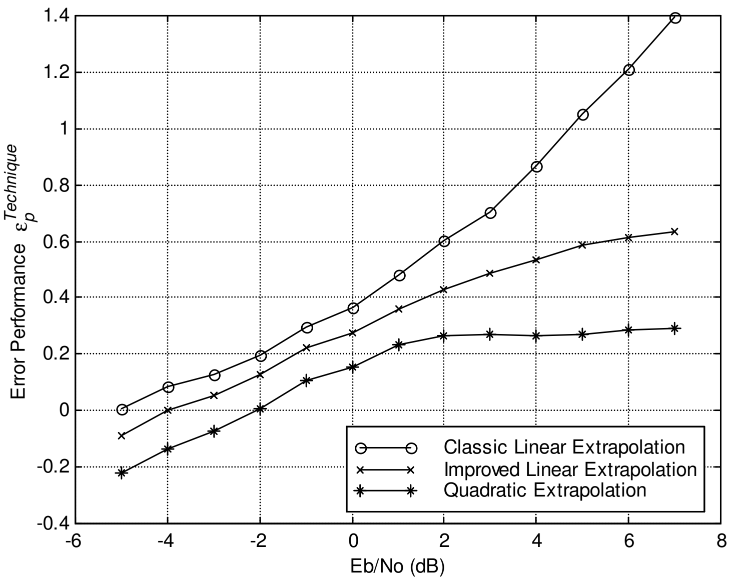


Figure 6.31. Error Performance vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a 5-User, Rate 1/3 Binary Convolutional coded System in MU-MFCS.

6.6.2 Calculation Speed

6.6.2.1 Uncoded System

Chapter 5, section 5.6.2 explains the experimental setup used to obtain the results presented in this section. It also refers to the explanation of the speed performance principles on Chapter 3, section 3.6.2. For clarity, we recall from Chapter 3, that in order to evaluate the speed performance s_p of an extrapolation technique, we need to determine 'how much faster' it performs compared to the classic Monte-Carlo procedure by using Eq. (3.34) of Chapter 3, section 3.6.2, and assuming equal errors counted for both processes. Also recall that all three extrapolation techniques will have the same speed performance because they share the same lowest pseudo-error value (see Chapter 3, section 3.6.2). Fig. 6.32 illustrates the speed performance as depicted by the abovementioned standards for both uncoded, 1 and 5-user systems in MU-MFCS.

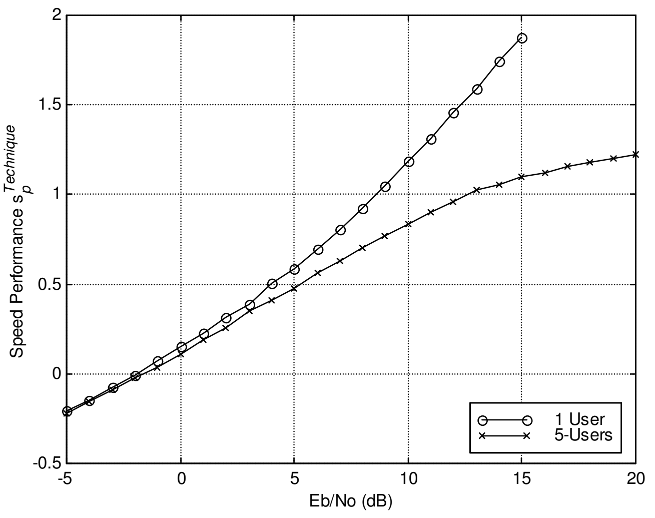


Figure 6.32. Log of Calculation Speed Performance s_p vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for an Uncoded, 1- and 5-User System in MU-MFCS.

6.6.2.2 Binary Hamming (7,4,3) Coded System

Fig. 6.33 illustrates the speed performance for a binary Hamming (7,4,3) coded, 1 and 5-user system in MU-MFCS.

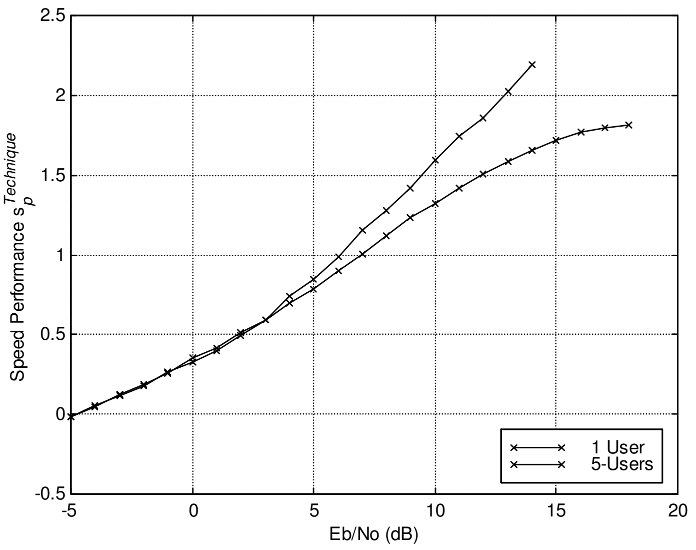


Figure 6.33. Log of Calculation Speed Performance s_p vs. E_b / N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a Binary Hamming (7,4,3) Coded, 1 and 5-User System in MU-MFCS.

6.6.2.3 Binary BCH (15,7,5) Coded System

Fig. 6.34 illustrates the speed performance for a binary BCH (15,7,5) coded, 1 and 5-user system in MU-MFCS.

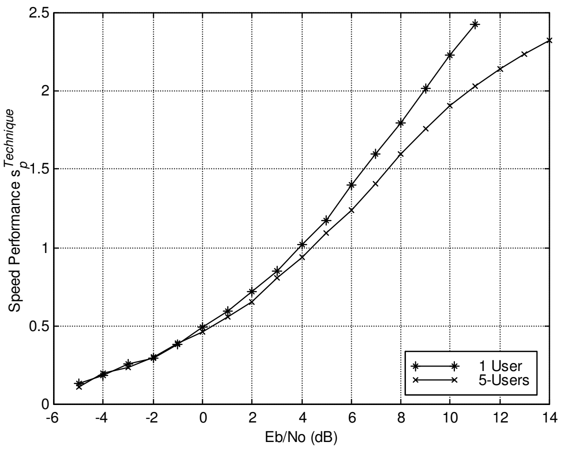


Figure 6.34. Log of Calculation Speed Performance s_p vs. E_b/N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a Binary BCH (15,7,5) Coded, 1 and 5-User System in MU-MFCS.

6.6.2.4 Rate 1/3 Binary Convolutional Coded System

Fig. 6.35 illustrates the speed performance for a rate 1/3 binary Convolutional coded, 1 and 5-user system over MU-MFCS.

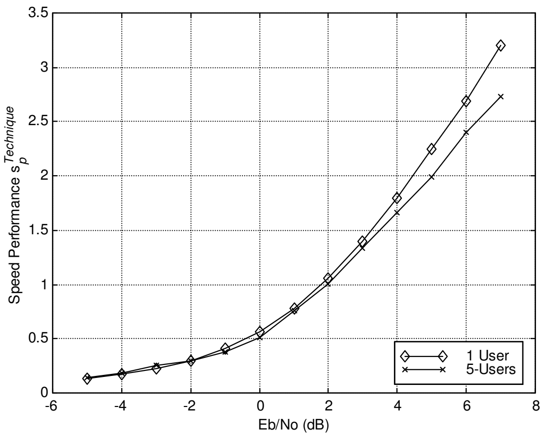


Figure 6.35. Log of Calculation Speed Performance s_p vs. E_b/N_o for Classic Linear Extrapolation, Improved Linear Extrapolation and Quadratic Extrapolation for a Rate 1/3 Binary Convolutional Coded, 1 and 5-User System in MU-MFCS.

6.6.3 Discussion of Simulation Results

- Figs. 6.16 and 6.17 reveals that improved linear and quadratic extrapolation have adequate estimation performance, whereas classic linear extrapolation have large deviation (> 10) within the chosen range for an uncoded system over MU-MFCS.
- Figs. 6.18 and 6.19 reveals that improved linear- and quadratic extrapolation seems to have an error performance within approximately $10^{0.2}$ of the classic Monte-Carlo results for $E_b / N_o > 2dB$, whereas classic linear extrapolation has rather large deviations (> 0.5) within the chosen range of E_b / N_o for an uncoded system over MU-MFCS.
- Figs. 6.20 through 6.31 illustrates that quadratic extrapolation has superior performance to that of classic and improved linear extrapolation for all coded cases in MU-MFCS for the chosen range of E_b / N_o. All error performance graphs reveal that quadratic extrapolation has an error within approximately $10^{0.2}$ for $E_b / N_o > 0dB$, which verifies the superior performance of this technique over the other two techniques for the chosen range of E_b / N_o.
- Fig. 6.32 illustrates that all extrapolation techniques yield an increase in calculation speed with an increase in E_b / N_o. It also reveals that for approximately $E_b / N_o < -2dB$, all extrapolation techniques are slower ($s_p < 0$) than the classic Monte-Carlo process. Fig. 6.32 also reveals that for a 1-user, uncoded system at $E_b / N_o = 15dB$, all extrapolation techniques are $s_p \approx 10^{1.8} \approx 63.1$ times faster than the classic Monte-Carlo process, and for a 5-user system at $E_b / N_o = 20dB$, all extrapolation techniques are $s_p \approx 10^{1.2} \approx 15.8$ times faster than the classic Monte-Carlo process.
- Figs. 6.33 through 6.35 reveal similar results as for the uncoded case, with calculation speed performances reaching as high as $s_p \approx 10^{3.2} \approx 1585$. Also, these figures reveal that in the chosen ranges of E_b / N_o, the extrapolation techniques are always faster than the classic Monte-Carlo process (i.e. $s_p > 0$) for all coding schemes of this study. Thus, the equal speed performance point on the curve ($s_p = 0$) is situated further to the left of the graph (i.e. for $E_b / N_o < -5dB$).

6.7 REAL-TIME ADAPTIVE CODING SIMULATION RESULTS

6.7.1 Single User System

Fig. 6.36 illustrates the real-time adaptive channel coding performance of the fully functional wideband communication system for 1 user in the system that corresponds with the simulation setup configuration explained in Chapter 5, section 5.7. The percentage throughput graphs were generated using Eq. 5.2 from Chapter 5, section 5.7. Also, the code mode graphs represent the state of the current code mode, which is linked to one of four possible states (see Chapter 4, section 4.4). We observe that the SNR per bit was varied over the range $-5dB < E_b / N_o < 17dB$, i.e. for a progressively better channel, over a time span of approximately 70 seconds. This figure reveals the state of the code mode, the percentage throughput and the estimated BER (log value) performance over the chosen SNR range and time span. Each of the points on the graph corresponds to the time that the QoSMU has successfully estimated a BER value.

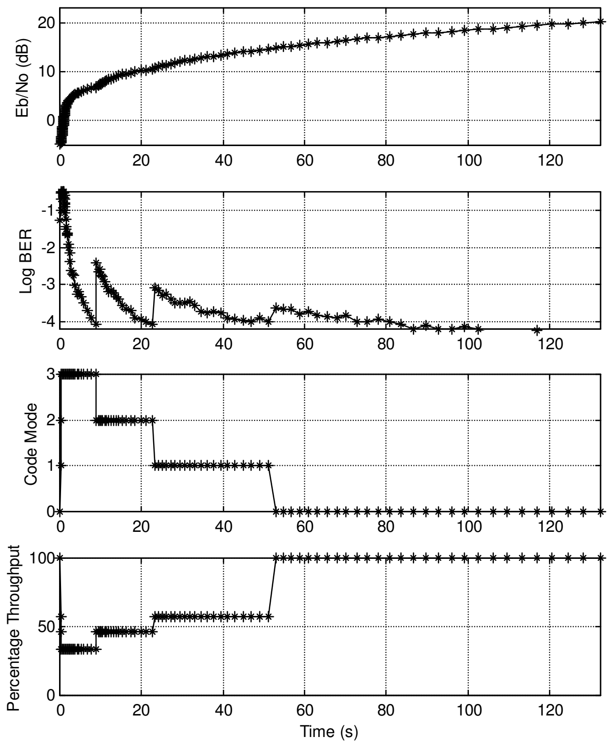


Figure 6.36. Adaptive Coding Performance of the Fully Functional Wideband Communication System for 1 User in the System for a Progressively Better Channel ($-5dB < E_b / N_o < 17dB$).

Fig. 6.37 illustrates similar adaptive channel coding performance for 1 user in the system. We observe that in this instance, the SNR per bit was varied over the range $9dB > E_b / N_o > -5dB$, i.e. for a progressively worse channel, over a time of approximately 10 seconds. This figure reveals the state of the code mode, the percentage throughput and the estimated BER (log value) performance over the chosen SNR range and time span. Each point on the graph corresponds with the time the QoSMU has successfully estimated a BER value.

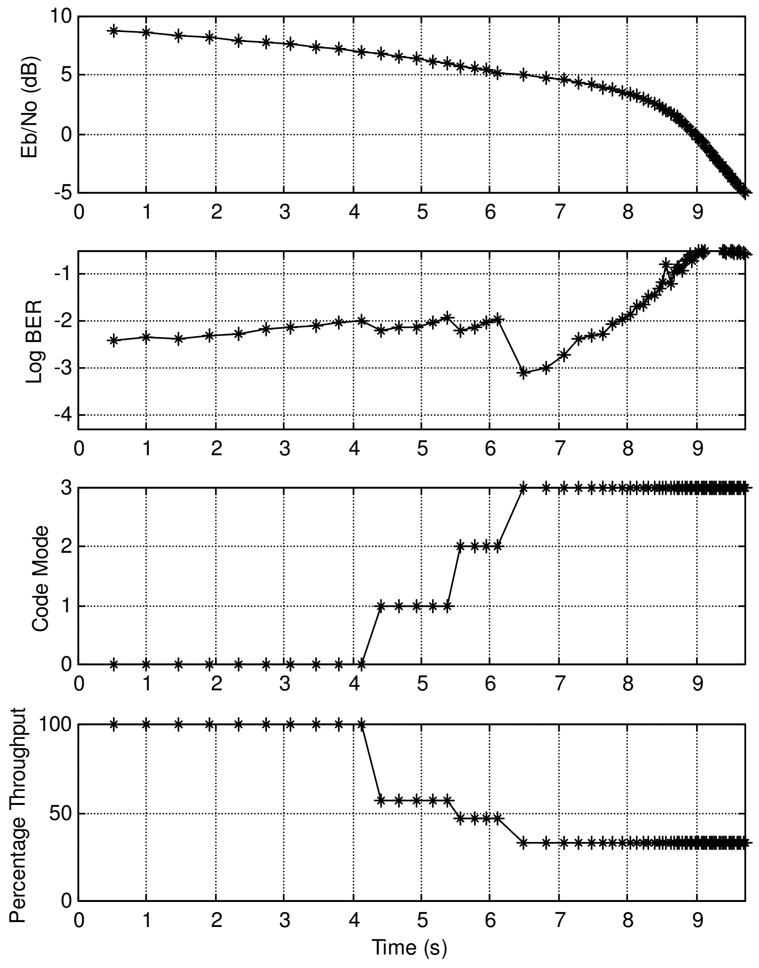


Figure 6.37. Adaptive Coding Performance of the Fully Functional Wideband Communication System for 1 User for a Progressively Worse Channel ($9dB > E_b / N_o > -5dB$)

6.7.2 Five User System

Fig. 6.38 illustrates the real-time adaptive channel coding performance of the fully functional wideband communication system for 5 users in the system that corresponds with the simulation setup configuration explained in Chapter 5, section 5.7. We observe that the SNR was varied over the range $-5dB < E_b / N_o < 30dB$, i.e. for a progressively better channel, over a time of approximately 110 seconds. This figure reveals the state of the code mode, the % throughput and the estimated BER (log value) performance over the chosen SNR range and time span. Each of the points on the graph corresponds to the time that the QoSMU has successfully estimated a BER value.

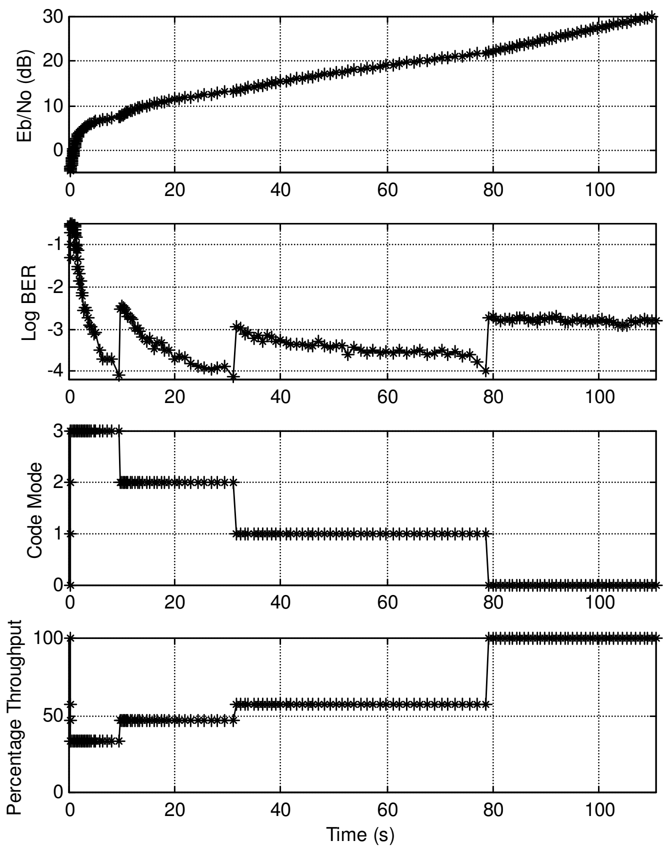


Figure 6.38. Adaptive Coding Performance of the Fully Functional Wideband Communication System for 5 Users for a Progressively Better Channel ($-5dB < E_b / N_o < 30dB$)

Fig. 6.39 illustrates similar adaptive channel coding performance for 5 users in the system. We observe that in this instance, the SNR per bit was varied over the range

$12dB > E_b / N_o > -5dB$, i.e. for a progressively worse channel, over a time of approximately 14 seconds. This figure reveals the state of the code mode, the percentage throughput and the estimated BER (log value) performance over the chosen SNR per bit range and time span. Each of the points on the graph corresponds to the time that the QoSMU has successfully estimated a BER value.

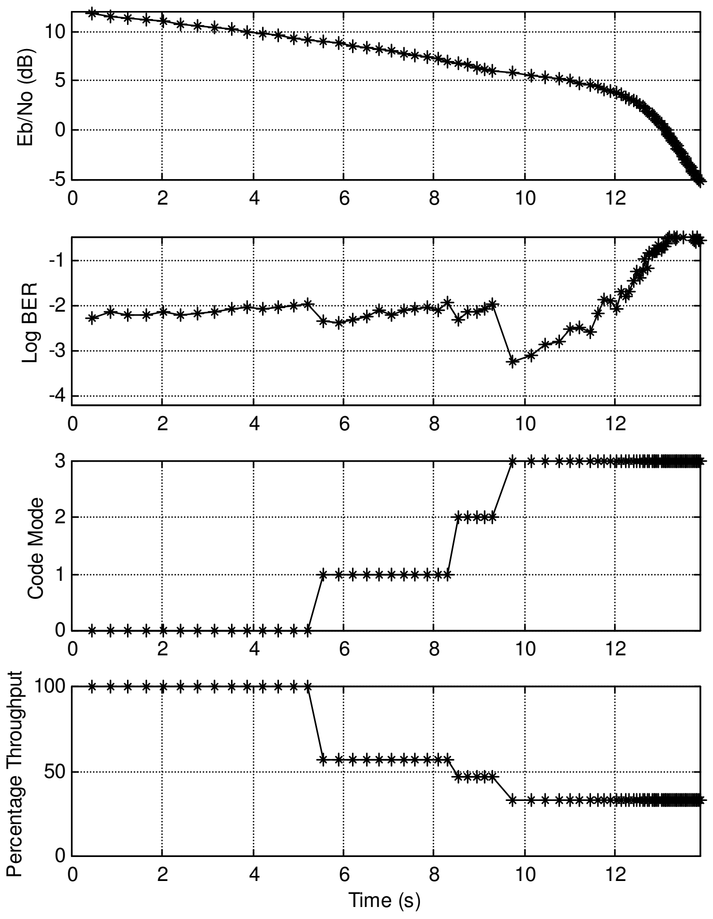


Figure 6.39. Adaptive Coding Performance of the Fully Functional Wideband Communication System for 5 Users for a Progressively Worse Channel ($12dB > E_b / N_o > -5dB$)

6.7.3 Discussion of Simulation Results

- Figs. 6.36 and 6.37 reveals that when the SNR was varied slowly, the complete 1-user system was successful in firstly sensing the needed adaptation, and further successful in performing the switching between code modes in order to keep the BER in the predetermined range $10^{-4.0} < BER < 10^{-2.0}$ (see Chapter 4, section 4.4).

- Figs. 6.38 and 6.39 reveals that when the channel condition (SNR) was varied slowly, the complete 5-user system was successful in firstly sensing the needed adaptation, as well as performing the switching between code modes in order to keep the BER in the predetermined range $10^{-4.0} < BER < 10^{-2.0}$ (see Chapter 4, section 4.4).
- In Figs 6.36 through 6.39 we observe that the separation between points on the graphs increase as the SNR per bit increases (improves). This can be explained by the fact that, as channel conditions improve, the BER decreases and the QoSMU takes a longer time in estimating the next BER value. The points are observed to be as far apart as approximately 1-second (for $\text{BER} \approx 10^{-4}$). This reveals that for a system operating at a bit rate of 100 kbps (see Chapter 5, section 5.7), the QoSMU is able to determine a new BER value every 1 second for $\text{BER} \approx 10^{-4}$.

6.8 CONCLUDING REMARKS

This Chapter was dedicated to the presentation of the simulation results for all of the simulation setup configurations of Chapter 5. Important simulation results and novel contributions made by this chapter are:

1. Important uncoded and coded BER performances for 1 as well as 5 user systems over a uniquely configured MU-MFCS were presented in section 6.4
2. Simulated PER vs. threshold characteristics for a coded and uncoded 1- and 5 user systems was presented in section 6.5. More specifically, the principles of *Gooding* [12, 13] have never before been applied to a proper MU-MFCS, nor has it ever been applied to a system that employs *Hagenauer's* SOVA [39]. Also, these simulations have never before been performed, nor have these results ever been presented in previous literature.
3. Important results that reveal the trade-offs between BER estimation accuracy and speed for the three extrapolation techniques of this study (see Chapter 3) over frequency-selective fading conditions (i.e. over MU-MFCS) were presented in section 6.6.
4. The last section of this chapter presented the most important simulation results of this study, namely that of the real-time adaptation performance of the fully

functional adaptively channel coded wideband communication system over a uniquely defined MU-MFCS and varying channel conditions.

CHAPTER SEVEN

CONCLUSIONS & FUTURE RESEARCH

7.1 CONCLUSIONS

This study investigated the performance of an adaptive channel coded communication system employing a novel, sophisticated MD DS/SSMA technique as well as a real-time BER estimation scheme by *Gooding* [12, 13]. The performance of this communication system was tested over a sophisticated, fully user-definable, software-based complex MU-MFCS by *Staphorst* [3]. Novel investigations in this study include the proposal of a unique multi-dimensional (MD) direct-sequence spread spectrum multiple access (DS/SSMA) transmitter and receiver structure, as well as the application of the well-known SOVA structure by *Hagenauer* [39] to the real-time BER estimation technique by *Gooding* [12, 13]. The most important conclusions that can be drawn from the results presented in this study are:

1. From the time and spectral characteristics results of section 6.2 it can be concluded that the MD DS/SSMA transmitter structure employing DSB CE-LI-RU filtered GCL CSSs can deliver a throughput of 1 kb/s in a bandwidth of only 31.5 KHz, proving that the transmitter poses to improve on system throughput, whilst being spectrally efficient for communication purposes.
2. From the Doppler as well as the path spectra results of section 6.3 it can be concluded that *Staphorst's* MFCS [3] is able to effectively reproduce realistic fading channel characteristics, creating an effective and user-definable test bed for all types of software-based mobile communication systems when expanded into a MU-MFCS.
3. From the uncoded and coded BER performance results for a fully functional wideband system (Fig. 6.6 and 6.7, chapter 6, section 6.4), it can be concluded that from the three codes being implemented by this study, the binary Hamming (7,4,3) block code is the weakest of the channel codes, whilst the rate 1/3 binary convolutional code is the strongest. It can also be seen that the error-correcting capability of the binary BCH (15,7,5) block code falls in between that of the

aforementioned codes. This fact is verified for both 1 as well as 5 user systems. It is also concluded that *Staphorst's* generic VA decoder [3] is fully functional and effective for both block and convolutional code structures.

4. The strong linear relationship between the PER and the threshold in the results on section 6.5, proves that the application of *Gooding's* extrapolation technique [3] is effective for frequency selective fading conditions (i.e. over MU-MFCS) and for an uncoded system with both 1 and 5 user systems. The PER vs. threshold results for a coded system in section 6.5.2 also proves that this linear relationship is retained when *Hagenauer*'s SOVA [39] for a coded system is included.
5. The accuracy and speed performance results of section 6.6 firstly proves that *Gooding's* BER estimation technique [12, 13] is a very effective technique to be used in a wideband communication system that employs real-time BER measurement techniques, because of the extreme acceleration of estimation speed without large degradation in accuracy. Secondly, it is concluded that the quadratic extrapolation technique is the better of the three techniques examined in this study (i.e. classic linear, improved linear and quadratic extrapolation), because of the increased accuracy without sacrificing calculation speed. Thirdly, it is concluded that the employment of the SOVA [39] in the system has BER estimation results that further support the fact that quadratic extrapolation is the stronger BER estimation technique, as well as the fact that the application of the SOVA is effective. These conclusions are all verified for both 1 and 5 user systems.
6. Lastly, from the real-time adaptive coding results of section 6.7 it can firstly be concluded that the complete system of this study is operating acceptably. Secondly, it can be observed from these results that switching to different channel codes in varying channel conditions will result in a more effective utilization of bandwidth by taking advantage of the time varying nature of the wireless communication channel. Thirdly, it can be concluded that the real-time BER measurement technique of this study proves to be faster than the classic Monte-Carlo technique with a small sacrifice of accuracy over large SNR per bit ranges. This conclusively proves that this technique is not only an effective solution to channel estimation for future systems, but also financially viable because of its simplicity.

7.2 FUTURE RESEARCH

This study not only opened up a large field of study into the workings of adaptive channel coded systems, but also into the application of real-time BER measurement for the estimation of channel conditions. The most relevant topics for future research into adaptive channel coding and real-time BER estimation that stem from this study are listed below:

1. In chapter 3, section 3.6, the choice of threshold values for all extrapolation techniques was not only chosen to be constant, but also the same for all techniques. The choice of thresholds is not necessarily the choice that ensures optimal trade-off between estimation speed and estimation accuracy. This assumption was only made in order to simplify and standardize the platforms for the evaluation of the speed and accuracy of these three extrapolation techniques. Future research should however, attempt to scrutinize the effects that the choice of threshold has on estimation speed and performance in order to determine the optimal choice of threshold.
2. The channel codes used in this study were chosen to be very simplistic in order to prove that the principles of adaptive channel coding are, in fact, practical. The choice of codes was also driven by the fact that there exists a direct correlation between the strength of the particular codes and their input/output relation. This fact is, however, not necessarily true for all channel codes. Thus, future research into the choice of proper channel codes for adaptive channel coding purposes should include measures to determine the optimal choice of codes that will ensure taking proper advantage of varying channel conditions.
3. Due to a lack of simulation hardware processing power, the results of chapter 6, section 6.6 only revealed the accuracy and speed performance of *Gooding's* extrapolation and BER estimation technique [12, 13] over MU-MFCS for a BER as low as approximately 10^{-4}. Future research into the application of Gooding's technique on MU-MFCS's should include performance results for BER much lower than 10^{-4}, which will further illustrate the applicability of this technique.
4. This study assumed that the reverse channel that passed receiver information back to transmitter, was ideal. This assumption is for all practical purposes, not true. Future research on adaptive coded systems should also include all perturbing effects due to reverse channel in the model.

5. It has been shown that the performance of the maximum a-posterior (MAP) algorithm far exceeds that of the SOVA [39]. Future research into the application of soft-output decoders to *Gooding's* extrapolation technique [12, 13] should include the application of this algorithm on the performance of the system.
6. The complete system of this study was designed to keep the estimated BER value within a predetermined range $a < BER < b$ (see chapter 4, section 4.4). The choice of this range is not only a function of the current application, but also of the performance of the available channel codes within the system. Future research into proper channel codes for adaptive channel coding purposes should thus include detailed measures on determining the choice of applicable channel codes and the relation between the code strength and size of the BER range.
7. This study only investigated the code switching performance for varying SNR per bit, whilst the number of users was kept constant throughout the simulation (i.e. either 1 or 5 user systems). Future research should include a varying number of system users in order to properly test the switching capability and real-time performance of the system.
8. No modulation (frequency up- or down conversion) were performed on the spreaded, multi-user signals during the simulation of the system. Future research should investigate the effects that different modulation techniques would have on system performance.
9. Perfect synchronization was assumed during the simulation of the system. Future research should include simulations for receiver synchronization for added realism.
10. Future research should investigate the performance of such a system in a hardware implementation (e.g. DSP, FPGA etc.) to investigate the characteristics of a real system..

APPENDIX A

REALIZATION OF THE DOPPLER SPREAD SPECTRAL SHAPING FILTERS

A.1 APPENDIX OVERVIEW

When surrounding objects in an environment are in motion, reflection of transmitted waves from the surrounding objects, or the relative motion between transmitter and receiver cause a shift in the frequency of the transmitted signal and ultimately a ‘smearing’ effect on the spectral characteristics of the signal [7], causing the signal to occupy more bandwidth than usual and also adds to the variations in the envelope and phase of the received signal. This effect is known as Doppler spreading [7, 8].The goal of this appendix is to firstly give a concise description of the mathematical Doppler spreading phenomenon, followed by a short explanation of the realization of a spectral shaping filter, called a Doppler filter, that is incorporated into the FFCS of chapter 2, section 2.3.1, in order to model realistic flat fading effects.

A.2 DOPPLER SPREADING

Doppler spread is a function of both the angle of arrival of the scattered signal waves and the relative motion between transmitter and receiver. The k^{th} Doppler spread associated with the corresponding k^{th} FFCS in the MFCS of Chapter 2, section 2.3.2, is defined as the single sided spectral density for a sinusoidal channel input signal. For an input signal that experiences a Doppler spread of $B_{Doppler}$, the PSD of the k^{th} FFCS’s output signal is approximated by [7, 8]:

$$PSD_{FFCS_k}(f) = \begin{cases} \dfrac{\sigma_{input}^2}{\pi\sqrt{(B_{Doppler})^2 - (f - f_c)^2}} & \left|f - f_c\right| \le B_{Doppler} \\ 0 & \left|f - f_c\right| > B_{Doppler} \end{cases}, \tag{A.1}$$

with σ_{input}^2 the average power of the input signal and f_c the carrier frequency. Fig. A.1 shows the classic Doppler spectrum for $B_{Doppler} / f_c = 0.5$ and $\sigma_{input}^2 = 1$.

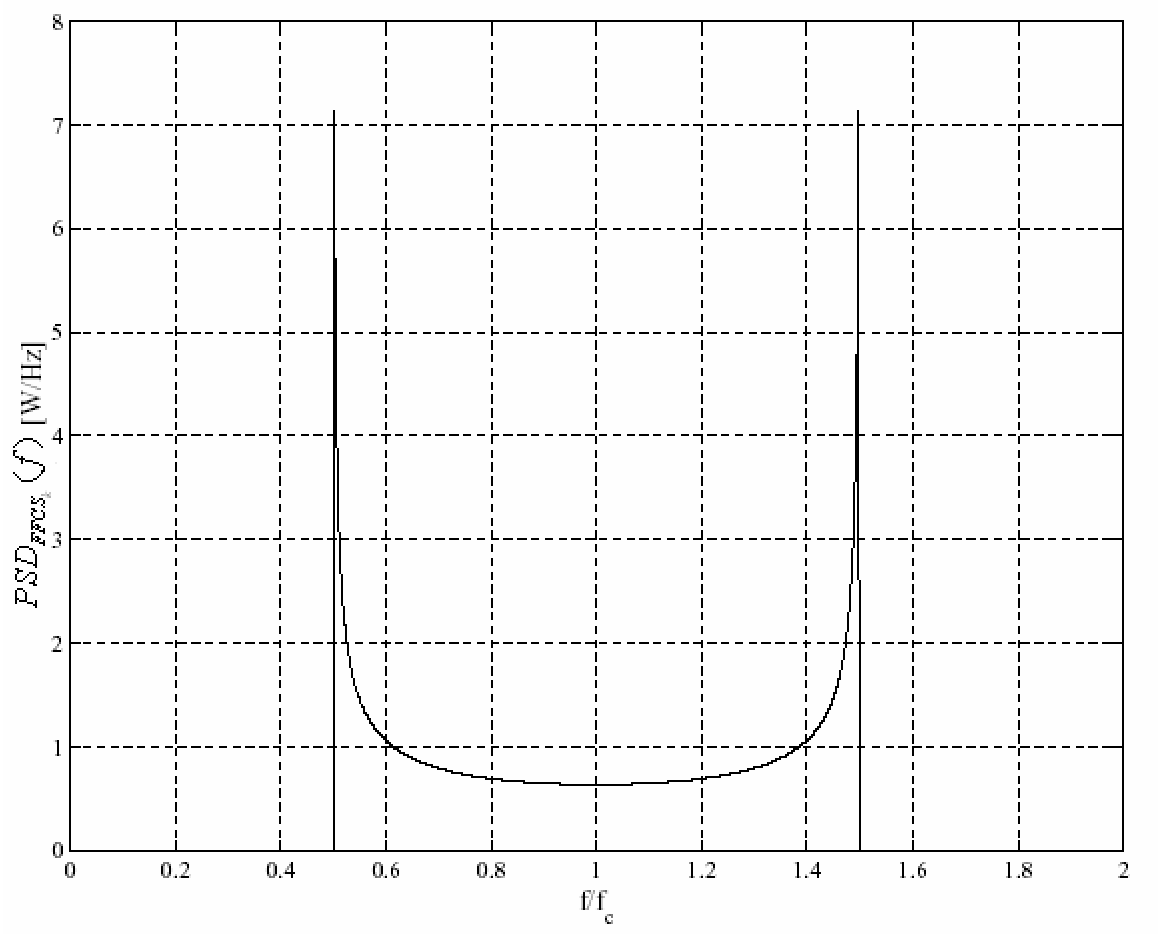


Figure A.1 Classic Doppler Spectrum for $B_{Doppler} / f_c = 0.5$ and $\sigma_{input}^2 = 1$ [Taken from [3]]

A.3 REALIZATION OF THE DOPPLER FILTER

We firstly assume a Doppler spread of $B_{Doppler}$ and a sampling period of T_s. Now, in chapter 2, section 2.3.1, the Doppler filters $DF_k^I(f)$ and $DF_k^Q(f)$ can be realized by using third order IIR filters with the following transfer function [32]:

$$DF(z) = K_{norm} \cdot \left(\frac{b_3 z^3 + b_2 z^2 + b_1 z + b_0}{a_3 z^3 + a_2 z^2 + a_1 z + a_0} \right). \tag{A.2}$$

In Eq. (A.2), the constants and variables are defined as follows [32]:

$$\begin{aligned}
b_0 &= b_3 = (2.\pi.B_{Doppler}.T_s)^3 \\
b_1 &= b_2 = 3.(2.\pi.B_{Doppler}.T_s)^3 \\
a_0 &= 8 + 8\pi.A_D.B_{Doppler}.T_s + 2.B_D.(2.\pi.B_{Doppler}.T_s)^2 + C_D.(2.\pi.B_{Doppler}.T_s)^3 \\
a_1 &= -24 - 8\pi.A_D.B_{Doppler}.T_s + 2.B_D.(2.\pi.B_{Doppler}.T_s)^2 + 3.C_D.(2.\pi.B_{Doppler}.T_s)^3 \\
a_2 &= 24 - 8\pi.A_D.B_{Doppler}.T_s - 2.B_D.(2.\pi.B_{Doppler}.T_s)^2 + 3.C_D.(2.\pi.B_{Doppler}.T_s)^3 \\
a_3 &= -8 + 8\pi.A_D.B_{Doppler}.T_s - 2.B_D.(2.\pi.B_{Doppler}.T_s)^2 + C_D.(2.\pi.B_{Doppler}.T_s)^3
\end{aligned} \tag{A.3}$$

with [3]:

$$\begin{aligned}
A_D &= 1.55 \\
B_D &= 1.090625 \quad . \\
C_D &= 0.9953125
\end{aligned} \tag{A.4}$$

Also, K_{norm} is a scaling factor that ensures unity power in the filter's output signal, depending on the input signal. Fig. A.2 illustrates the pole-zero plot of the filter transfer function of Eq. (A.2) for $B_{Doppler} = K_{norm} = T_s = 1$.

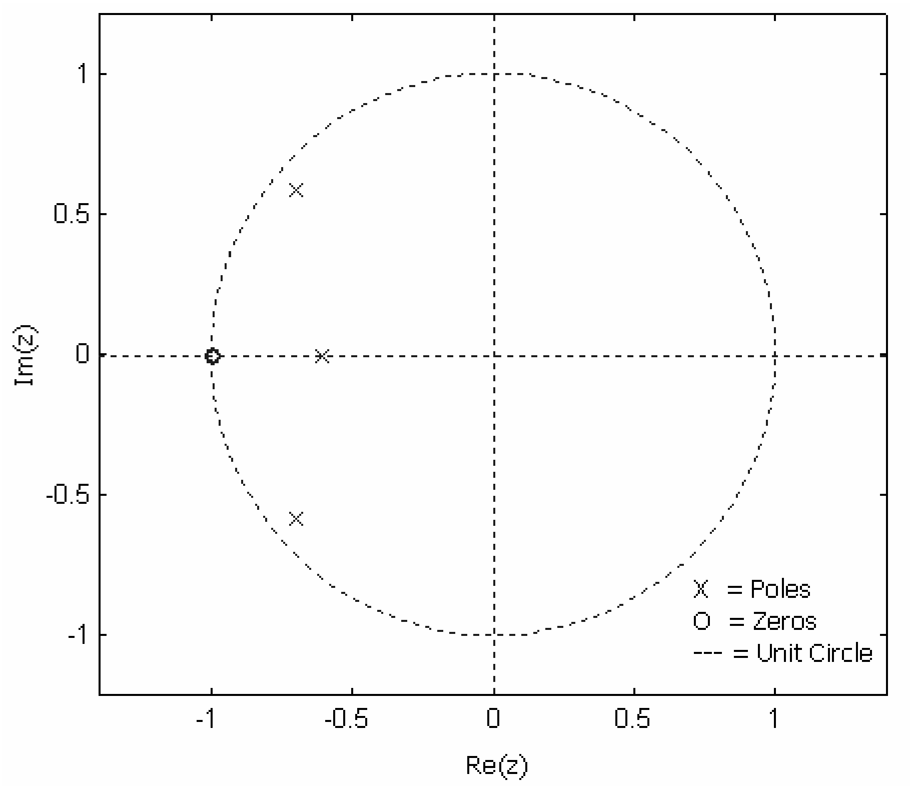


Figure A.2 Pole-Zero Plot of the Doppler Filter Transfer Function.

From figure A.2 it is clear that the filter is marginally stable because of the zeros that are close to the unit circle. Fig. A.3 shows the frequency response of $DF(z)/K_{norm}$ and the frequency axis normalized with respect to $B_{Doppler}$.

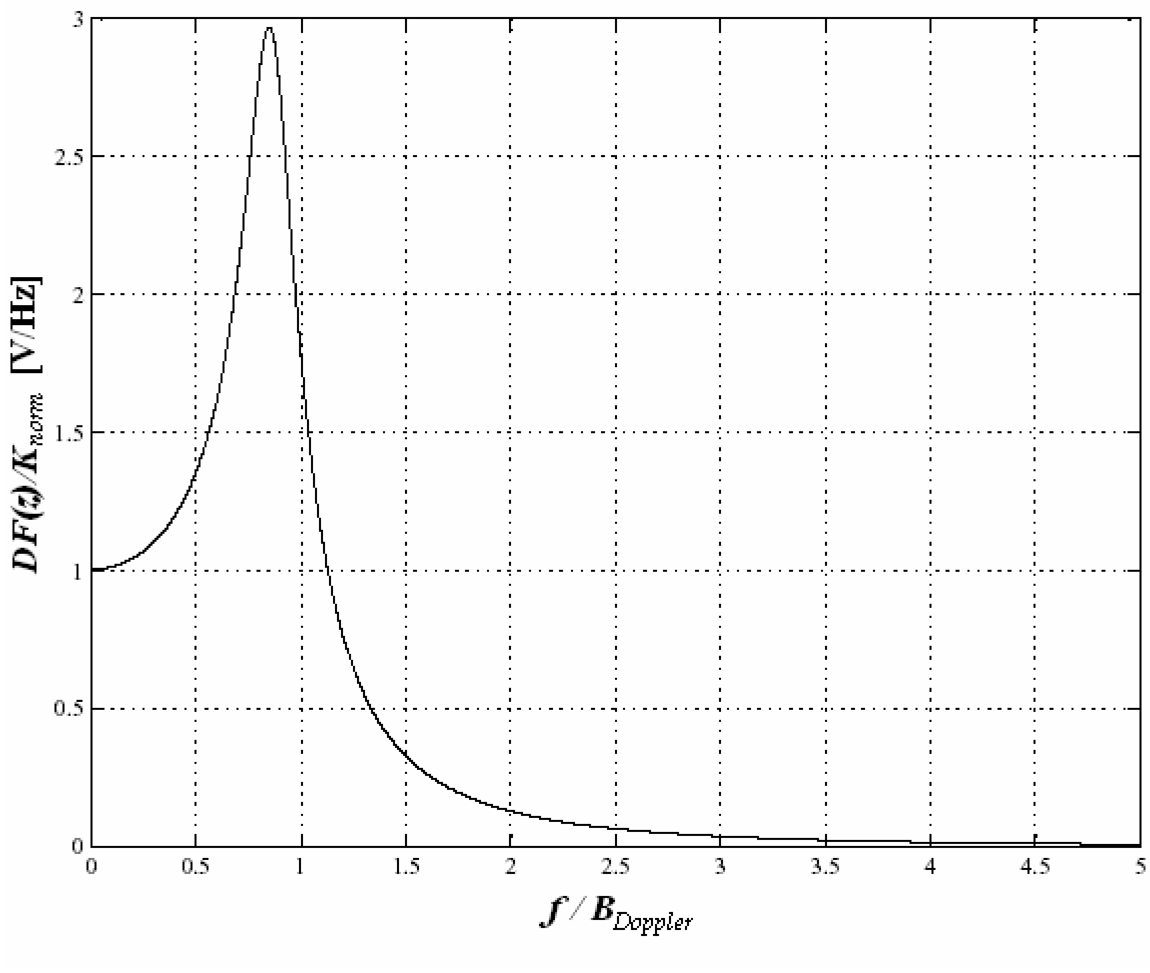


Figure A.3 Frequency Response of the Doppler Filter [Taken from [3]]

When comparing Fig's A.1 and A.3 it can be seen that the high spectral content in the tail-end of the frequency response of the Doppler filter $(f > B_{Doppler})$ will cause a more severe Doppler spread than by the mathematical model of section A.2. Using higher order filters to improve this problem will, however, result in increased simulator complexity.

APPENDIX B

DSB CE-LI-RU FILTERED GCL COMPLEX SPREADING SEQUENCES

B.1 APPENDIX OVERVIEW

There exists a wide research field on the generation of orthogonal or low cross-correlation sequences. The existing problem with the usual binary sequences, like *Walsh* orthogonal codes, *Gold*, and *Kasami* sequences is that their auto- and cross-correlation properties compared to their family size make them, in many cases, unsuited for use. This has caused SS researchers to look more toward the use of non-binary and *Complex Spreading Sequences* (CSS) which enables the generation of *Constant Envelope* (CE), as well as *Single Sideband* (SSB) signals. The goal of this appendix is to give a concise overview of the nature and generation of the complex spreading sequences used for the DS/SSMA purposes of this study, known as DSB CE-LI-RU filtered GCL complex spreading sequences. These sequences are derived from the family of complex unfiltered Zadoff-Chu sequences. Thus, before the DSB sequences are considered, a short overview of unfiltered Zadoff-Chu complex spreading sequences will be presented.

B.2 UNFILTERED ZADOFF-CHU COMPLEX SPREADING SEQUENCES

Let us assume that $\overline{C}_{ZC}^{z}[x]=\{C_{ZC}^{z}[0],C_{ZC}^{z}[1],C_{ZC}^{z}[2],...,C_{ZC}^{z}[Z_{length}-1]\}$ is the vector of chips in the z^{th} unfiltered ZC CSS with length Z_{length}. If we define $j=\sqrt{-1}$, then the x^{th} chip in the z^{th} ZC CSS sequence $C_{ZC}^{z}[x]$, is generated as follows [34, 35]:

$$C_{ZC}^{z}[x]=\begin{cases} e^{\frac{j.\pi.\varpi.x^{2}}{Z_{length}}} & for \quad Z_{length} \quad even \\ e^{\frac{j.\pi.\varpi.x(x+1)}{Z_{length}}} & for \quad Z_{length} \quad odd \end{cases}, \qquad \text{(B.1)}$$

with ϖ an integer value that is relatively prime to Z_{length}. The family size of the length-Z_{length} ZC CSS is defined by [3]:

$$Z_{size} = 1 + \sum_{\varpi=2}^{Z_{length}-1} \begin{cases} 1 & for \quad Z_{length}.\mathrm{mod}(\varpi) \neq 0 \\ 0 & for \quad Z_{length}.\mathrm{mod}(\varpi) = 0 \end{cases}. \tag{B.2}$$

The largest families are generated when Z_{length} is a prime, in which case the family size will be $Z_{size} = Z_{length} - 1$ [34, 36].

B.3 DSB CE-LI-RU FILTERED GCL COMPLEX SPREADING SEQUENCES

It can be easily shown that the ZC sequences of section B.2 contain all frequencies in the range $[0, Z_{size} / T_c)$ [Hz], with T_c, the chip duration [35], making the bandwidth of these sequences a function of family size. The dependency of the ZC sequence on the sequence index ϖ can be removed whilst simultaneously band-limiting the sequence by incorporating a $\mathrm{mod}(2.\pi)$ constraint. This will result in a *Chu* sequence chip vector [3]:

$$C_{Chu}^{z}[x] = \begin{cases} e^{\frac{j.\pi.\varpi.x^2}{Z_{length}}} \ \mathrm{mod}(2.\pi) & for \quad Z_{length} \quad even \\ e^{\frac{j.\pi.\varpi.x(x+1)}{Z_{length}}} \ \mathrm{mod}(2.\pi) & for \quad Z_{length} \quad odd \end{cases}. \tag{B.3}$$

It can also be shown that the bandwidth of the *Chu* sequences is $1/T_c$ [Hz] [34]. Finally, DSB CE-LI-RU filtered GCL sequences are obtained by filtering the *Chu* sequences $C_{Chu}^{z}[x]$ with a linearly interpolating root-of-unity filter [36, 37] that achieves the minimum Nyquist bandwidth of $1/(2.T_c)$ [Hz]. The family size of these sequences is also given by Eq. (B.2). Fig. B.1 illustrates the real and imaginary parts of a length-63 DSB CE-LI-RU filtered GCL sequence with $\varpi = 1$, a chip rate of $f_c = 1/T_c = 63000$ Hz at 16 samples per chip. Fig. B.2 illustrates the complex envelope of the sequence, proving that it is constant. By examining this figure is becomes clear that when these sequences are utilized in a system, partially linear power amplifiers can be efficiently utilized because of the constant instantaneous power output signal.

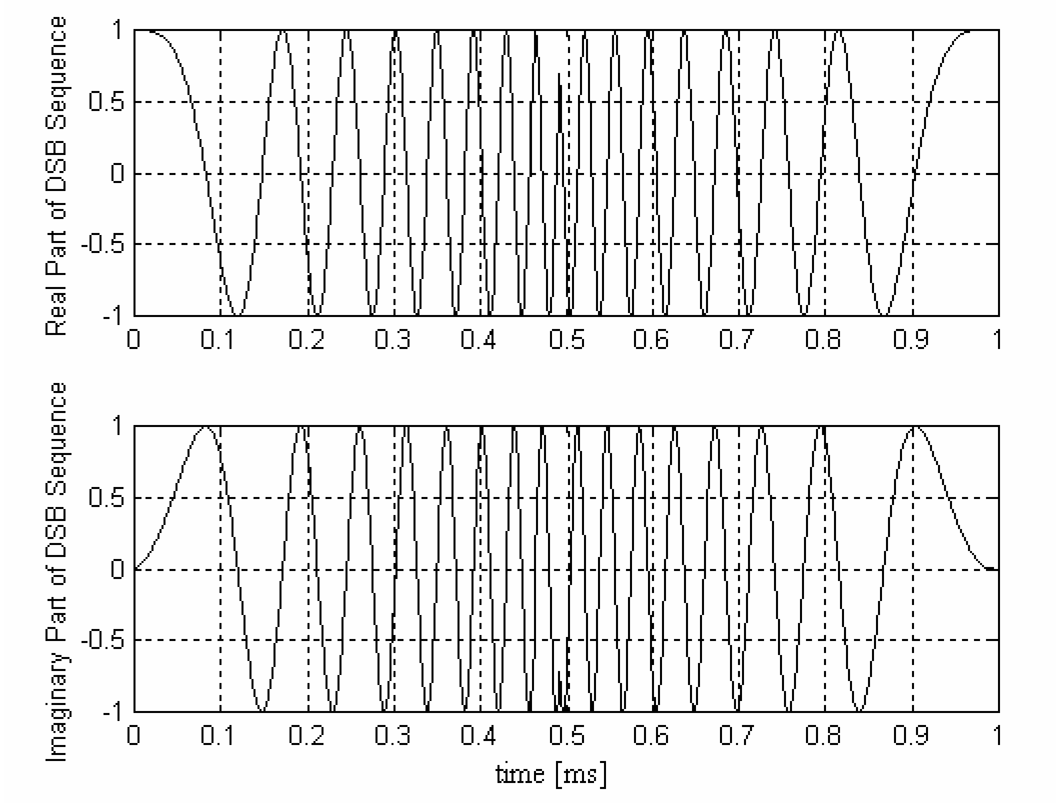


Figure B.1 Real and Imaginary parts of a length-63 DSB CE-LI-RU filtered GCL Sequence with $\varpi = 1$, $f_c = 1/T_c = 63000$ Hz and 16 Samples per Chip

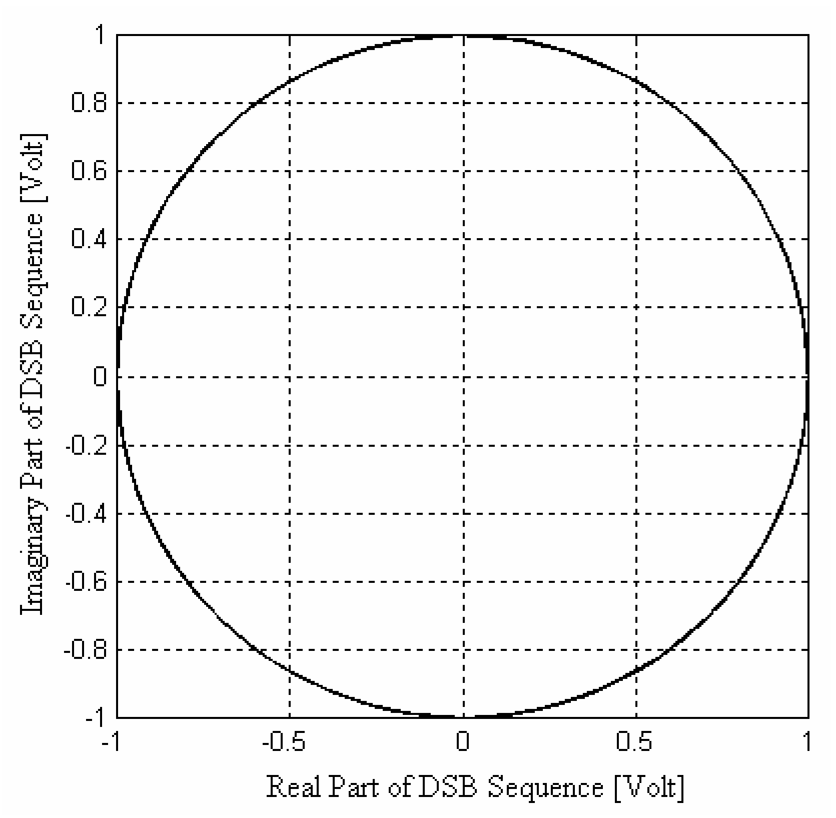


Figure B.2 Envelope of a Length-63 DSB CE-LI-RU filtered GCL Sequence with $\varpi = 1$, $f_c = 1/T_c = 63000$ Hz and 16 Samples per Chip

APPENDIX C

SIMULATION SOFTWARE INDEX

C.1 APPENDIX OVERVIEW

In order to create a fully functional adaptive coded communication system to evaluate all results presented in chapter 6, an extensive set of software tools was developed that created each of the functional units of the system. C++ was chosen as the platform to develop each of the functional units of the total system. Thus, each of these subsystems exists as a compiled C++ class.

This appendix will present a list of all the compiled C++ classes that exist in the simulation software that make up the complete system. This list will include the names of the respective header and *.cpp* files. Since the author of this dissertation is not the author of all the software modules, this list will include the author of a particular class. Because of the 'black box' approach for the chosen channel coding unit, the author approached Mr. Leonard Staphorst [3] and with his consent, used the pre-developed classes used in his study in [3]. The cases in which the required software classes did not exist, Mr. Staphorst developed them, and was extensively used by the author.

C.2 C++ CLASSES

Table C1: List of All C++ Classes Developed and Used Throughout This Study

C++ Class	Class Description	Author
Adaptive_Coder.h Adaptive_Coder.cpp	Class that employs all relevant channel codes and interfaces with each of the respective channel code classes Also performs all code switching actions	Estian Malan

BER_Meter.h BER_Meter.cpp	Bit-error-rate meter class	Estian Malan
BCH_Code.h BCH_Code.cpp	Combines and interfaces with all relevant BCH code classes to create a complete (15,7,5) BCH binary block code functional unit	Estian Malan
Hamming_Code.h Hamming_Code.cpp	Combines and interfaces with all relevant Hamming code classes to create a complete (7,4,3) binary Hamming block code functional unit	Estian Malan
BCJR_Trellis.h BCJR_Trellis.cpp	Generic BCJR trellis class	Leonard Staphorst
Blockwise_SOVA_Block_Code.h Blockwise_SOVA_Block_Code.cpp	Soft-output Viterbi algorithm block code decoder class	Leonard Staphorst
Generic_Block_Coder.h Generic_Block_Coder.cpp	Generic $(\delta, \varsigma, d_{\min})$ block code encoder class	Leonard Staphorst
GF_2_to_m_calculator.h GF_2_to_m_calculator.cpp	Galois field mathematics calculator class	Leonard Staphorst
Comp_Channel.h Comp_Channel.cpp	Complex QPSK-based flat-fading channel model class	Estian Malan

Thirdrate_Convolutional_Code.h Thirdrate_Convolutional_Code.cpp	Combines and all relevant convolutional code classes to create a complete rate 1/3 binary convolutional code functional unit	Estian Malan
Conv_Coder.h Conv_Coder.cpp	Generic rate δ / ς convolutional code encoder class	Leonard Staphorst
Conv_Trellis.h Conv_Trellis.cpp	Rate δ / ς convolutional code trellis class	Leonard Staphorst
Sliding_Window_SOVA_Conv.h Sliding_Window_SOVA_Conv.cpp	Sliding window soft-output Viterbi algorithm convolutional code decoder class	Leonard Staphorst
Doppler_IIR.h Doppler_IIR.cpp	IIR Doppler filter class	Estian Malan
Gauss_Gen.h Gauss_Gen.cpp	Gauss noise sample generator class	Estian Malan
Integrate_Dump.h Integrate_Dump.cpp	Integrate-and-dump device class	Estian Malan
MPFC.h MPFC.cpp	Multi-user multipath fading channel simulator class	Estian Malan
PMU.h PMU.cpp	Performance monitoring unit, or QoSMU class	Estian Malan
Rand_Bit_Gen.h Rand_Bit_Gen.cpp	Random bit generator class	Estian Malan
Receiver.h Receiver.cpp	Complex receiver class	Estian Malan

Scaled_Noise_Gen.h Scaled_Noise_Gen.cpp	Class that generates correctly scaled E_b / N_o noise samples	Estian Malan
Shift_Reg.h Shift_Reg.cpp	Generic shift-register class	Estian Malan
SS_Tx_Rx.h SS_Tx_Rx.cpp	Spread spectrum spreading/despreading class	Estian Malan

Matlab was also extensively used for the graphical presentation of data, but because the execution of these processes were all performed form the Matlab command prompt, no files were created, and are therefore not included in this appendix.

REFERENCES

[1] C. Shannon, “A Mathematical Theory of Communications,” *Bell Systems Technical Journal*, vol. 27, October 1948.

[2] G. D. Forney, Jr., “The Viterbi Algorithm,” in *Proc. of the IEEE*, vol. 61, no. 3, pp. 268–277, March 1973.

[3] L. Staphorst, “Viterbi Decoded Linear Block Codes for Narrowband and Wideband Wireless Communication over Mobile Fading channels,” Master’s dissertation, University of Pretoria, 2005.

[4] R. L. Peterson, R. E. Ziemer, and D. E. Borth, *Introduction to Spread Spectrum Communications*. Upper Saddle River, NJ, USA: Prentice Hall, 1995.

[5] IS-95 Interim Standard: *’An Overview of the Application of Code Division Multiple Access (CDMA) to Digital Cellular Systems and Personal Cellular Networks’*, Qualcomm Inc., May 1992.

[6] J. G. Proakis, *Digital Communications*. New York, NY, USA: McGraw-Hill, Fourth ed., 2001.

[7] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice Hall, Second ed., 2002.

[8] K. Feher, *Wireless Digital Communications - Modulation and Spread-Spectrum Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, First ed., 1995.

[9] W. Stallings, *Wireless Communications and Networks.* Upper Saddle River, NJ, USA: Prentice-Hall, First ed., 2002.

[10] H.A. Sunkenberg and L.E. Jankauskas, “Performance Assessment of High-Speed Digital Transmission Systems,” in *National Telecommunications Conference Rec.*, 1976, pp. 51.5-1 – 51.5-5.

[11] K.S. Shanmugam and P. Balaban, "A Modified Monte-Carlo Simulation Technique for the Evaluation of Error Rate in Digital Communications Systems," *IEEE Transactions on Communication Technology*, vol. COM-28, pp 1916-1924, November 1980.

[12] D.J. Gooding, "Performance Monitor Techniques for Digital Receivers Based on Extrapolation of Error Rate," *IEEE Transactions on Communication Technology*, vol. COM-16, pp 380-387, June 1968.

[13] D.J. Gooding, "Performance Monitor Techniques for Digital Receivers," in *Proceedings of the National Electronics Conference*., 1969, pp. 432-437.

[14] B.J. Leon, J.L. Hammond, P.A. Vena, W.E. Sears and R.T. Kitahara, "A Bit Error Rate Monitor for Digital PSK Links," *IEEE Transactions on Communication Technology*, vol. COM-23, No. 5, pp 518-525, May 1975.

[15] K. Feher and A. Bandari, "Pseudoerror On-line Monitoring: Concept, Design and Evaluation," *Canadian Electronic Engineering Journal*, vol. 2, pp. 33-36, April 1977.

[16] J. M. Keelty and K. Feher, "On-line Pseudo-error Monitors for Digital Transmission Systems," *IEEE Ttransactions on Communications*, vol. COM-26, No. 8, pp.1275-1282, Aug. 1978.

[17] E.A. Newcombe and S. Pasupathy, "Error Rate Monitoring in a Partial Response System," *IEEE Transactions on Communications*, vol. COM-28, No. 7, pp.1052-1061, July 1980.

[18] E.A. Newcombe and S. Pasupathy, "Error Rate Monitoring for Digital Communications," *Proceedings of the IEEE*, vol. 70, No. 8, pp.805-828, August 1982.

[19] J.B. Scholz, "Error Performance Monitoring of Digital Communication Systems," *Aust. Telecommun. Res.*, vol. 25, no. 2, 1991.

[20] S. Haykin, *Communication Systems.* New York, NY, USA: Wiley, Fourth ed., 2001.

[21] D. W. Hsiung and J.F. Chang, “Performance of Multi-code CDMA in a Multipath Fading Channel,” in *Proceedings in IEEE Communications*, vol. 147, Issue 6, pp.365-370, December 2000.

[22] D. I. Kim and V. K. Bhargava, “Performance of Multidimensional Multicode DS-CDMA using Code Diversity and Error Detection,” *IEEE Transactions on Communications, May 2001*, vol. 49, No. 5, pp. 875-887.

[23] H. Y. Kong and H. W. Bae, “Design of Multidimensional Multicode SS System using HCOC & AOCG for High-speed Data Transmission,” in *TENCON 2004. 2004 IEEE Region 10 Conference*, *21-24 November, 2004,* vol. B, pp. 473-476.

[24] I. Chih-Lin, G .P. Pollini, L. Ozarow and R. D. Gitlin, “Performance of Multi-code CDMA Wireless Personal Communications Networks,” in *Proceedings of the 45th IEEE Vehicular Technology Conference, 25-28 July, 1995*, vol. 2, pp. 907-911.

[25] B. Wichmann and D. Hill, “Building a Random Number Generator,” *Byte,* pp. 127-128, March 1987.

[26] G. Marsaglia and T.A. Bray, “A Convenient Method for Generating Normal Variables,” *SIAM Rev*., Vol. 6, pp. 260-264, 1964.

[27] R. E. Larson, R. P. Hostetler and B. H. Edwards, *Calculus,* Fifth ed. Lexington, MA: Heath and Company, 1994.

[28] D. G. Zill and M. R. Cullen, Advanced Engineering Mathematics, First ed. Sudbury, MA: Jones and Bartlett Publishers, 1996.

[29] M. C. Jeruchim, “Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communication Systems,” *IEEE Journal on Selected Areas in Communications*, vol. SAC-2, no. 3, pp. 153-170, 1984.

[30] S. M. Berber, “An Automated Method for BER Characteristics Measurement,” *IEEE Transactions on Instrumentation and Measurement, April 2004,* vol. 53, no. 2, pp.575-580.

[31] W. H. Büttner, L. Staphorst, and L. P. Linde, “Trellis Decoding of Linear Block Codes,” in *Proc., IEEE COMSIG’98*, (Cape Town, South Africa), pp. 171–174, 7-8 September 1998.

[32] L. Staphorst and L. P. Linde, “Performance Evaluation of Viterbi Decoded Reed-Solomon Block Codes in Additive White Gaussian Noise and Flat Fading Channel Conditions,” in *Proc., IEEE WCNC’2002*, (Orlando, Florida, USA), 17-21 March 2002.

[33] J. S. Swarts, “Aspects of Multipath Channel Characterization,” Master’s Thesis, Rand Academic University, November 1998.

[34] M. Jamil, L. P. Linde, J. E. Cilliers, and D. J. van Wyk, “Comparison of Complex Spreading Sequences Based on Filtering Methods and Mean Square Correlation Characteristics,” *Transactions of the SAIEE*, vol. 89, no. 3, pp. 98–112, September 1998.

[35] R. L. Frank and S. A. Zadoff, “Phase Shift Pulse Codes with Good Periodic Correlation Properties,” *IRE Transactions on Information Theory*, vol. IT-7, pp. 381–382, October 1962.

[36] M. Jamil, “Comparative Study of Complex Spreading Sequences for CDMA Applications,” Master’s Thesis, University of Pretoria, May 1999.

[37] M. P. Lötter and L. P. Linde, “Constant Envelope Filtering of Complex Spreading Sequences,” *IEEE Electronics Letters*, vol. 31, no. 17, pp. 1406–1407, 17 August 1995.

[38] M. P. Lötter, “A Generalised Linear Root-of-Unity Interpolation Filter,” in *Proc., IEEE COMSIG’ 95*, (University of Pretoria, Pretoria, South Africa), pp. 43–46, September 1995.

[39] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and its Applications," in *Proc., IEEE GLOBECOM*, (Dallas, TX, USA), pp. 47.1.1–47.1.7, 27-30 November 1989.

[40] L. Hanzo, T.H. Leiw and B.L. Yeap, *Turbo Coding, Turbo Equalization and Space-Time Coding for Transmission over Fading Channels.* West Sussex, UK: Wiley, 2002.

[41] J. Wolf, "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis," *IEEE Transactions on Information Theory*, vol. IT-24, pp. 76–80, January 1978.

[42] T. Matsumoto, "Trellis Decoding of Linear Block Codes in Digital Mobile Radio," *38th IEEE Conference on Vehicular Technology*, pp. 6-11, June 1988.

[43] L. H. C. Lee and L. W. Lee, "Scarce-state-transition Error-trellis Decoding of Block Codes with BPSK Signals," *IEEE Electronics Letters*, vol. 30, no. 14, pp. 1120–1121, 7 July 1994.

[44] B. Honary, G. Markarian and M. Darnell, "Low-Complexity Trellis Decoding of Linear Block Codes," in *Proc., IEE Communications*, vol. 142, no. 4, pp. 338-342, March 1999.

[45] M.P.C. Fossorier, S. Lin and D. Rhee, "Bit-error Probability for Maximum-Likelihood Decoding of Linear Block Codes and Related Soft-Decision Decoding Methods," *IEEE Transactions on Information Theory*, vol. 44, no. 7, pp. 3083–3090, November 1998.

[46] A.A. Luna, F.M. Fontaine and S.B. Wicker, "Iterative Maximum-likelihood Decoding of Linear Block Codes," *IEEE Transactions on Communications*, vol. 47, no. 3, pp. 338–342, March 1999.

[47] D. Yuan, C. Gao and L Zhang, "Performance of Trellis Decoding of Block Codes and the Practice in Compressed Image Communication over Rayleigh fading Channel," in *Proc., 21st Century Military Communications Conference*, vol. 1, pp. 407-411, 22-25 October 2000.

[48] S.K. Shin, S.I. Lee and S.P. Lee, "Evaluation of Block Turbo Code Performance with the Reduced Search Trellis Decoding Method," in *Proc., IEE Communications*, vol. 148, no. 3, pp. 125-131, June 2001.

[49] L Zhau and J. Huber, "Design and Multiple Trellis Decoding for Concatenated Constant Envelope Non-Orthogonal Space-Time Block Codes," *10th International Conference on Telecommunications*, vol. 2, pp. 1184-1188, 23 February – 1 March 2003.

[50] E. Bertrand and F. Labeau, "Simplified Trellis Decoding of Block Codes by Selective Pruning," *Conference Record of the Thirty-Seventh Conference on Signals, Systems and Computers*, vol. 1, pp. 1115-1119, November 2004.

[51] E. Malan and L. Staphorst, "A Software Simulation Study of a MD DS/SSMA Communication System with Adaptive Channel Coding," in *Proc., IEEE GLOBECOM*, (Cap Esterel, Côte d'Azur, France), on CD, 28 August-3 September 2006.