UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Low Complexity Iterative MLSE Equalization in Extremely Long Rayleigh Fading Channels

By

Hermanus Carel Myburgh

Study Leader: Professor J.C. Olivier

Submitted in partial fulfillment for the degree

Master of Engineering (Electronic)

in the
Department of Electronic, Electrical and Computer Engineering
in the
Faculty of Engineering, Built Environment and Information Technology

University of Pretoria

May 2010

"I am the Way, the Truth and the Life..."

Jesus of Nazareth

Summary

Low Complexity Iterative MLSE Equalization
in Extremely Long Rayleigh Fading Channels

by

Hermanus Carel Myburgh

Study Leader: Professor J.C. Olivier
Department of Electronic, Electrical and Computer Engineering
Master of Engineering (Electronic)

In mobile wireless communication systems, the transmitted signal is subjected to various impediments, among which intersymbol interference (ISI) poses a major challenge to communication system designers. ISI is the result of the dispersive nature of a wireless communication channel, causing multiple delayed copies of the original transmitted signal to arrive at the receiver. Since the communication channel acts like a finite impulse response (FIR) filter on the transmitted data, the effect of the channel needs to be reversed in order to reconstruct the information contained in the original transmitted signal. The process of reversing the effect of the channel on the transmitted signal is known as detection or equalization. The computational complexity of conventional optimal equalizers is linear in the length of the transmitted data block and exponential in the channel impulse response (CIR) length. When the bandwidth of the communication signal is low, the duration between the first and the last arrival of the same signal, known as the channel delay spread, only spans a small number of symbols, which implies that the CIR is short. In these systems conventional equalizers can be used to optimally mitigate the effect of ISI. In high bandwidth communication systems, however, the channel delay spread may span tens to hundreds of symbols, which implies very long CIR lengths, rendering conventional equalizers infeasible due to severe computational strain. In this thesis a low complexity maximum likelihood sequence estimation (MLSE) equalizer is developed which has computational complexity quadratic in the length of the transmitted data block and approximately independent of the CIR length for practical single-carrier communication systems. This equalizer is therefore able to equalize signals in systems with hundreds of interfering symbols, at a fraction of the computational complexity of conventional equalizers. Using the Hopfield neural network (HNN) as foundation, this equalizer is applied to underwater communication systems, code division multiple access (CDMA) communication systems, multiple antenna communication systems, and Turbo Equalization, with great success in this thesis.

Keywords:
Equalization, low complexity, Hopfield neural network, Rayleigh fading.

Opsomming

Lae-Kompleksiteit Iteratiewe MWSE Vereffening
in Buitengewoon Lank Rayleigh Duinende Kanale

deur

Hermanus Carel Myburgh

Studieleier: Professor J.C. Olivier
Departement van Elektriese, Elektroniese en Rekenaaringenieurswese
Meester van Ingenieurswese (Elektronies)

In mobiele draadlose kommunikasiestelsels is die gestuurde sein onderhewig aan verskillende belemmeringe, waarvan inter-simbool inmenging (ISI) 'n groot uitdaging vir kommunikasiestelselingenieurs inhou. ISI is die gevolg van die dispersiewe eienskap van 'n draadlose kommunikasiekanaal, wat veroorsaak dat menigte verdraagte kopiëe van dieselfde sein by die ontvanger arriveer. Aangesien die kommunikasiekanaal soos 'n beperkte impuls reaksie (BIR) filter op die gestuurde sein inwerk, moet die effek van die kanaal omgekeer word sodat die oorspronklike data in die sein gerekonstrueer kan word. Hierdie omkeerproses is bekend as deteksie of vereffening. Die verwerkingskompleksiteit van konvensionele optimale effenaars is lineêr tot die lengte van die getransmitteerde datablok en eksponensiëel verwand aan die lengte van die kanaal impulse reaksie (KIR). Wanneer die bandwydte van die sein laag is, strek die duur tussen die eerste en die laaste arrivering van dieselfde seinkomponent, bekend as die kanaalvertra-gingsverspreiding, slegs oor 'n klein hoeveelheid simbole, wat impliseer dat die KIR kort is. In hierdie stelsels kan konvensionele effenaars gebruik word om die effek van ISI optimaal te verwyder, maar in hoë-bandwydte stelsels kan die kanaal vertrag-ingsverspreiding oor tiene of selfs honderde simbole strek, wat veroorsaak dat konvensionele effenaars onbruikbaar is as gevolg van die uitermatige hoë verwerkingslas. In hierdie tesis word 'n lea-kompleksiteit maksimum waarskynlikheid sekwensie estimasie (MWSE) effenaar onwikkel met verwerkingskompleksiteit wat kwadraties verwant aan die lengte van die getransmitteerde datablok is en ongeveer onafhanklik van die lengte van die KIR vir praktiese enkeldraer kommunikasiestelsels. Gevolglik het die effenaar die vermoë om seine te vereffen in stelsels waar daar honderde inmengende simbole is teen 'n fraksie van die verwerkingskompleksiteit van konvensionele effenaars. Hierdie effenaar, wat die Hopfield neurale netwerk (HNN) as fondasie gebruik, word in hierdie tesis met groot sukses toegepas op onderwater kommunikasiestelsels, kode verdeling meervoudige toegang (KVMT) kommunikasiestelsels, vervoudige antenna kommunikasiestelsels, en Turbo Vereffening.

Sleutelwoorde:
Vereffening, lae-kompleksiteit, Hopfield neurale netwerk, Rayleigh duining.

Acknowledgements

The author of this thesis would like to thank the following people and institutions:

- Professor J.C. Olivier, whom I have come to know not only as a study leader, but also as a person. I want to thank him for his guidance and thoughtful advice since 2007, and for all the good times we have had during overseas trips.
- Telkom Centre of Excellence (CoE) for granting me a bursary. Without their financial support this would not have been possible.
- Centre for Teletraffic Engineering in an Information Society (CeTEIS) for their financial assistance for international conference attendance.
- Everyone who supported me troughout my postgraduate studies, and especially during the writing of this thesis.

Contents

List of Figures

List of Tables

Abbreviations

AWGN	Additive White Gaussian Noise
BER	Bit-Error Rate
BPSK	Binary Phase-Shift Keying
CDMA	Code Division Multiple Access
CIR	Channel Impulse Response
CSI	Channel State Information
DDFE	Delayed Decision Feedback Equalizer
DFE	Decision Feedback Equalizer
DSSS	Direct Sequence Spread Spectrum
ECC	Error-Correction Coding
EEG	Electroencephalograph
HNN	Hopfield Neural Network
ICI	Intercarrier Interference
ISI	Intersymbol Interference
LLR	Log-Likelihood Ratio
LOS	Line Of Sight
M-QAM	M-ary Quadrature Amplitude Modulation
MAI	Multiple Access Interference
MAP	Maximum Aposterior Probability
MC	Multi-Carrier
MIMO	Multiple Input Multiple Output
ML	Maximum Likelihood
MLSE	Maximum Likelihood Sequence Estimation
MMSE	Minimum Mean Square Error
MRC	Magnetic Recording Channel

MRI	Magnetic Resonance Imaging
MSE	Mean Square Error
MWC	Microwave Channel
NP	Nondeterministic Polynomial
OFDM	Orthogonal Frequency Division Multiplexing
PAPR	Peak-to-Average Power Ratio
PDF	Probability Density Function
PDP	Power Delay Profile
PLC	Power Line Channel
QAM	Qaudrature Amplitude Modulation
SC	Single-Carrier
SFE	Soft Feedback Equalizer
SIR	Signal-to-Interference Ratio
SISO	Single Input Single Output
SNR	Signal-to-Noise Ratio
TSP	Travelling Salesman Problem
UAC	Underwater Acoustic Channel
VA	Viterbi Algorithm

This thesis is dedicated firstly to my Father in heaven, to whom I owe my life, and secondly to my earthly father, who exchanged the perishable for the imperishable. I hope to see them soon.

Chapter 1

Introduction

Multipath propagation in communication systems is a challenge that has received much attention over the last few decades. This phenomenon, caused by the arrival of multiple delayed copies of the transmitted signal at the receiver, results in intersymbol interference (ISI), severely distorting the transmitted signal at the receiver.

Detection, also known as equalization, is necessary in the receiver to mitigate the effect of ISI, in order to produce reliable estimates of the transmitted information. In the early 1970's, Forney proposed an optimal equalizer [1] based on the Viterbi algorithm (VA) [2], able to optimally estimate the most likely sequence of transmitted symbols. The VA was proposed a few years before for the optimal decoding of convolutional error-correction codes. Shortly afterward the BCJR algorithm [3], also known as the Maximum Aposterior Probability (MAP) algorithm, was proposed, able to produce optimal estimates of the transmitted symbols.

The development of an optimal equalizer was an extraordinary achievement, as it enabled wireless communication system designers to design receivers that can optimally detect a sequence of transmitted symbols, corrupted by ISI, for the first time. Although the Viterbi Maximum Likelihood Sequence Estimation (MLSE) algorithm and the MAP algorithm estimate the transmitted information with maximum confidence, the computational complexity is prohibitive, increasing exponentially with an increase in channel memory [4].

The computational complexity of these optimal algorithms is linear in the data block length and exponential in the channel memory length, rendering them practically infeasible in communication systems with moderate to large bandwidth. For this reason, communication system designers were forced to use suboptimal equalization algorithms to alleviate the computational strain of optimal equalization algorithms, sacrificing system performance in the process.

In this thesis a low complexity MLSE equalizer is developed. Using the Hopfield Neural Network (HNN) [5] as the foundation, this equalizer has complexity quadratic in the data block length and approximately independent of the channel memory length for practical

single-carrier (SC) communication systems.[1] Its low computational complexity, compared to that of the Viterbi MLSE and MAP algorithms, is due to the high parallelism and high level of interconnection between the neurons of its underlying HNN structure. This equalizer, dubbed the *HNN MLSE equalizer*, iteratively mitigates the effect of ISI, producing near-optimal estimates of the transmitted symbols in systems with extremely long memory.

1.1 Motivation

Although the focus has shifted from SC to multi-carrier (MC) communication systems in recent years, it is still important to have an equalizer that is able to equalize signals in systems with possibly hundreds of resolvable multipath components. In MC communication systems the symbol period is increased so that the signal delay is absorbed within one symbol, eliminating the problem of ISI. However, in the case of high bandwidth MC communication, where the symbol periods are shorter, the problem of ISI cannot be ignored. Equalization is therefore necessary.

There are a number of communication channels that have extremely long memory. Among these are underwater acoustic channels (UAC), magnetic recording channels (MRC), power line channels (PLC) and micro-wave channels (MWC) [6]. In these channels there may be hundreds of multipath components, leading to severe ISI [6, 7, 8, 9]. Also, in spread spectrum systems, where information symbols are transformed into chips,[2] the signal suffers from interchip interference, analogous to ISI, that may also involve tens or hundreds of chips. Due to the massive amount of interfering symbols, the use of conventional optimal equalization algorithms is infeasible.

A number of suboptimal equalization algorithms have been considered where optimal equalizers cannot be used due to constraints on the processing power. Although these equalizers allow for decreased computational complexity, their performance is not comparable to that of optimal equalizers. The minimum mean squared error (MMSE) equalizer and the decision feedback equalizer (DFE), and variants thereof, are often used in systems where the channel memory is too long for optimal equalizers to be applied [4, 10].

It is clear that there exists a need for an equalizer that is able to optimally equalize signals in systems with long memory. Near-optimal low complexity MLSE equalization in systems with long memory is therefore a promising research topic, one that, if successful, will contribute greatly to digital communication technology. Such an equalizer is developed in this thesis.

[1] In practical systems the data block length is larger that the channel memory length.
[2] In spread spectrum systems, the chips have much shorter period than the information symbol.

1.2 Objectives

The objective of this thesis is to develop an MLSE equalizer, using the HNN as basis, able to equalize signals in systems with severe ISI beyond the capabilities of conventional optimal equalizers, while offering superior computational complexity.

1.2.1 M-QAM HNN MLSE Equalizer

The first objective of this study is to develop a general HNN MLSE equalizer model for systems using M-QAM modulation, able to equalize signals in systems transmitting N symbols through a channel with L resolvable multipath elements. This problem can be subdivided into three smaller problems. The equalizer must be able to equalize signals in systems

1. with extremely long memory,
2. using complex signal constellations and,
3. using multi-amplitude signal constellations.

If and only if these three problems are solved, will this objective be met.

1.2.2 HNN MLSE Turbo Equalization

The second objective is to use the HNN MLSE equalizer as a low complexity alternative to the MAP equalizer in a Turbo Equalizer configuration. Due to the soft-input soft-output (SISO) capability of the HNN MLSE equalizer, as well as its low computational complexity, it is an attractive candidate for use as a low-complexity replacement for the MAP equalizer in a Turbo Equalizer. Because of the excellent performance gains produced by using Turbo Equalization [11, 12, 13, 14], it is expected that the performance of the HNN MLSE equalizer can also be improved by using it in a Turbo Equalizer configuration. This will also allow for Turbo Equalization in systems with extremely long memory.

1.3 Contributions

The following research outputs have been produced during the course of this investigation:

1. "Near-Optimal Low Complexity MLSE Equalization" in the proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2008), Las Vegas, NV, USA, in [15].

2. "Low Complexity Iterative MLSE Equalization in Highly Spread Underwater Acoustic Channels" in the proceedings of IEEE Oceans 2009, Bremen, Germany, in [16].

3. "Low Complexity Iterative MLSE Equalization of M-QAM Signals in Extremely long Rayleigh Fading Channels" in the proceedings of IEEE Eurocon 2009, St. Petersburg, Russia, in [17].

4. "A Low Complexity Recurrent Neural Network MLSE Equaliser: Applications and Results" in the proceedings of South African Telecommunication, Networks and Applications Conference (SATNAC 2009), Mbabane, Swaziland, in [18].

Other research outputs produced during the course of this investigation but not directly related to it:

1. "Reduced Complexity Combined Soft-Decision MLSE Equalization and Decoding" in the proceedings of Australasian Telecommunication, Networks and Applications Conference (ATNAC 2008). Adelaide, Australia, in [19].

1.4 Structure of this Thesis

This thesis is structured as follows: *Chapter 2* gives a brief overview of artificial neural networks and introduces the reader to the HNN, used as the basis of the HNN MLSE equalizer, followed by a discussion on the various approaches to low complexity equalization. In *Chapter 3* all of the most important digital communication concepts, as considered in this thesis, are discussed, after which the HNN MLSE equalizer is derived and developed from first principles in *Chapter 4*. A number of applications of the HNN MLSE equalizer, as well as the simulation results thereof, are discussed and presented in *Chapter 5*. Conclusions are drawn in *Chapter 6*.

Chapter 2

Background

2.1 Artificial Neural Networks

An artificial neural network is a mathematical model that imitates the processing of information in the brain. Since 1861 the functioning of the human brain has been studied [20]. In 1929 the measuring of brain activity became possible with the invention of the electroencephalograph (EEG). The recent development of magnetic resonance imaging (MRI) provides neuroscientists with images of brain activity that corresponds with cognitive processes. These advancements were complemented by the study of single neuron activity, which entails the collection and processing of data on the finest level in the brain. A collection of these neurons, with each neuron being described by a mathematical model, is used to model the processing of the brain in the form of a neural network. Neural networks can be used to solve difficult scientific and engineering problems such as biometric identification, statistical prediction and signal processing [20].

A single neuron is activated by the collective severity of its inputs from other neurons. The level of each input is determined by the level of activation of the neuron that produced that signal as an output. All the inputs to a neuron are scaled by a certain weight, which is a function of the whole network, or at least the weights and activation levels of the neurons in the earlier stages of the network. Fig. 2.1 shows a single neuron [21]. It has n inputs and one output. The input is given by

$$u = \sum_{i=1}^{n} w_i x_i, \tag{2.1}$$

where x_i is the ith input and w_i is the ith corresponding input weight. The output of a single neuron is given by

$$y = g(u) = g\left(\sum_{i=1}^{n} w_i x_i\right). \tag{2.2}$$

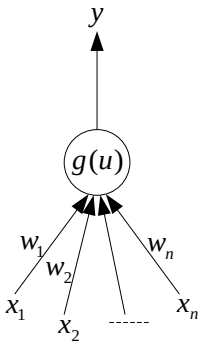

FIGURE 2.1: A single neuron.

In a discrete neural network, a neuron is either active or inactive, but in a continuous network, the level of activation is determined by a continuous function, limited by the discrete activation levels. The function $g(u)$ that governs the activation of a neuron is the *sgn*-function for a discrete network and the *tanh*-function[1] in the case of a continuous network. These functions are called activation-, decision-, or threshold functions [21].

There are two kinds of neural networks: *feed-forward* and *recurrent*.[2] A feed-forward network is the simplest kinds of neural network in that it only represents a function of its current inputs. The only useful output of these networks is the output of the final layer. Recurrent networks, on the other hand, feed the output of each neuron to the input of each other neuron in the network, since all the neurons in this network are connected. This enables the network to converge to a stable state after a number of iterations, provided that certain conditions are met [5], to find a near-optimal solution. The solution is read from the outputs of each of the neurons, which will settle in a stable state after a number of iterations. The recurrent neural network is a more accurate model of the brain, since it models associative memory. That is, the networks are able to draw conclusions based on partial information [20, 21].

Neural networks need to be *trained* to be of any use. During training, the input weights are adjusted in order to allow the neural network to respond appropriately. This can be done by either *supervised* or *unsupervised* learning. Supervised neural networks are given data in the form of inputs and desired outputs, where the desired outputs are a specification of the appropriate response of the neural network to the input. Unsupervised networks are given a set of training data - a set of examples of inputs and desired outputs. Each example is then used to adjust the input weights [21].

[1] The sigmoid function is also often used.
[2] Recurrent neural networks are also called feedback neural networks.

2.1.1 The Hopfield Neural Network

The Hopfield neural network (HNN) is a recurrent neural network and can be applied to optimization as well as pattern recognition problems, of which the former is of interest in this thesis. In 1985 Hopfield and Tank showed how neurobiological computations can be modeled with the use of an electronic circuit [5]. This circuit is shown in Fig. 2.2.

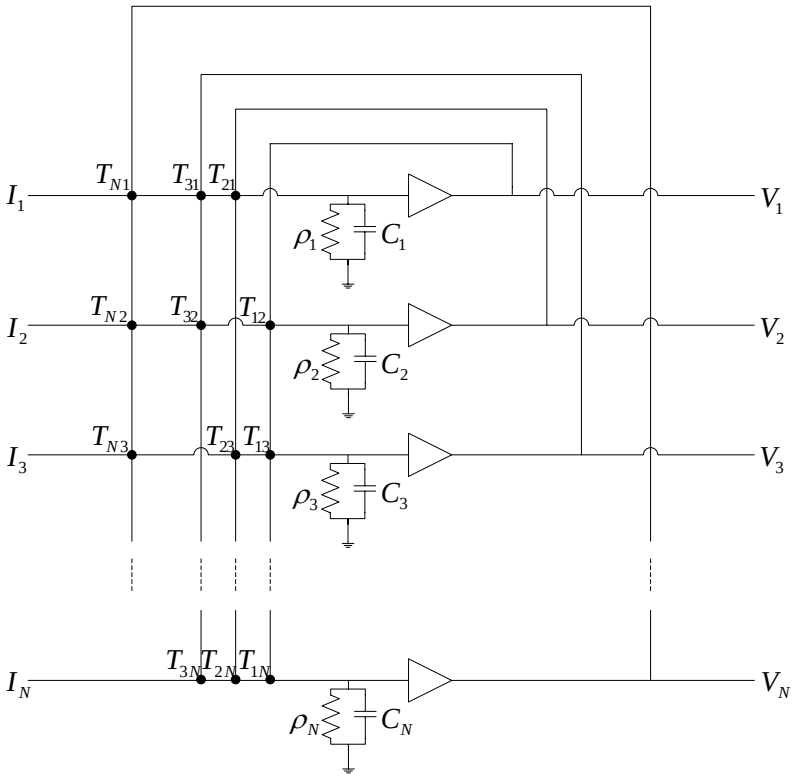

FIGURE 2.2: Hopfield network circuit diagram.

By using basic electronic components, they constructed a recurrent neural network and derived the characteristic equations for the network. The set of equations that describe the dynamics of the system is given by [5]

$$\begin{aligned} C_i \frac{du_i}{dt} &= -\frac{u_i}{\tau_i} + \sum_j T_{ij} I_i + I_i \\ V_i &= g(u_i) \end{aligned} \tag{2.3}$$

with T_{ij}, the dots, describing the interconnections between the amplifiers, u_1-u_N the input voltages of the amplifiers, V_1-V_N the output voltages of the amplifiers, C_1-C_N the capacitor values, ρ_1-ρ_N the resistivity values, and I_1-I_N the bias voltages of each amplifier. Each

amplifier represents a neuron. The transfer function of the positive outputs of the amplifiers represents the positive part of the activation function $g(u)$ and the transfer function of the negative outputs[3] represents the negative part of the activation function $g(u)$. The activation function is shown in Fig. 2.3.

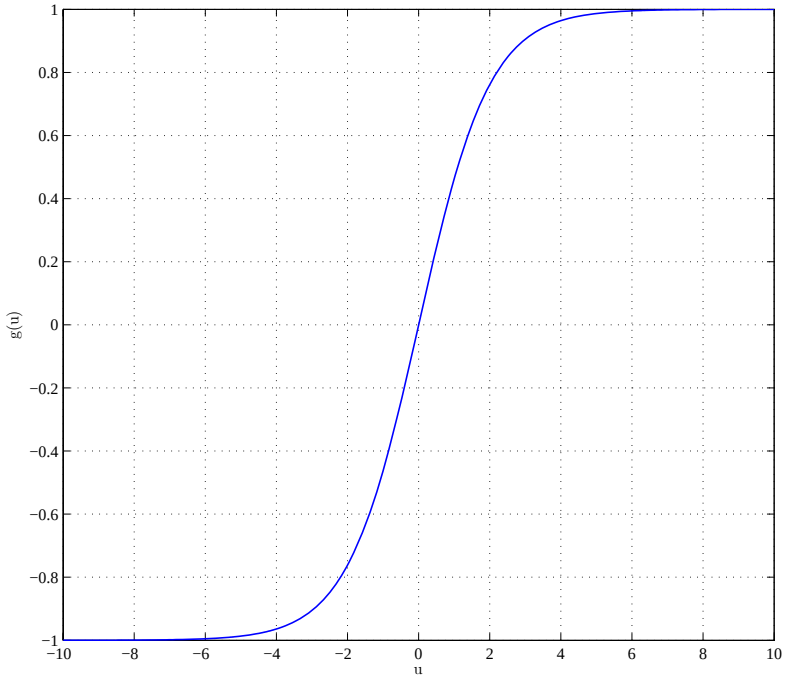

FIGURE 2.3: Activation function.

It was shown in [5] that the stable state of this circuit network can be found by minimizing the function

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} V_iI_i \tag{2.4}$$

provided that $T_{ij} = T_{ji}$ and $T_{ii} = 0$, implying that T is symmetric around the diagonal and its diagonal is zero [5]. There are therefore no self-connections. This function is called the energy function or the *Lyapunov* function, which by definition is a monotonically decreasing function, ensuring that the system will converge to a stable state [5]. When minimized, the network converges to a local minimum in the solution space to yield a *"good"* solution. The solution is not guaranteed to be optimal, but by using optimization techniques, the quality of the solution can be improved. To minimize (2.4) the system equations in (2.3) are solved iteratively until the outputs V_1-V_N settle.[4]

Hopfield also showed that this kind of network can be used to solve the Travelling Salesman Problem (TSP). This problem is of a class called NP-complete, the class of nondeterministic

[3]Negative outputs are not shown here.
[4]The iterative nature of the HNN will be descibed in great detail in *Section 4.3*.

polynomial problems. Problems that fall in this class can be solved optimally by enumerating each possible solution and choosing the best solution from all possible solutions [22]. Complete enumeration is a time-consuming and computationally expensive exercise, with the number of possible solutions growing exponentially with a linear increase in the number of unknowns. Complete enumeration is therefore not a feasible approach to solving real-time NP-complete problems, of which MLSE equalization is of concern in this thesis.

2.2 Low Complexity Equalization

2.2.1 MMSE Equalization

A Minimum Mean Square Error (MMSE) equalizer is often used to mitigate the effect of ISI, due to its relatively low computational complexity compared to other optimal equalizers. Its computational complexity is $O(N_l^2)$ to $O(N_l^3)$, where N_l is the filter length [10]. Using the MMSE criterion, that is, minimizing the mean square error between the estimated and transmitted symbol, an MMSE equalizer aims to determine a set of filter coefficients that will minimize the error between the estimated and transmitted symbols, before filtering the ISI-corrupted received symbols to produce the estimated sequence of symbols [4]. Due to its finite length, the MMSE cannot optimally reverse the effect of the channel on the transmitted symbols. Also, the MMSE equalizer is inadequate as a compensator for ISI on channels with spectral nulls, which are often encountered in wireless radio transmission [4]. Noise enhancement therefore limits the performance of MMSE equalizer, especially as the number of resolvable multipath elements grows. A complete description of MMSE equalization is presented in *Section 3.7.3*.

It was shown in [13] and [14] how an MMSE equalizer can be used in the place of an optimal MAP equalizer in a Turbo Equalization setup. Due to the high computational complexity of Turbo Equalization, there is a need for low complexity equalizers to enable Turbo Equalizers to equalize signals in systems with long memory.

2.2.2 Channel Shortening

One technique often used is to artificially shorten the estimated channel in order to perform equalization with lower complexity. Channel shortening is achieved by filtering the received symbols with an MMSE prefilter, in order to concentrate most of the signal energy in the first few taps of the CIR, so that the Viterbi MLSE equalizer or the MAP equalizer can be used for equalization using only the leading taps of the CIR [23, 24]. Since finite length filters are used, this technique also causes noise enhancement, ultimately limiting system performance.

2.2.3 DFE Equalization

Other techniques from the class of decision feedback equalizers (DFE) are also proposed, where the received symbols are filtered with an MMSE prefilter in order to produce a minimum phase CIR, after which symbol-by-symbol decisions are made and fed back to be used in subsequent decision stages [25]. The use of Delayed Decision Feedback Equalization (DDFE) is also proposed in [26], where the first few taps are equalized using a reduced state trellis, while a feedback loop is used to mitigate the ISI caused by the rest of the taps in the CIR. In general, the performance obtained through DFE techniques are limited due to error propagation, but DFEs are known to outperform their MMSE counterparts [4].

2.2.4 RAKE demodulator

In direct sequence spread spectrum (DSSS) systems the bandwidth is increased by spreading the source symbols with a spreading sequence. Depending on the spreading gain, there may be tens to hundreds of memory elements, rendering even the best opportunistic equalizers infeasible. The approach taken to equalize signals in a DSSS system is to use a RAKE demodulator [4, 10]. The RAKE demodulator uses fingers to select the highest energy taps in the CIR and coherently combines the energy in the selected taps in order to produce estimates of the transmitted chips [4]. The processing gain in a DSSS system compensates for the losses incurred during energy recombination, enabling transmission of multiple data streams on one frequency. A complete description of MMSE equalization is presented in *Section 3.7.4*

The performance of the RAKE demodulator is however limited due to the naive approach of coherent energy combination. It will be shown in *Chapter 5* that the proposed HNN MLSE equalizer can replace the RAKE demodulator in a spread spectrum system to yield better system performance.

DSSS systems are discussed in some detail in *Appendix A*.

2.2.5 OFDM Equalization

Due to the problem of severe ISI in wideband communication systems [4, 10, 27], multi-carrier (MC) modulation techniques have been proposed. The rationale behind using multi-carrier techniques is to increase the symbol period in order to alleviate the detrimental effects of ISI on the transmitted signal, facilitating trivial equalization complexity. Even when ISI cannot be eliminated completely, less ISI will ease the effort of equalization in each subcarrier.

There are however a few problems when MC-modulation is employed. If the symbol interval is increased, it necessitates a corresponding decrease in the carrier spacing in order to adhere to the data rate and bandwidth conditions and restrictions. Decreased carrier spacing will

render the system more susceptible to Doppler shifts, resulting in inter-carrier interference (ICI) which will inevitably lead to degraded system performance [10].

Orthogonal Frequency Division Multiplexing (OFDM) modulation can completely eliminate the effect of multipath on the system performance by exploiting the orthogonality properties of the Fourier matrix and through the use of a cyclic prefix, while maintaining trivial per symbol complexity. OFDM, however, is very susceptible to Doppler shift, suffers from a large peak-to-average power ratio (PAPR), and requires large overhead when the channel delay spread is very long compared to the symbol interval [4, 28].

2.2.6 HNN Equalization

It has been shown by various authors [29, 30, 31, 32] that the problem of MLSE can be solved using the HNN. It seems, however, that none of the authors applied the HNN MLSE model to systems with extremely long memory. Also, none of the authors attempted to develop an HNN MLSE equalizer for higher order signal constellations. Only BPSK and QPSK modulation are addressed, using short length static channels. The prospect of using a HNN MLSE equalizer in a Turbo Equalizer has also not been considered yet in other works. In this thesis MLSE equalization is performed in extremely long fading channels and the HNN MLSE equalizer developed herein is also applied to turbo equalization. Due credit is given where the ideas of other authors are used.

2.2.7 Turbo Equalization

Turbo Equalization has its origin in the *Turbo Principle*, first proposed in [33] where it was applied to the iterative decoding of concatenated convolutional codes. The results were astounding, with system performance almost reaching the all evasive *Shannon bound* [33]. The concept of *Turbo Coding* was applied to equalization in [11, 12] by viewing the channel as an outer code and replacing one of the convolutional MAP decoders with a MAP equalizer. The MAP equalizer and the MAP decoder iteratively exchange information. By iterating the system a number of times, the bit-error rate (BER) performance can be increased greatly, but at the cost of computational complexity and a substantial overhead due to error-correction coding.

It is however suggested in [13], and discussed in length in [14], that an MMSE equalizer can be modified to take advantage of prior information on the symbols to be estimated. By replacing the optimal MAP equalizer with a suboptimal, low complexity MMSE equalizer, low complexity turbo equalization is achieved, while still achieving matched filter bound performance using a static channel of length five [14]. It was also shown in [14] how a DFE equalizer can be used in a Turbo Equalizer configuration.

A soft-feedback equalizer (SFE) was also proposed in [6] with performance superior to that proposed in [14]. The author of [6] expanded upon ideas proposed in [14], where hard decisions on the equalizer output were fed back, by combining prior information with soft decisions [6]. The performance of the SFE Turbo Equalizer was evaluated for a MRC (9 taps), a MWC (44 taps) and a PLC (58 taps), outperforming the low complexity Turbo Equalizers proposed in [14], while doing so at reduced complexity compared to those proposed in [14].

The HNN MLSE equalizer developed in this thesis will also be applied to Turbo Equalization and compared to an MMSE-based turbo equalizer, as proposed in [14]. The concept of Turbo Equalization is discussed in length in *Section 3.7.5* and simulation results are presented in *Chapter 5*.

2.3 Concluding Remarks

In this chapter a brief overview on artificial neural networks was given, followed by a description of the HNN used as the foundation of the HNN MLSE equalizer developed in this thesis. A number of low complexity equalization techniques, together with their respective drawbacks, were discussed, serving as motivation for the development of a low complexity MLSE equalizer for use in extremely long channels.

Chapter 3

Digital Communication System Concepts

In this chapter the fundamental building blocks of a wireless communcation system, such as will be used as a platform of evaluation in this thesis, are discussed. Where applicable, models used as functional blocks for simulation are also discussed.

3.1 Overview

Fig. 3.1 shows the communication system considered in this thesis. The source bits $\mathbf{z}$ are encoded with a convolutional encoder of rate R_c to produce a sequence of N coded bits $\mathbf{u}$. The coded bits are mapped symbols s' according to the constellation map specified by the modulation scheme. The symbols are chosen from an alphabet $\mathcal{D}$ containing M complex symbols. The coded symbols are interleaved ($\prod$) so as to randomize the occurrence of errors during transmission. The interleaver produces a sequence of symbols $\mathbf{s}$ of which each individual symbol is transmitted through a multipath channel with impulse response $\mathbf{h}$, where the channel acts like a filter on the transmitted symbols.

The received symbols $\mathbf{r}$ are corrupted by intersymbol interference (ISI), due to the multipath channel and independent additive white Gaussian noise (AWGN) samples $\mathbf{n}$ from a zero mean Gaussian distribution with variance σ^2. The resulting symbol sequence $\mathbf{r}$ at the receiver is therefore the result of the convolution of the transmitted sequence ($\mathbf{s}$) with the channel ($\mathbf{h}$), plus added noise.

The channel estimator uses the known pilot symbols $\mathbf{s}_{(\mathbf{p\_start}:\mathbf{p\_end})}$ together with the received symbols $\mathbf{r}$ to produce an estimate $\hat{\mathbf{h}}$ of the CIR. The equalizer uses the estimated CIR ($\hat{\mathbf{h}}$) together with the received symbols ($\mathbf{r}$) to produce estimates $\hat{\mathbf{s}}$ of the transmitted symbol sequence. The transmitted symbol estimates ($\hat{\mathbf{s}}$) are deinterleaved ($\prod^{-1}$) to reverse the effect

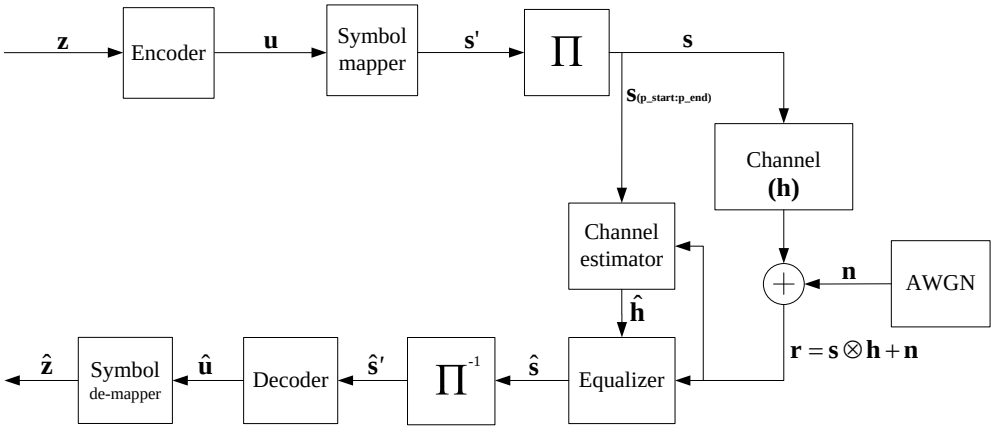

FIGURE 3.1: Communication System.

the interleaver had on the coded symbols in the transmitter and produces $\hat{\mathbf{s}}'$, which is then used as input to the decoder.

The decoder takes $\hat{\mathbf{s}}'$ as input and exploits the redundant information in the estimated coded symbol sequence to correct errors, and produces an estimate $\hat{\mathbf{u}}$ of the source symbols ($\mathbf{u}$) with maximum confidence. The source symbol estimates ($\hat{\mathbf{u}}$) are then demapped to produce estimates $\hat{\mathbf{z}}$ of the source bits ($\mathbf{z}$).

It must be noted that some components of a digital communication system are not considered, as it is not necessary to evaluate the equalizer developed in this thesis. As such, *pulse shaping*, *modulation*, *demodulation* and *matched filtering* are not considered here, as it is assumed that these processes are performed perfectly. It is assumed that the received symbols ($\mathbf{r}$) are produced by the matched filter with maximum SNR. All simulations are performed at baseband.

Each of the subsystems in this communication system is discussed in turn in what follows.

3.2 Symbol Mapping and De-mapping

In order to transmit information in a communication system, a carrier frequency must be modulated by a representation of the source data. To perform this task, a symbol constellation map is used to transform the source information from bits to modulation symbols. These modulation symbols are then used to modulate the carrier frequency in order to convey information. In this thesis three modulation schemes are considered, namely binary phase-shift keying (BPSK), quaternary quadrature amplitude modulation (4-QAM) and 16-QAM, each of which uses a unique constellation map. The number of bits that can be transmitted during symbol period T_s using a constellation map with M symbols, is $log_2(M)$ [4].

Figures 3.2 to 3.4 show the normalized[1] Gray coded[2]constellation maps using for BPSK, 4-QAM and 16-QAM modulation, respectively, with their respective mappings shown in tables 3.1 to 3.3. The source bits are mapped to modulation symbols according the constellation map. For BPSK, one information bit is transmitted during one symbol interval T_s. For 4-QAM, two information bits are transmitted during one symbol interval, and for 16-QAM, four information bits are transmitted during one symbol interval. Once the symbols are estimated at the receiver, the constellation map is again used to map the symbols back to bits in order to produce estimates of the transmitted source information.

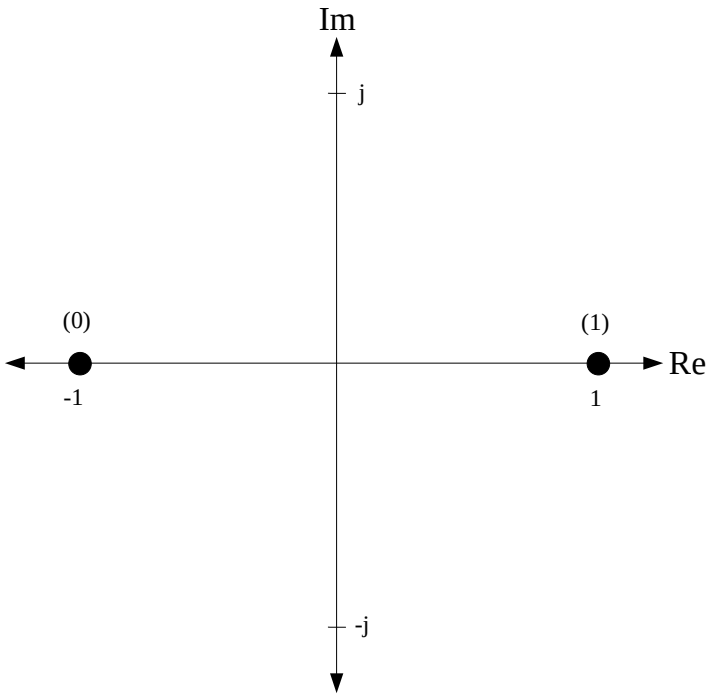

FIGURE 3.2: BPSK constellation map.

TABLE 3.1: BPSK symbol mapping.

Bits	**Symbol (Rectangular)**	**Symbol (Polar)**
1	1	$\angle 0$
0	-1	$\angle \pi$

[1]The constellation maps are normalized such that the Euclidean distance from the orgin to the farthest symbols is equal to 1. During simulation, a scaling factor is used to normalize the noise so that the average signal energy is equal to 1.

[2]Gray coding is employed so that each tuple of bits from neighboring symbols only differ with one bit. This will increase the probability of one bit error when a symbol error is made during detection [4].

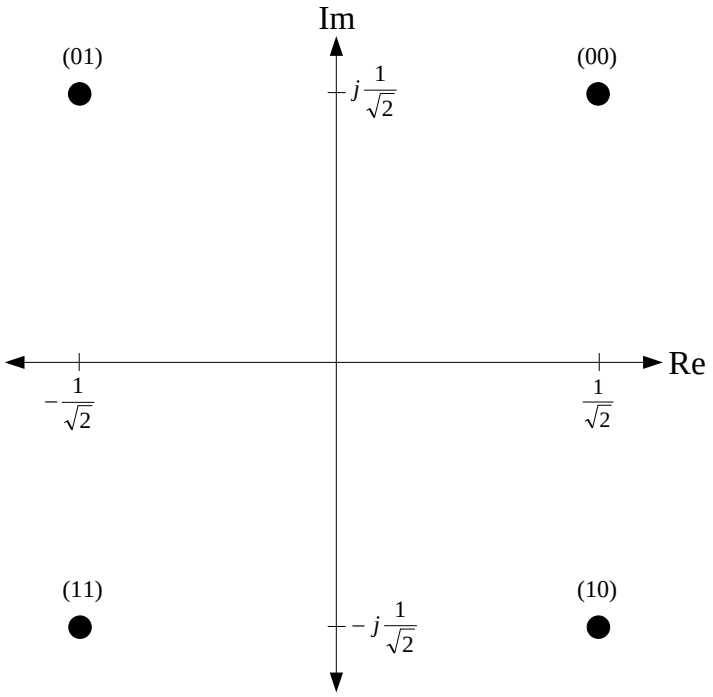

FIGURE 3.3: 4-QAM constellation map.

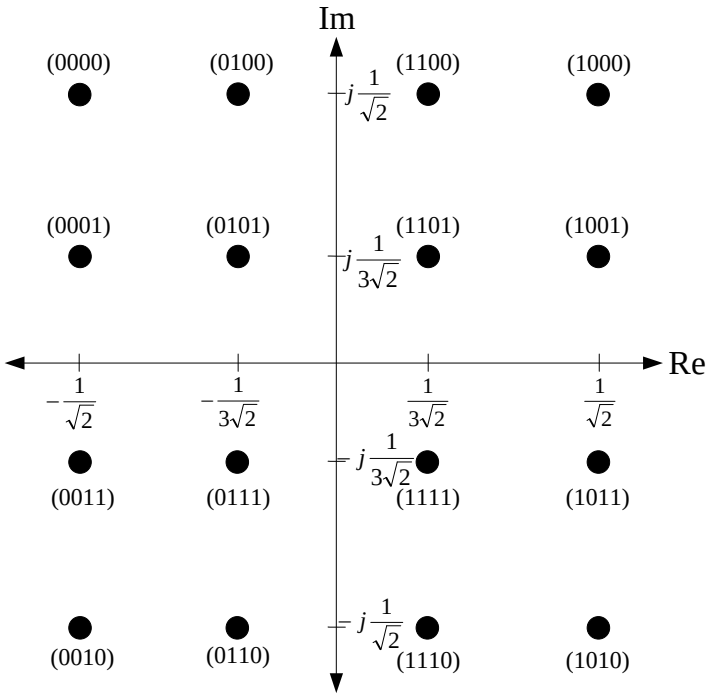

FIGURE 3.4: 16-QAM constellation map.

TABLE 3.2: 4-QAM symbol mapping.

Bits	Symbol (Rectangular)	Symbol (Polar)
00	$\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$	$\angle\frac{\pi}{4}$
01	$-\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$	$\angle\frac{3\pi}{4}$
10	$\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$	$\angle\frac{-3\pi}{4}$
11	$-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$	$\angle\frac{-\pi}{4}$

TABLE 3.3: 16-QAM symbol mapping.

Bits	Symbol (Rectangular)	Symbol (Polar)
0000	$-\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$	$\angle\frac{3\pi}{4}$
0001	$-\frac{1}{\sqrt{2}} + j\frac{1}{3\sqrt{2}}$	$\frac{1}{3}\sqrt{5}\angle(\pi - 0.3218)$
0010	$-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$	$\angle\frac{-3\pi}{4}$
0011	$-\frac{1}{\sqrt{2}} - j\frac{1}{3\sqrt{2}}$	$\frac{1}{3}\sqrt{5}\angle(\pi + 0.3218)$
0100	$-\frac{1}{3\sqrt{2}} + j\frac{1}{\sqrt{2}}$	$\frac{1}{3}\sqrt{5}\angle(\frac{\pi}{2} + 0.3218)$
0101	$-\frac{1}{3\sqrt{2}} + j\frac{1}{3\sqrt{2}}$	$\frac{1}{3}\angle\frac{3\pi}{4}$
0110	$-\frac{1}{3\sqrt{2}} - j\frac{1}{\sqrt{2}}$	$\frac{1}{3}\sqrt{5}\angle(-\pi - 0.3218)$
0111	$-\frac{1}{3\sqrt{2}} - j\frac{1}{3\sqrt{2}}$	$\frac{1}{3}\angle\frac{-3\pi}{4}$
1000	$\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$	$\angle\frac{\pi}{4}$
1001	$\frac{1}{\sqrt{2}} + j\frac{1}{3\sqrt{2}}$	$\frac{1}{3}\sqrt{5}\angle(0.3218)$
1010	$\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$	$\angle\frac{-\pi}{4}$
1011	$\frac{1}{\sqrt{2}} - j\frac{1}{3\sqrt{2}}$	$\frac{1}{3}\sqrt{5}\angle(-0.3218)$
1100	$\frac{1}{3\sqrt{2}} + j\frac{1}{\sqrt{2}}$	$\frac{1}{3}\sqrt{5}\angle(\frac{\pi}{2} - 0.3218)$
1101	$\frac{1}{3\sqrt{2}} + j\frac{1}{3\sqrt{2}}$	$\frac{1}{3}\angle\frac{\pi}{4}$
1110	$\frac{1}{3\sqrt{2}} - j\frac{1}{\sqrt{2}}$	$\frac{1}{3}\sqrt{5}\angle(-\pi + 0.3218)$
1111	$\frac{1}{3\sqrt{2}} - j\frac{1}{3\sqrt{2}}$	$\frac{1}{3}\angle\frac{-\pi}{4}$

3.2.1 Random Number Generator

A random number generator is used to generate uniformly distributed real numbers $I \in [0; 1]$ to be used for the generation of random source bits in the receiver. The Wichman-Hill algorithm [34] is used to generate uniformly distributed numbers with probability density function (PDF)

$$p(x) = \frac{1}{b-a}, \qquad a \leq x \leq b, \tag{3.1}$$

as depicted in Fig. 3.5 for $a = 0$ and $b = 1$. Fig. 3.6 shows the simulated PDF using 1280000 samples generated with the Wichman-Hill algorihtm. A 1-bit is generated if $I \geq 0.5$, and a 0-bit is generated otherwise.

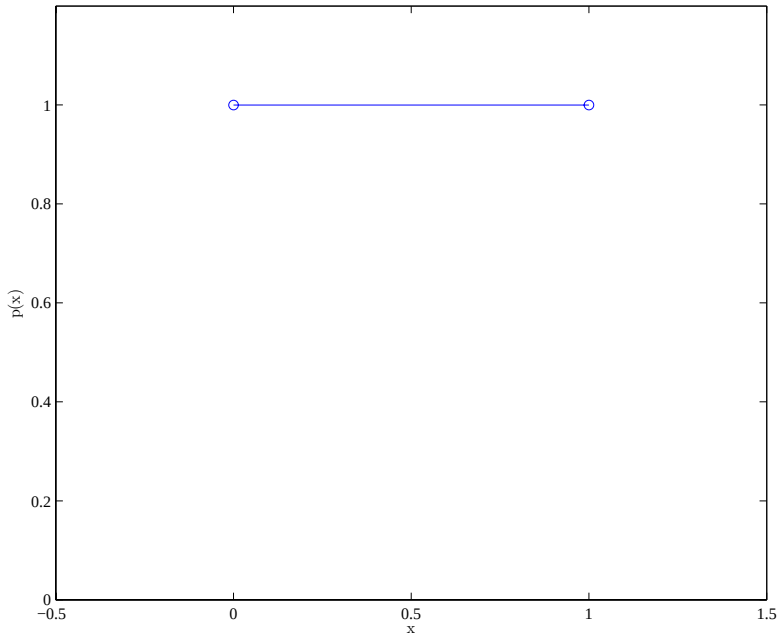

FIGURE 3.5: Theoretical uniform PDF for $a = 0$ and $b = 1$.

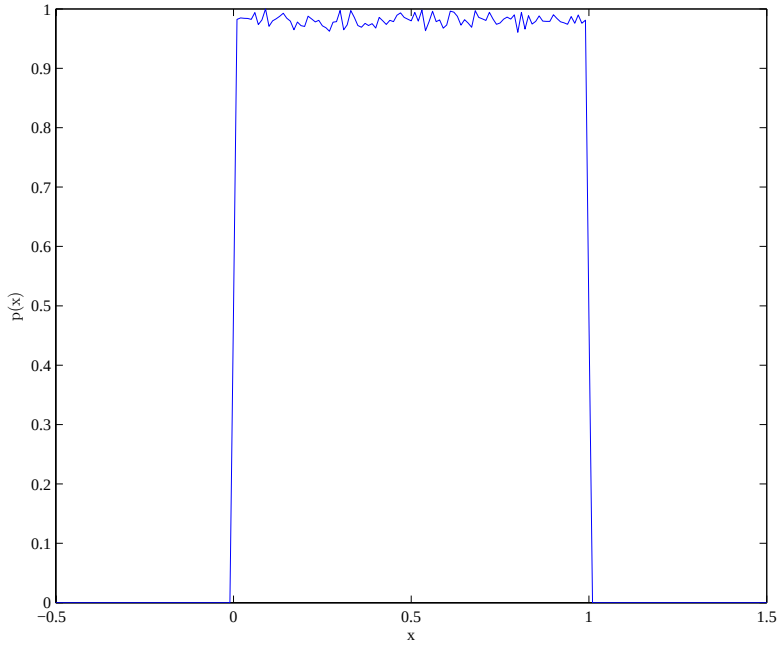

FIGURE 3.6: Simulated uniform PDF.

3.3 Error-Correction Coding

Error-correction coding (ECC), also referred to as error control coding, plays an important role in digital communication systems. As the name suggests, error-correction coding is used to allow for the correction of errors in the received symbol sequence. EEC is performed by adding redundant information, in a controlled fashion, to the information being transmitted. Since the redundant information is correlated with the original information, errors can be corrected [4]. Although the redundancy adds an overhead to the transmitted data, the performance increase overshadows this drawback. During the last few decades, ECC has been subjected to much research and significant contributions and advancements have been made, with some authors reporting to have achieved near *Shannon bound* performance [21, 33].

3.3.1 Convolutional Codes

The coding scheme considered in this thesis is convolutional encoding. A convolutional code is generated by passing the information sequence to be transmitted through a linear finite state shift register, consisting of K stages.[3] The binary input is shifted into the shift register k bits at a time, producing n output bits. The code rate, which was mentioned before, is $R_c = k/n$. The encoder can be expressed in terms of n generator polynomials, describing the connections between the states of the encoder shift register.

Fig. 3.7 shows the rate $R_c = k/n = 1/3$, constraint length $K = 3$, convolutional encoder considered in this thesis.

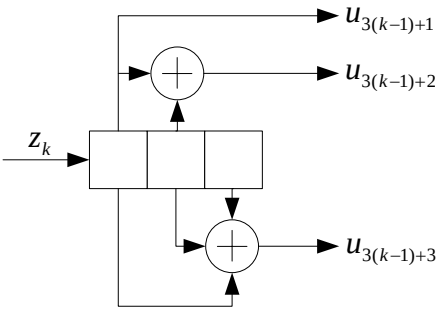

FIGURE 3.7: Rate 1/3 convolutional encoder.

[3] K is also known as the constraint length.

The generator polynomials that describe the connections between the elements in the shift register are given in vector form as

$$\begin{aligned} \mathbf{g}_1 &= [100] \\ \mathbf{g}_2 &= [110] \\ \mathbf{g}_3 &= [111]. \end{aligned} \tag{3.2}$$

For every input source bit z_k the encoder produces three output bits $u_{3(k-1)+1}$, $u_{3(k-1)+2}$ and $u_{3(k-1)+3}$, where

$$\begin{aligned} u_{3(k-1)+1} &= z_k \\ u_{3(k-1)+2} &= z_k \oplus z_{(k-1)} \\ u_{3(k-1)+3} &= z_k \oplus z_{(k-1)} \oplus z_{(k-2)}, \end{aligned} \tag{3.3}$$

where $\oplus$ is the exclusive OR operation. It is assumed that the encoder starts in the all-zero state for every data block that is encoded. Also, the encoder is forced into a zero-state after the data block is encoded by appending $1/R_c = 3$ zero bits to the data block. This is done to enable decoding using a VA or MAP decoder [2, 3].

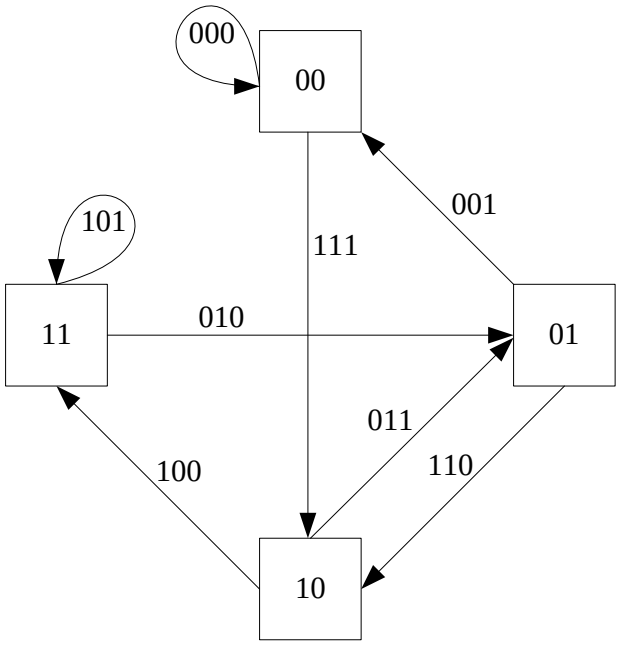

FIGURE 3.8: Rate 1/3 convolutional encoder state diagram.

Each convolution encoder has a corresponding state diagram to describe the output in terms of the state transitions, which is used to determine the most likely state transitions during decoding. The state diagram of the encoder in Fig. 3.7 is shown in Fig. 3.8. Each state

contains two bits representing the two leftmost elements in the encoder. As bits are shifted through the encoder, transitions occur, with each state transition producing three output bits. This should be clear from Fig. 3.8.

To determine the minimum free distance of this code, the simplified method proposed in [35] was used. The minimum free distance was found to be $d_{free} = 6$. Therefore, the coding gain achievable with this code is upper-bounded by $10log(R_c d_{free})$ [4]. Therefore, the coding gain achievable with this code is less than or equal to 3.01 dB.

3.4 Interleaving

Interleavers are used to randomize the noise introduced to the transmitted information during transmission and reception. In a mobile communication environment, where the channel is characterized by multipath and fading, burst errors are common, which will cause a stream of adjacent symbols to be damaged beyond recognition. Signal fading due to time-invariant multipath propagation often causes the signal to fall below the noise level, resulting in a large number of consecutive errors [4]. This will inhibit the decoder from correcting errors, since the decoder relies on the structure in the coded information as introduced by the encoder.

When the coded symbols are interleaved before transmission, burst errors will be transformed into independent errors. An interleaver reorders the symbols before transmission so that the errors will be decorrelated after deinterleaving in the receiver. Interleaving aids the decoder in that the structure introduced by the encoder to the transmitted symbols can be exploited to the full, which will yield performance gains.

A number of interleavers are widely used. These are convolutional-, block-, and S-random interleavers [4]. The interleaver considered in this thesis is of the class of random interleavers, without the restriction imposed on S-random interleavers [36].[4] Fig. 3.9 shows a random interleaver of length $N = 20$.

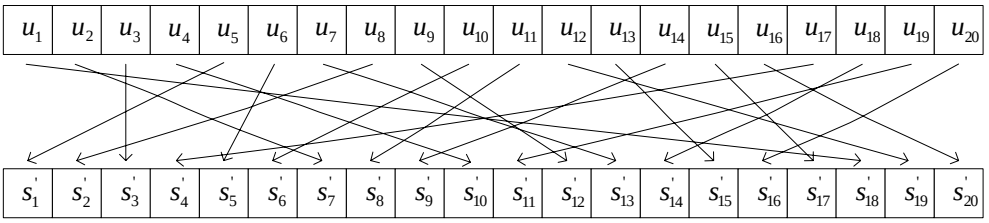

FIGURE 3.9: Random interleaver of length $N = 20$.

[4]For S-random interleavers, each pair in the group of S consecutive bits must be at least S indices apart after interleaving.

The connections between the u_k nodes and $s_k^{'}$ are determined by selecting a number at random from a list containing the numbers 1 to 20. After each selection, the selected number is removed from the list and appended to a new array. This process continues until the list is empty. Each element in the array contains the new position of the symbol corresponding to that element. The coded symbols $\mathbf{u}$ are reordered according to the indices in the new array to produce $\mathbf{s}'$, as indicated in Fig. 3.9.

To reverse the process of interleaving, deinterleaving is performed. Deinterleaving maps the interleaved symbols back to their original positions, as dictated by the interleaver.

3.5 The Channel

A mobile communication system transmission channel is characterized by multipath and fading. Multipath is the phenomenon resulting from time spreading of the transmitted signal as it is transmitted through the channel. Fading, on the other hand, results from time variations in the structure of the transmission medium, causing the nature of the multipath channels to vary with time [4]. During transmission, the transmitted symbols pass through the channel which acts like a filter. The channel has a continuous impulse response which is estimated at the reciever to aid in the estimation of the transmitted information.

3.5.1 Multipath

Each coefficient, or tap, in the impulse response of a multipath fading channel is modeled as a continuous function of time, where each coefficient in the impulse response corresponds to symbol period intervals kT_s. As such, a tapped delay line is used to model the behavior of this channel, as shown in Fig. 3.10.

Fig. 3.10 indicates that the kth transmitted symbol s_k is delayed by T_s seconds, $L-1$ times, where L is the CIR length. Each delayed copy of s_k is multiplied by $h_k^{(l)}$, $l = 1, 2, ..., L-1$, corresponding to the lth delay branch at time k. Therefore, the kth received symbol can be described by [1, 4]

$$r_k = \sum_{l=0}^{L-1} h_k^{(l)} s_{k-l} + n_k, \tag{3.4}$$

where n_k is the kth Gaussian noise sample $\mathcal{N}(0, \sigma^2)$. Each $h_k^{(l)}$ is a sample taken from one of L time-varying functions, where k corresponds with the kth transmitted symbol and $l = 1, 2, ..., L-1$ is the CIR tap number. Each CIR tap is modeled as an independent uncorrelated Rayleigh fading vector, describing the fading behaviour of that tap.

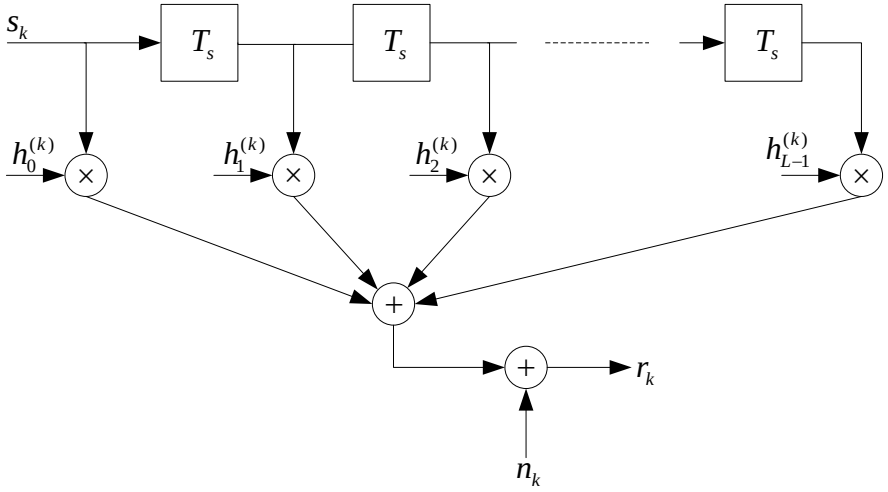

FIGURE 3.10: Tapped delay line for a multipath fading channel.

If it is assumed that the CIR is time-invariant for the duration of a data block, (3.4) can be rewritten as

$$r_k = \sum_{l=0}^{L-1} h_l s_{k-l} + n_k, \tag{3.5}$$

where each CIR coefficient h_l is assumed to be time-invariant for a given time interval. The CIR is assumed static, or time-invariant, for the duration of a data block.

3.5.2 Fading

It was stated that each coefficient, or tap, in the impulse response of a multipath fading channel is modeled as a continuous function of time. Each tap, then, can be modeled as a flat fading channel. If a signal with bandwidth $B << B_c$ is transmitted, the fading across the entire signal bandwidth is highly correlated, which is referred to as *flat* fading. Here B_c is the coherence bandwith of the system, defined as the minimum frequency separation for which the CIR is independent [10]. On the other hand, if the bandwidth $B >> B_c$, then the channel amplitude values at frequencies separated by more than the coherence bandwidth are roughly independent, which is referred to as frequency-selective [10]. In flat fading channels there is no ISI and in frequence-selective channels there is ISI.

When channel fading is considered, distinction needs to be made between Rayleigh fading and Rician fading. When it is assumed that there is no line of sight (LOS) between the transmitter and the receiver, Rayleigh fading is used, and Rician fading is used when there is LOS between the transmitter and the receiver [4]. The former is considered in this thesis.

For a Rayleigh fading channel the fading amplitude at instance k is given by

$$R_k = \sqrt{X_k^2 + Y_k^2}, \tag{3.6}$$

where X_k and Y_k are samples taken from a Gaussian random process $\mathcal{N}(0, \sigma_r^2)$ [4]. The probability density function of a Rayleigh random variable is given by

$$p(x) = \frac{x}{\sigma_r^2} exp\left(\frac{-x^2}{2\sigma_r^2}\right), \qquad x \geq 0, \tag{3.7}$$

as shown in Fig. 3.11. The simulation of uncorrelated Rayleigh fading vectors is discussed next.

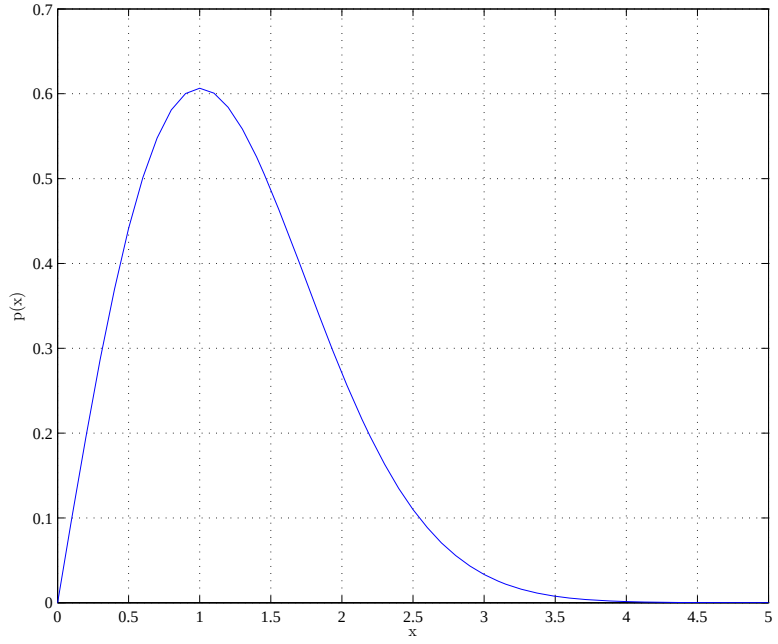

FIGURE 3.11: Rayleigh random process PDF for $\mu = 0$ and $\sigma_r^2 = 1$.

3.5.2.1 Generating Uncorrelated Fading Vectors

The model proposed in [37] is used for the generation of uncorrelated fading vectors. This model is based on a sum-of-sinusoids approach, which is an extension off Clarke's frequency nonselective fading process model [38]. Clarke's model is given by

$$g(t) = \sqrt{\frac{2}{N}} exp\left(\omega_d t cos\alpha_n + \phi_n\right) \tag{3.8}$$

where α_n is the angle of the incoming wave, ϕ_n is the initial phase associated with the nth propagation that, and ω_d is the maximum angular Doppler frequency when $\alpha_n = 0$.

The model in (3.8) is modified such that $N = 4M$, $\phi_{n+M} = -\varphi + \frac{\pi}{2}$, $\phi_{n+2M} = -\phi_n$, $\phi_{n+3M} = \varphi_n + \frac{\pi}{2}$, and $\alpha_n = (2 - \pi + \theta)/N$, where θ is uniformly distributed in $[-\pi; \pi)$ [37]. The complex low-pass Rayleigh fading process in (3.8) is therefore described by

$$g(t) = \sqrt{\frac{2}{N}} \left(\sum_{n=1}^{M} 2cos(w_d t cos\alpha_n + \phi_n) + j \sum_{n=1}^{M} 2cos(w_d t sin\alpha_n + \varphi_n) \right). \tag{3.9}$$

of which the PDF of the magnitude will approximate a Rayleigh distribution [37]. To produce a new set of uncorrelated fading vectors, one for the real symbol components and one for the imaginary symbol components, the normalized low-pass fading process of a new statistical sum-of-sinusoids simulation model is defined by [37]

$$\begin{aligned} Z(t) &= Z_i(t) + jZ_q(t) \\ Z_i(t) &= \sqrt{\frac{2}{N}} \left(\sum_{n=1}^{M} cos(w_d t cos\alpha_n + \phi_n) \right) \\ Z_q(t) &= \sqrt{\frac{2}{N}} \left(\sum_{n=1}^{M} cos(w_d t sin\alpha_n + \varphi_n) \right), \end{aligned}$$

where

$$\alpha_n = \frac{2 - \pi + \theta}{4M}$$

and ϕ_n, φ_n and θ are statistically independent and uniformly distributed in $[-\pi; \pi)$ for all n. Choosing $M = 10$, this model is used to generate a fading waveform of $10,000,000$ samples with $f_D = 500$ Hz, carrier frequency $f_c = 900$ MHz and symbol period $T_s = 3.7\mu$s. The resulting PDF is shown in Fig. 3.12 where it is compared to the theoretical PDF in Fig. 3.11. Fig. 3.13 shows the amplitudes of four uncorrelated fading vectors for a data block length of $N = 500$, Doppler frequency $f_D = 400$ Hz $f_c = 900$ Mhz, and the symbol period $T_s = 3.7$ μs. AGWN is applied to the ISI-corrupted received symbols.

3.5.3 White Gaussain Noise

In addition to ISI, the transmitted symbols are also corrupted by AWGN, mainly caused by electron fluctuations in electronic components and amplifiers in the receiver [4]. This type of noise is characterized as a zero-mean Gaussian noise process with variance σ_n^2. The Gaussian process PDF is given by

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{|x - \mu|^2}{2\sigma_n^2} \right), \tag{3.11}$$

where μ is the mean and σ_n^2 is the variance. The PDF of a zero mean, unity variance, Gaussian random process is shown in Fig. 3.14.

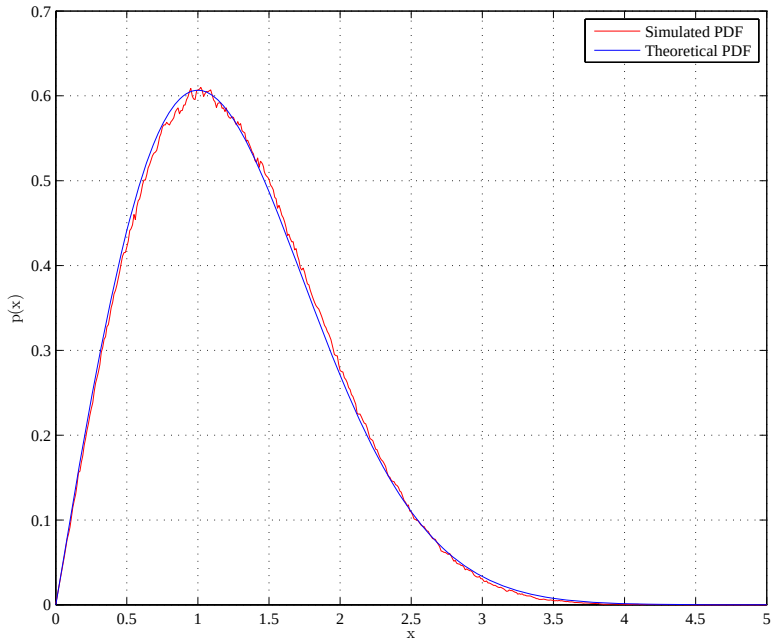

FIGURE 3.12: Simulated Rayleigh PDF compared to the theoretical PDF.

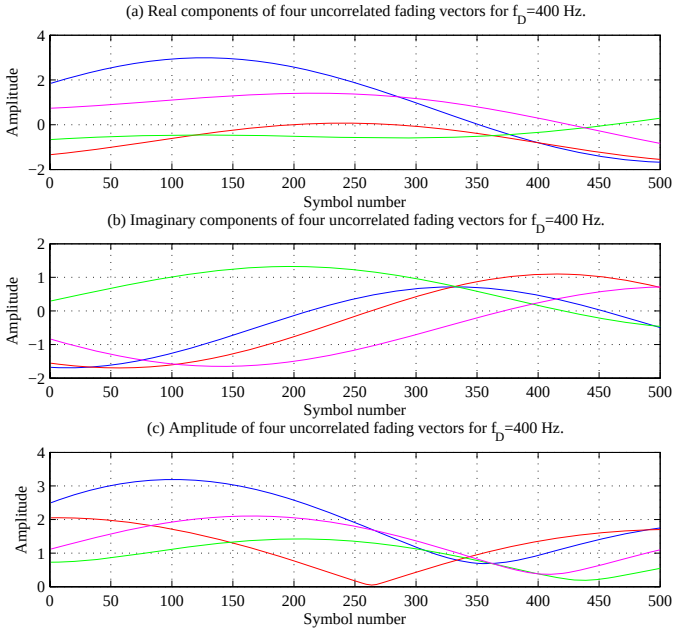

FIGURE 3.13: Four uncorrelated fading vectors for $f_D = 500$ Hz. (a) Real components (b) Imaginary components (c) Magnitude.

The assumption of AWGN is important for designing equalizers for digital communication systems. This will become apparent in *Section 3.7*. The simulation of Gaussian noise, and

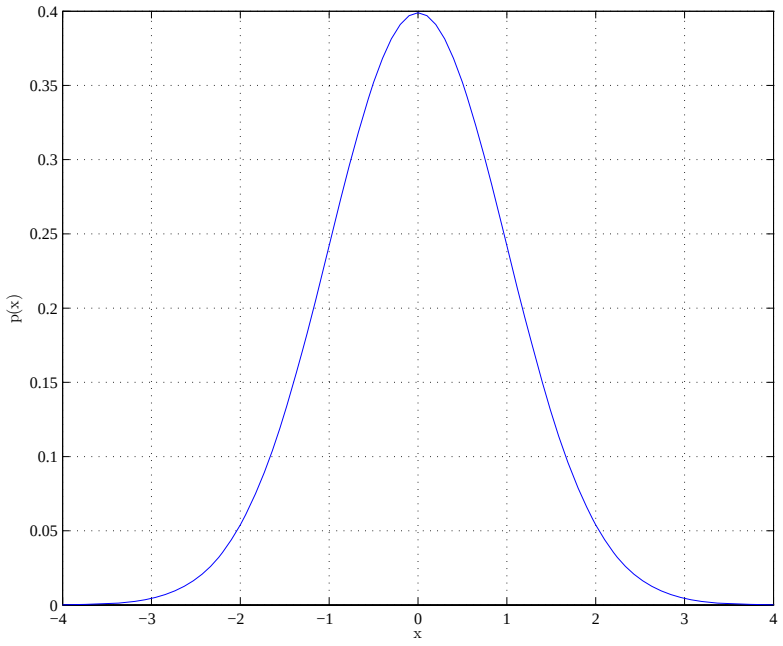

FIGURE 3.14: Gaussian random process PDF for $\mu = 0$ and $\sigma_n^2 = 1$.

how it is applied to the received symbols, is discussed next.

3.5.3.1 AWGN Simulation

To generate AWGN for use in the simulation of a communication system, the Marsaglia-Bray algorithm proposed in [39] is used. Fig. 3.14 shows the PDF for a zero-mean unit variance Gaussian random variable described by (3.11). Generating 1280000 zero-mean unit variance Gaussian noise samples with the Marsaglia-Bray algorithm produces the PDF in Fig. 3.15, which corresponds to the theoretical PDF in Fig. 3.14.

3.5.3.2 Applying Gaussian noise to received symbols

In order to evaluate the performance of digital communication systems, it is useful to evaluate the systems for a given signal-to-noise ratio (SNR), expressed as

$$SNR = \frac{\sigma_s^2}{\sigma_n^2}, \tag{3.12}$$

where σ_s^2 is the average signal power and σ_n^2 is the average noise power. Since the SNR is meaningless unless the noise equivalent bandwidth is specified [40], it is instructive to use a normalized measure of the SNR, namely the energy per bit to noise spectral density ratio E_b/N_0. Following the derivation in [40], the value by which the Gaussian random number

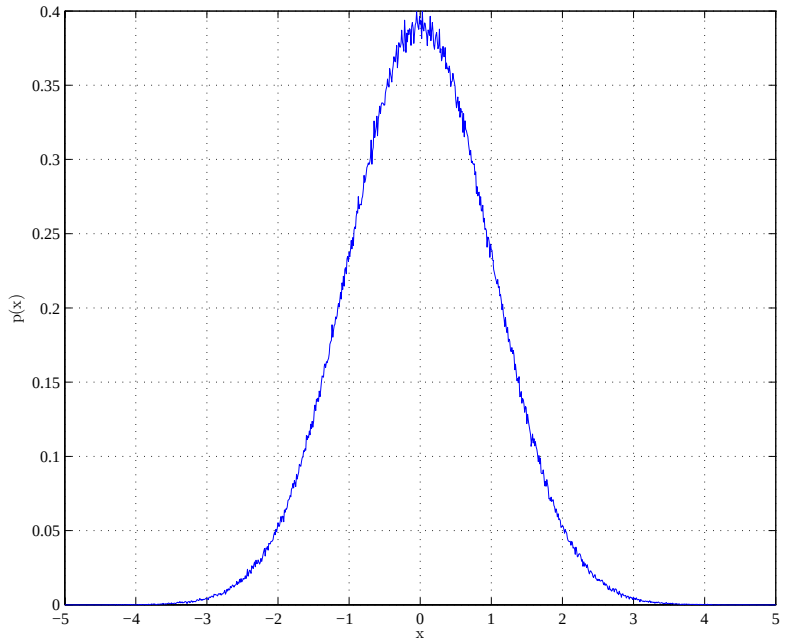

FIGURE 3.15: Simulated Gaussian PDF for $\mu = 0$ and $\sigma^2 = 1$.

generator output is scaled, is expressed as

$$k_\eta = \sqrt{\frac{\sigma_s^2 . f_{samp}}{2 . f_{bit} . 10^{\frac{E_b}{10 . N_0}}}}, \tag{3.13}$$

where f_{samp} is the frequency at which Gaussian noise samples are generated, f_{bit} is the frequency of uncoded bits per transmitted symbol and E_b/N_0 is the signal strength (in dB) where the systems is evaluated. By scaling the output of the zero mean, unit variance Gaussian noise generator by k_η, Guassian noise samples with variance σ_n^2 are produced, which are then added to the ISI-corrupted symbols in the receiver.

3.5.4 Doppler Frequency

Doppler frequency shifts are encountered when there is a change in the relative velocity between the transmitter and the receiver. If the relative velocity increases, the Doppler frequency is positive and when the relative velocity decreases, the Doppler frequency is negative. As the Doppler frequency increases, there is a corresponding increase in the fading rate. The Doppler frequency is determined by [10]

$$f_D = \frac{v f_c}{c} \theta_D, \tag{3.14}$$

where v is the relative velocity between the transmitter and the receiver, f_c is the carrier frequency. θ_D is the incident angle of the received signal and c is the transmission speed in the medium, which is commonly assumed to be equal to the speed of light, $c = 3 \times 10^8$m/s for wireless communication through the atmosphere. A number of parameters are associated with the Doppler frequency [10]:

- The Doppler spread B_D is a measure of the expansion of the spectral band of the signal being transmitted and is given by $B_D \approx 1/T_c$, where T_c is the coherence time.
- The coherence time T_c specifies the period during which the CIR remains time-invariant and is approximated as $T_c \approx 1/B_D$.
- The coherence bandwidth B_c is the minimum separation of frequencies for which the CIR is independent and is given by $B_c \approx 1/\tau_{max}$, where τ_{max} is the channel delay spread, i.e. the duration between the first and the last arrival of the same signal component.

Fig. 3.16 shows the real components for relative mobile velocities of 3 km/h, 50 km/h, 120 km/h and 250 km/h, where the incident angle is ignored, corresponding to Doppler frequencies of 2.5 Hz, 41.67 Hz, 100 Hz and 208.33 Hz, where the data block length is $N = 500$, the carrrier frequency $f_c = 900$MHz, and the symbol period $T_s = 3.7$ μs. It is clear that as the relative mobile velocity, and therefore the Doppler frequency, increases, the fading rate increases correspondingly.

3.5.5 Power Delay Profiles

The power delay profile (PDP) represents the average power associated with a given multipath delay [10]. The power delay profile is a continuous function of time, but in the receiver the channel delay profile is segmented according to the number of resolvable multipath elements. Two multipath components are said to be resolvable if the difference between their respective delays is less than the inverse of the signal bandwidth [10]. In other words, since the symbol period

$$T_s = \frac{1+\alpha}{B}, \tag{3.15}$$

where α is the roll-off factor[5] and B is the signal bandwidth, two multipath components are resolvable if the difference between their respective delays is equal to or greater than the symbol period, assuming that $\alpha = 0$. When the multipath elements are not resolvable, these nonresolvable components are combined into a single multipath component with delay approximately equal to the delays of those elements, with amplitude and phase corresponding to the sum of the different components [10].

[5]The roll-off factor is a parameter of the square-root Nyquist filter which determines the cut-off rate of the filter.

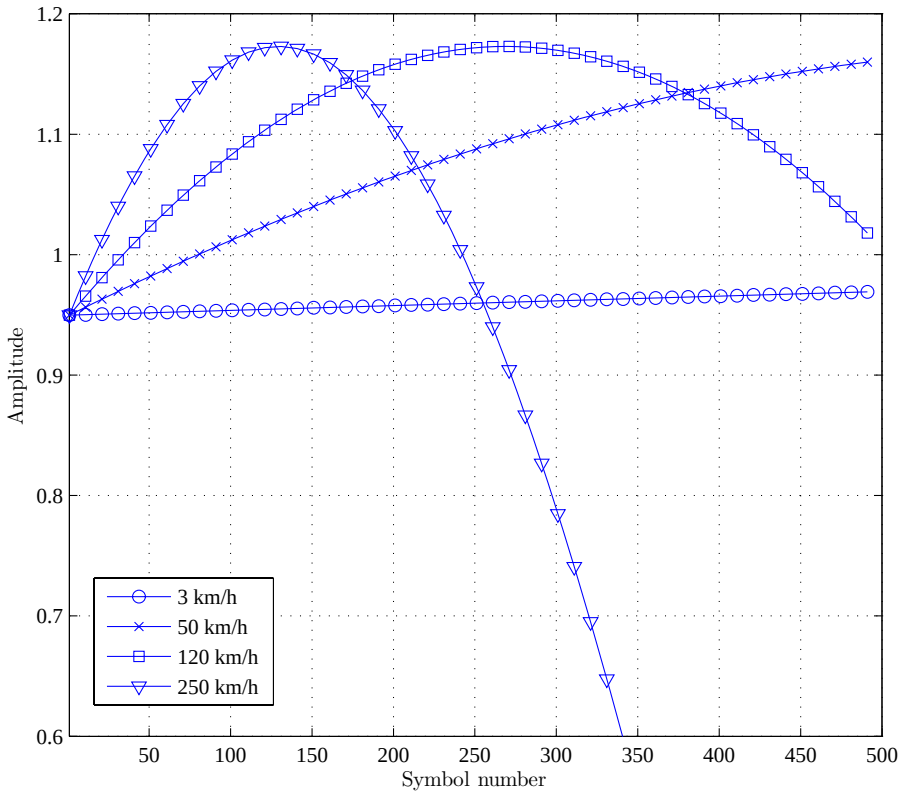

FIGURE 3.16: Fading vectors for relative mobile velocities of 3 km/h, 50 km/h, 120 km/h and 250 km/h.

A number of power delay profiles are used in this thesis to evaluate the performance of the system for various channel conditions. The power delay profile models are described in turn below:

3.5.5.1 Exponential PDP

The exponential PDP coefficients are determined by

$$v_k = exp\Big(\frac{-k\tau_{max}}{L\tau_e}\Big) \tag{3.16}$$

where $k = 0, 1, 2, ..., L-1$, τ_{max} is the channel delay spread duration, and τ_e is the time constant of the profile, determined by

$$\tau_e = \frac{-\tau_{max}}{ln\Big(10^{\frac{P_{drop}}{10}}\Big)}, \tag{3.17}$$

where P_{drop} is the relative power drop between $t = 0$ and $t = \tau_{max}$, with a value of -30 dB [40]. The normalized exponential PDP coefficients are shown in Fig. 3.17 for $L = 50$.

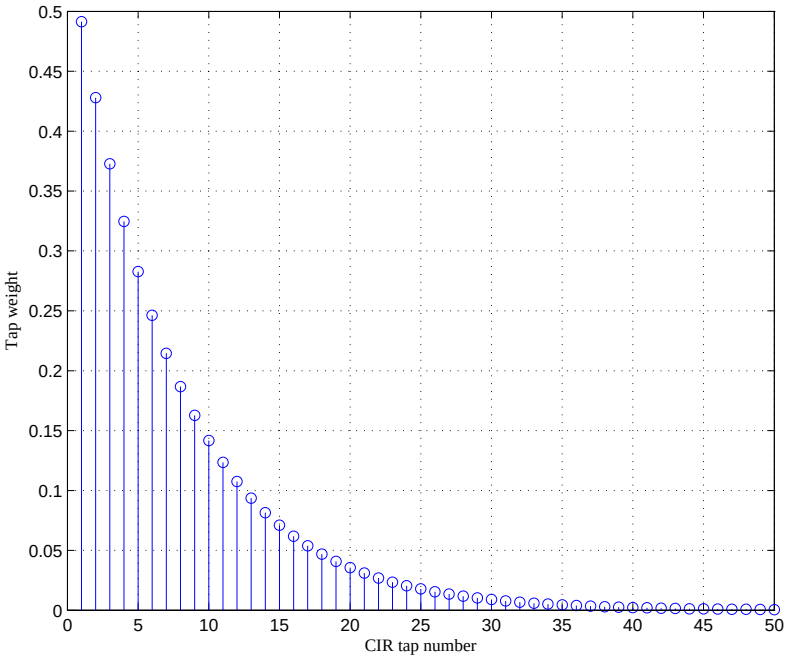

FIGURE 3.17: Exponential power delay profile for $L = 50$.

3.5.5.2 Linear PDP

The linear PDP coefficients are determined by

$$v_k = \frac{L-k}{L}, \tag{3.18}$$

where $k = 0, 1, 2, ..., L-1$. The normalized linear PDP is shown in Fig. 3.18 for $L = 50$.

3.5.5.3 Random Profile

The random PDP coefficients are determined by

$$v_k = Random(), \tag{3.19}$$

where $k = 0, 1, 2, ..., L-1$ and the function $Random()$ generates uniformly distributed random numbers such that $v_k \in [0; 1]$. Fig. 3.19 shows a normalized random PDP $L = 50$.

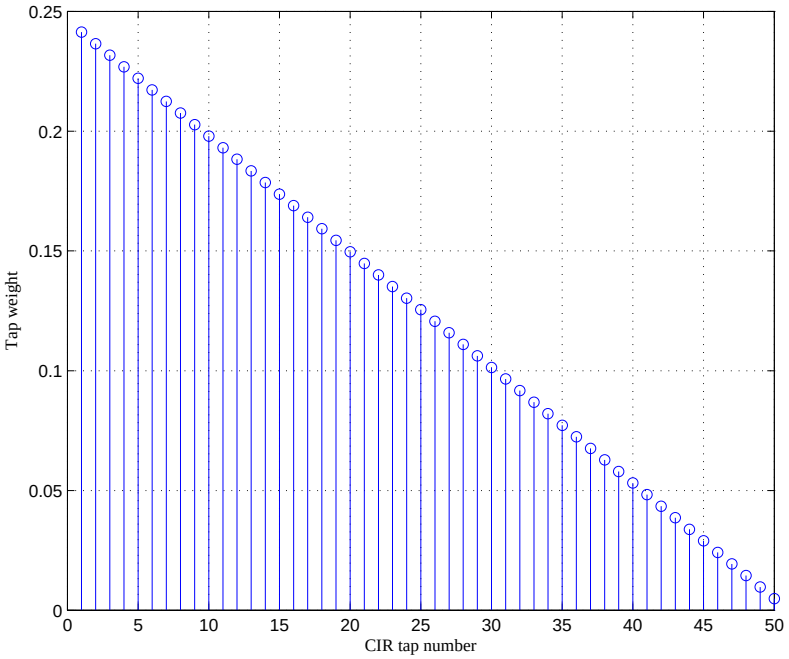

FIGURE 3.18: Linear power delay profile for $L = 50$.

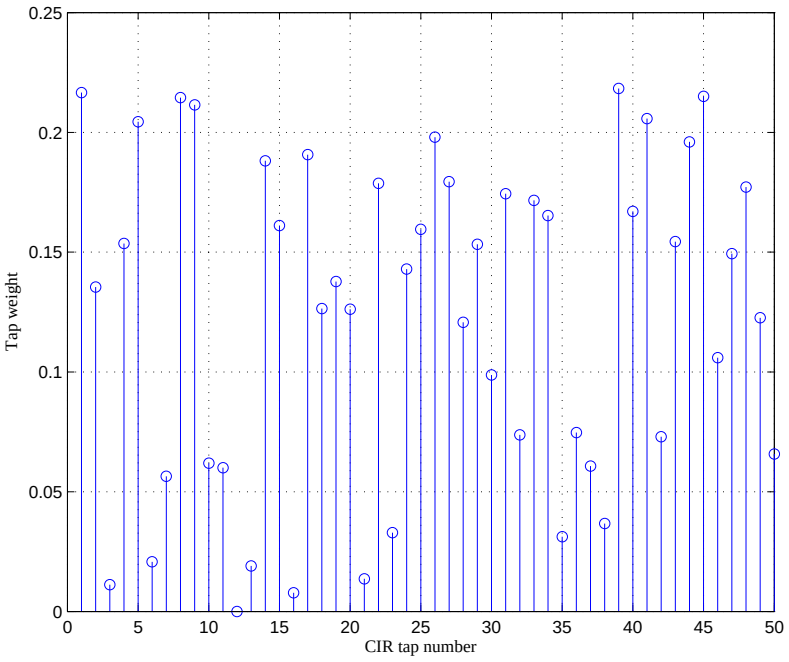

FIGURE 3.19: Random power delay profile for $L = 50$.

3.5.5.4 Uniform Profile

The uniform PDP coefficients are determined by

$$v_k = 1, \tag{3.20}$$

where $k = 0, 1, 2, ..., L-1$. Fig. 3.20 shows a normalized uniform PDP for $L = 50$.

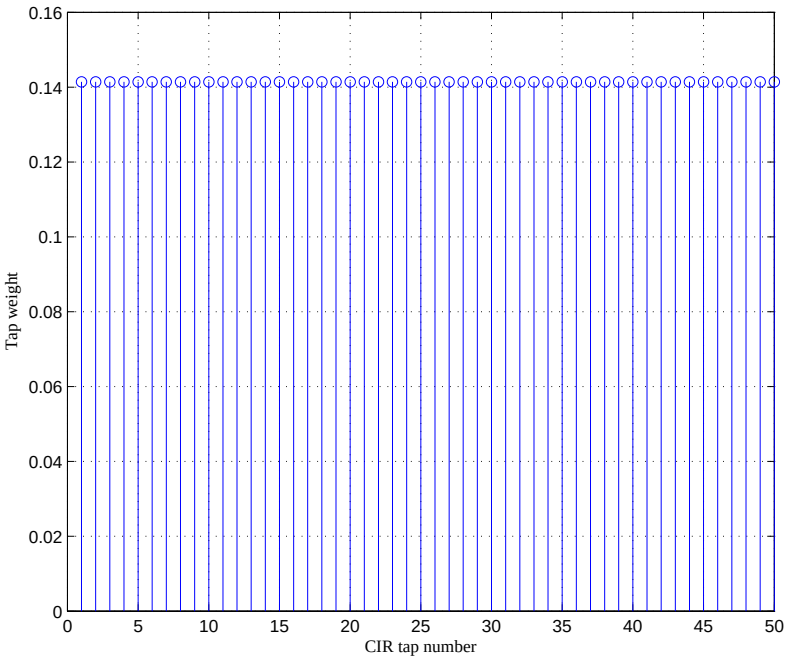

FIGURE 3.20: Uniform power delay profile for $L = 50$.

3.5.5.5 Energy Normalization

Disregarding fading, it is assumed that the received energy is equal to the transmitted energy. In a system transmitting information through a multipath channel, the transmitted energy is spread among the various independent fading channels. Since it is assumed that the received energy is equal to the transmitted energy the nominal channel weights $\mathbf{v} = \{v_0, v_1, \cdots, v_{L-1}\}^T$ are normalized such that

$$\mathbf{v}^{'}\mathbf{v} = 1. \tag{3.21}$$

To achieve this, irrespective of the power delay profile of the channel, the nominal tap weights are divided by the norm of the nominal channel weight vector. That is,

$$\mathbf{v} = \frac{v_0}{\|\mathbf{v}\|}, \frac{v_1}{\|\mathbf{v}\|}, \cdots, \frac{v_{L-1}}{\|\mathbf{v}\|}, \tag{3.22}$$

where the norm $\|\mathbf{v}\|$ is expressed as

$$\|\mathbf{v}\| = \sqrt{v_0^2 + v_1^2 + ... + v_{L-1}^2}. \tag{3.23}$$

The normalized nominal tap weights are then used to scale the respective Rayleigh fading vectors for each corresponding CIR tap, which will ensure that no energy is added during transmission.

3.6 Channel Estimation

In a digital communication system, the channel estimator is used at the receiver to estimate the impulse response of the overall channel between the transmitter and the receiver. As described earlier, the received signal suffers from multipath, resulting in ISI at the receiver, necessitating equalization in an attempt to mitigate the effect of ISI in the received symbols. To effectively invert the effect of the channel on the transmitted symbols, some measure thereof is needed. The channel estimator provides an estimate of the channel in the form of a discrete impulse response.

In short, the channel estimator works as follows: the receiver has knowledge of a number of the symbols that are transmitted in every data block. This series of symbols is agreed upon by the transmitter and the receiver and are commonly known as the pilot or training symbols. The channel estimator examines these pilot symbols to determine how they interfered with each other during transmission. Because it is assumed that the channel is static, or time-invariant for the duration of a data block, it is also assumed that the ISI experienced by the pilot symbols will also have been experienced by the other symbols in that data block. The channel estimator module produces a vector known as the CIR that contains the coefficients of a finite impulse response filter. The CIR reprensents the channel by which the transmitted symbols were filtered during transmission. Channel estimation is performed for each data block that arrives at the receiver.

The effectiveness of channel estimation is affected by the data block length and the number of pilot symbols in the data block. Firstly, the data block length has to be kept short enough so that the assumption of a time-invariant channel holds. If the duration of a data block exceeds the coherence time T_c of the channel, the time-invariance assumption fails, which will result in poor channel estimation. Secondly, the number of pilot symbols in the data block must be high enough to include all of the multipath effects encountered by all the symbols in the data block, but at the same time low enough so as to minimize the data overhead in the

data block. In the Global System for Mobile communications (GSM) standard, 18 percent of the transmitted symbols are pilot symbols [41].

3.6.1 Least Squares Channel Estimation

In this thesis Least Squares (LS) approximation is used to perform channel estimation. Assuming a static channel for the duration of one data block, channel estimation can easily be formulated in the LS sense. The explanation is closely related to the derivation in [22].

The channel estimator receives $K+1$ ISI-corrupted pilot symbol observations $\mathbf{r} = \{r_1, r_2, ..., r_K\}^T$, where $K+1$ is the number of pilot symbols and the channel impulse response is denoted by $h = \{h_0, h_1, ..., h_{L-1}\}$, where the observations are described by

$$\mathbf{r} = \mathbf{Q}\mathbf{h} + \mathbf{n}. \tag{3.24}$$

The matrix $\mathbf{Q}$ is fully populated by $K+1$ known pilot symbols, corresponding to the ISI-corrupted received pilot symbols $\mathbf{r}$, and $\mathbf{n}$ contains Gaussian noise samples from the distribution $\mathcal{N}(0, \sigma^2)$. The matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = \begin{bmatrix} t_n & t_{n-1} & \cdots & t_{n-L+1} & t_{n-L} \\ t_{n+1} & t_n & \ddots & t_{n-L} & t_{n-L+1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ t_{n+(K-1)} & t_{n+(K-2)} & \ddots & t_{n-L+K} & t_{n-L+(K-1)} \\ t_{n+K} & t_{n+(K-1)} & \cdots & t_{n-L+K+1} & t_{n-L+K} \end{bmatrix} \tag{3.25}$$

To estimate $\mathbf{h}$, the squared error between $\mathbf{r}$ and $\mathbf{Q}\mathbf{h}$ is

$$\epsilon^2 = tr[(\mathbf{r} - \mathbf{Q}\mathbf{h})^\dagger(\mathbf{r} - \mathbf{Q}\mathbf{h})] = \mathbf{n}^\dagger\mathbf{n}. \tag{3.26}$$

which is differentiated with respect to $\mathbf{h}$ to obtain

$$\frac{\partial \epsilon^2}{\partial \mathbf{h}} = 2\mathbf{Q}^\dagger(\mathbf{r} - \mathbf{Q}\mathbf{h}). \tag{3.27}$$

Equating the gradient to zero produces the LS estimate

$$\hat{\mathbf{h}} = (\mathbf{Q}^\dagger\mathbf{Q})^{-1}\mathbf{Q}^\dagger\mathbf{r}, \tag{3.28}$$

The pilot sequence must be chosen such that its autocorrelation approaches the *Delta Dirca* function. This will ensure that the Grammian Matrix ($\mathbf{Q}^{\dagger}\mathbf{Q}$) is diagonally dominant and therefore invertible, which will simplify calculation of $\hat{\mathbf{h}}$ in (3.28).

3.6.2 Channel Impulse Response Characterization in Narrow and Wideband Systems

Fig. 3.21 shows a continuous channel impulse response in a hypothetical communication system. The delay spread τ_s, the time between the first and the last arrival of a transmitted symbol, is 25 μs.

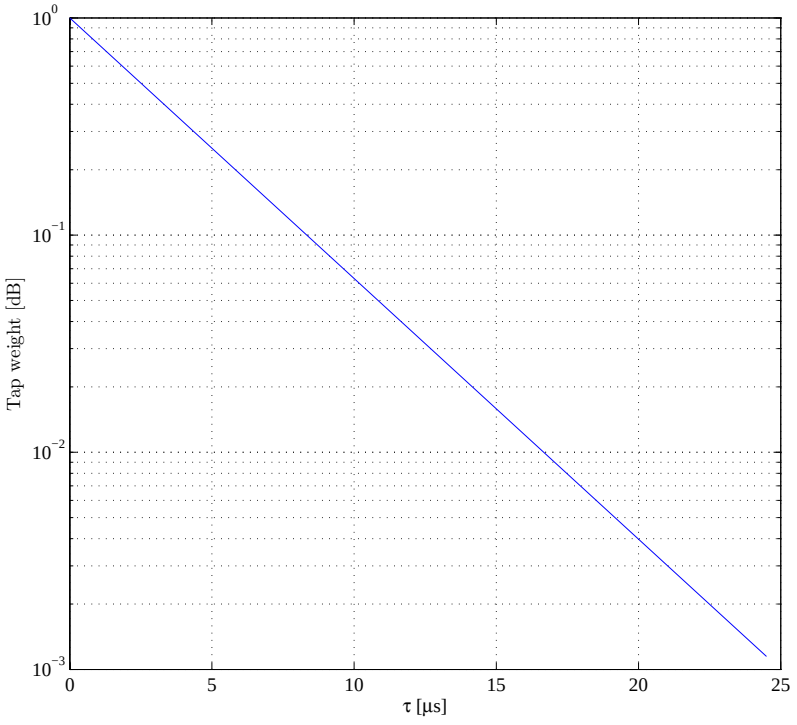

FIGURE 3.21: Continuous CIR.

3.6.2.1 Narrowband Systems

Assuming that the symbol period T_s in a narrowband communication system is 5 μs, the CIR length, or the number of resolvable CIR coefficients, can be calculated by

$$L = \frac{\tau_s}{T_s}, \tag{3.29}$$

rounding up to the highest integer, yielding $L = 5$ for this scenario. Upon demodulation and matched-filtering of the ISI-corrupted received signal, the channel impulse response is estimated by the channel estimator. The channel estimator produces the CIR with length L to be used together with the received sequence of symbols in the equalizer, in order to estimate the sequence of transmitted source symbols with maximum confidence. Fig. 3.22 shows the CIR produced by the channel estimator for the above scenario. Note that the CIR coefficients are samples of the channel delay spread, taken in the center of the symbol period.

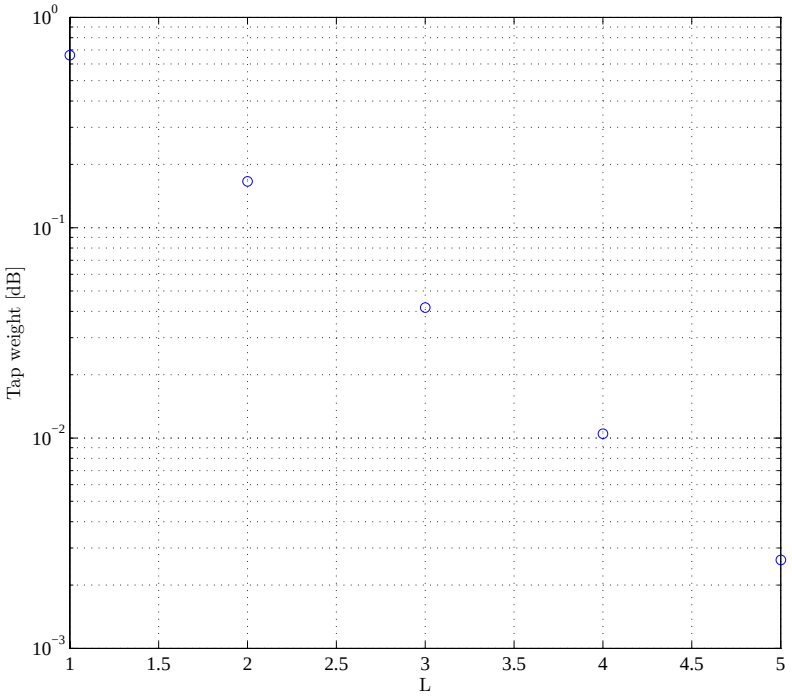

FIGURE 3.22: CIR for a narrowband system.

3.6.2.2 Wideband Systems

In a wideband communication system, L in (3.29) increases due to a decrease in the symbol period, given that τ_s remains constant. Assuming that the symbol period for a wideband communication system is $T_s = 0.5$ μs, given the same delay spread as before, the CIR length is determined by (3.29) to be $L = 50$. Fig. 3.23 shows the CIR produced by the channel estimator, where there are now ten times as many CIR coefficients as in the narrowband system.

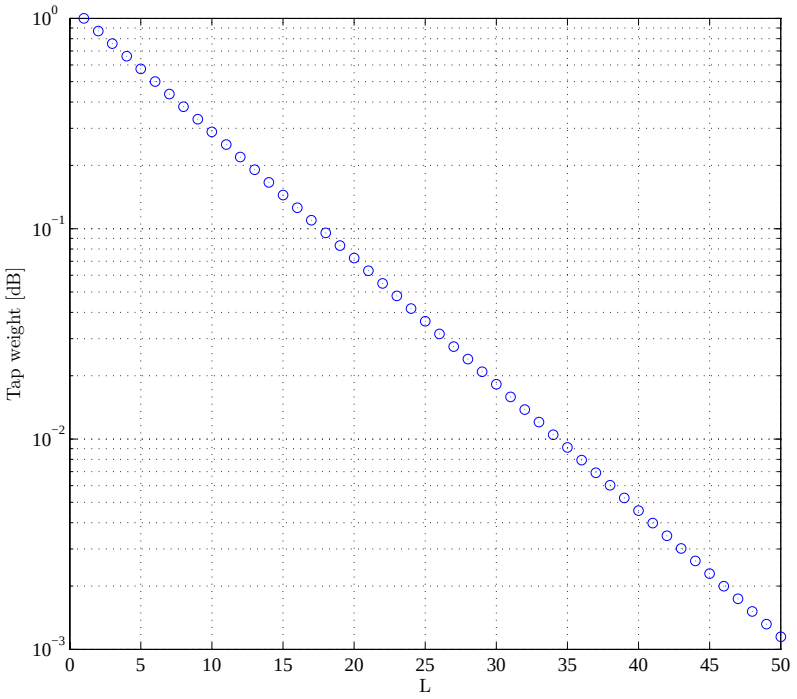

FIGURE 3.23: CIR for a wideband system.

3.7 Equalization

As stated before, the received symbols are described by [1, 4]

$$r_k = \sum_{j=0}^{L-1} h_j s_{k-j} + n_k \tag{3.30}$$

for systems employing single-carrier modulation in a multipath fading environment, assuming a time-invariant CIR, where s_k denotes the kth complex symbol in the transmitted sequence of N symbols chosen from an alphabet $\mathcal{D}$ containing M complex symbols, r_k is the kth received symbol, n_k is the kth Gaussian noise sample $\mathcal{N}(0, \sigma^2)$, and h_j is the jth coefficient of the estimated CIR. The equalizer is responsible for reversing the effect of the channel on the transmitted symbols in order to produce an estimated sequence of transmitted symbols with maximum confidence.

Bayes' theorem can be used to express the probability of a sequence of transmitted symbols [4],

$$P(\mathbf{s}|\mathbf{r}) = \frac{P(\mathbf{r}|\mathbf{s})P(\mathbf{s})}{P(\mathbf{r})}, \tag{3.31}$$

where $\mathbf{s}$ and $\mathbf{r}$ are the respective estimated transmitted symbol sequence and received symbol sequence. The likelihood $P(\mathbf{r}|\mathbf{s})$ that a sequence $\mathbf{r}$ was observed, given a sequence $\mathbf{s}$ is expressed as [21]

$$P(\mathbf{r}|\mathbf{s}) = \prod_{k=1}^{N} P(r_k|s_k). \tag{3.32}$$

Because it is assumed that all the symbols from the modulation alphabet $\mathcal{D}$ are equiprobable, $P(\mathbf{s})$ is reduced to a constant. Also, since $P(\mathbf{r})$ is a common denominator, it will not influence the choice of $\mathbf{s}$.

3.7.1 MLSE Equalizer

In 1972 Forney [1] showed that the Viterbi algorithm [2], first developed by Viterbi in 1967 to decode convolutional error correction codes, also an NP-complete problem, can be used to determine the most likely transmitted sequence. This is done by using a trellis, a special graph, or remerging tree structure, representing all possible combinations of transmitted symbols, to determine the solution with the lowest cost through the trellis. The sequence of symbols with the lowest cost maximizes the probability that said sequence was transmitted, thus producing the optimal estimate for the transmitted sequence [4]. Although the necessary processing power was not available at that time, this pioneering work would change the face of wireless communications forever. MLSE exploits the redundancy provided by the interfering symbols to optimally reconstruct the transmitted symbol sequence from the ISI-corrupted received symbols, resulting in optimal sequence detection.

The problem of MLSE equalization is an NP-complete mathematical problem, becoming increasingly complex to solve as the channel memory grows.[6] Complete enumeration is not a feasible solution for solving the MLSE problem, as the complexity thereof is exponentially related to the number of possible solutions.

Since it is assumed that the received symbol sequence is corrupted by white noise, $P(r_k|s_k)$ is expressed as

$$P(r_k|s_k) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{\left| r_k - \sum_{j=0}^{L-1} h_j s_{k-j} \right|^2}{2\sigma^2} \right). \tag{3.33}$$

Substituting (3.33) in (3.32), $P(\mathbf{r}|\mathbf{s})$ can be rewritten as

[6] The length of the sequence to be detected N and the length of the CIR L dictates the size of the solution space.

$$P(\mathbf{r}|\mathbf{s}) = \frac{1}{\left(\sqrt{2\pi}\sigma\right)^N} exp\left(-\frac{\sum_{k=1}^{N}\left|\, r_k - \sum_{j=0}^{L-1} h_j s_{k-j}\right|^2}{2\sigma^2}\right). \tag{3.34}$$

As stated, it is assumed that all the symbols from the modulation alphabet $\mathcal{D}$ are equiprobable, and therefore $P(\mathbf{s})$ is reduced to a constant. Also, since $P(\mathbf{r})$ is a common denominator, it will not influence the choice of $\mathbf{s}$. It is concluded that it is necessary to find the sequence of symbols that maximizes (3.34). In other words, to optimally estimate the transmitted sequence of length N in a wireless communication system, the cost function [1]

$$\mathcal{L} = \sum_{k=1}^{N}\left| r_k - \sum_{j=0}^{L-1} h_j s_{k-j}\right|^2, \tag{3.35}$$

must be minimized. Here $\mathbf{s} = \{s_1, s_2, \cdots, s_N\}^T$ is the most likely transmitted sequence that will maximize $P(\mathbf{r}|\mathbf{s})$ and therefore $P(\mathbf{s}|\mathbf{r})$. It must be noted that, although the minimization of (3.35) will maximize $P(\mathbf{s}|\mathbf{r})$, the resulting sequence estimates will only be optimal in the sequence sense. There is therefore no sequence of transmitted symbols that are more probable, but the probability of each estimated symbol in the sequence will not necessarily be maximized among all possible transmitted symbols from the alphabet $\mathcal{D}$. The MAP algorithm discussed in *Section 3.7.2* achieves this. In order to minimize (3.35), the min-sum algorithm is used.

3.7.1.1 The Min-Sum Algorithm

Viterbi and Forney in [1, 2] showed that the min-sum algorithm can be used to eliminate more costly contending paths at each state or node, in the trellis, thereby greatly reducing the computational complexity due to complete enumeration. For equalization, by eliminating more costly contending paths, there will always be M^{L-1} possible paths at every stage in the trellis, where M is the modulation alphabet size, which will be reduced to only one winning path at the end of the trellis.[7]

Consider a BPSK communication system transmitting N symbols through a multipath channel with a CIR of length $L = 4$. The number of states per stage in the trellis will therefore be $M^{L-1} = 2^3$ and the trellis will have N stages, one state for every transmitted symbol. Also assume that $L-1 = 3$ known tail symbols are added to the front and back of the transmitted sequence of N symbols, where each of the tail symbols is equal to 1. The trellis structure is shown in Fig. 3.24.

The trellis diagram in Fig. 3.24 shows all possible transitions, where the solid lines indicate a 1 transition and the dashed lines indicate a -1 transition. In order to find the sequence of

[7] The trellis is initiated and terminated by adding $L-1$ known tail symbols to the front and back of the transmitted data sequence to initialize and terminate the trellis.

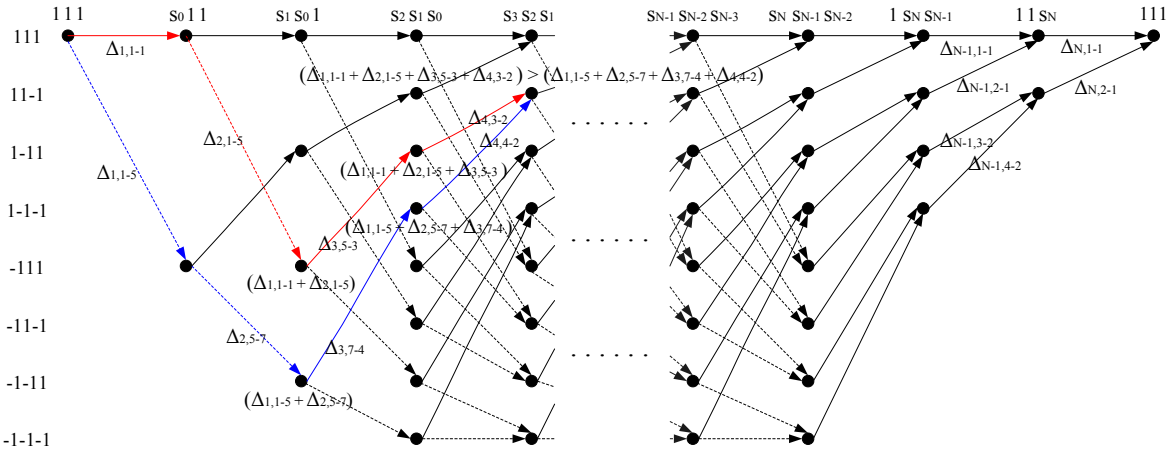

FIGURE 3.24: Trellis structure for BPSK with $L = 4$.

symbols that will maximize $P(\mathbf{s}|\mathbf{r})$, the shortest path through the trellis must be found. The cost between each possible transition is calculated according to

$$\Delta_{k,i-j} = \left| r_k - \sum_{j=0}^{L-1} h_j s_{k-j} \right|^2, \tag{3.36}$$

where $i = 1, 2, ..., 2^3$ denotes the number of the parent state, $j = 1, 2, ..., 2^3$ denotes the number of the child state, and k is the number of the stage in the trellis. At each state, all the previous $\Delta_{k,i-j}$'s are accumulated, and where there are contending paths, the path with the largest accumulated cost is eliminated.

Consider state two in stage four in the trellis diagram in Fig. 3.24. To determine which path will survive in the trellis, the accumulated cost of both paths leading up to that state is compared. If the cost of the red path is greater than that of the blue path, the red path is eliminated.

There will therefore be 2^3 surviving paths in each stage on the trellis for stages beyond $k = 2$. When stages $k = N - 2$ to $k = N$ are considered, the number of allowed transition decreases, due to the known tail symbols added to the end of the transmitted symbol sequence. Because these tail symbols are known to be equal to 1, only 1-transitions are allowed. For the last few states in the trellis the contending paths are also eliminated, until only one possible path remains at stage $k = N$. This path is then traced back to determine the most probable sequence of transmitted symbols.

The MLSE equalizer produces outputs from the set $\mathcal{D}$ containing M symbols. These estimates are called hard outputs, since the estimates do not contain any probabilistic information

regarding the reliability of those estimates. The MAP equalizer can be used to produce probabilistic information as an indication of the reliability of the estimates. The MAP algorithm used for equalization is discussed next.

3.7.2 MAP Equalizer

Following the development of the Viterbi MLSE decoding algorithm [2], the BCJR algorithm [3], named after its developers L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, also known as the MAP algorithm, was developed in 1974, also for the decoding of convolutional codes. In the artificial intelligence community this algorithm was developed independently by Pearl and is called *Belief Propagation* [42]. The MAP algorithm developed for the decoding of convolutional codes is easily modified to produce the posterior probability of each symbol in the estimated transmitted sequence, as opposed to maximizing the probability of the whole transmitted sequence, as done by the Viterbi MLSE equalizer [4].

In an uncoded system the use of a MAP equalizer instead of a MLSE equalizer is obsolete, as the overall performance will be no different.[8] However, when error-correction coding is employed in the system, the ability of the MAP equalizer to produce posterior probabilities on the transmitted symbols is beneficial to the decoder. Also, when Turbo Equalization is employed, the MAP algorithm is used in the equalizer and the decoder [11, 12, 13, 14].

The aim of the MAP equalizer is to maximize posterior distribution directly. While the prior probabilities $P(\mathbf{s})$ of the transmitted symbols are assumed by the MLSE equalizer to be equal, the MAP equalizer is able to consider the prior probabilties $P(\mathbf{s})$, if necessary, in order to produce posterior probabilistic information on each individual transmitted symbol. This algorithm produces the marginalized probability for each estimated symbol. That is [21]

$$P(s_k|\mathbf{r}) = \sum_{s_{k^\dagger}:k^\dagger \neq k}^{N} P(\mathbf{s}|\mathbf{r}) \tag{3.37}$$

where $\mathbf{r}$ is the received sequence, s_k is the kth symbol to be estimated from a modulation alphabet consisting of M symbols, and N is the number of symbols in the received sequence. The probability of a transition from s_j at time $k-1$ to s_i at time k is given

$$\omega_k(s_j, s_i) = P(\mathbf{r}_k|s_j, s_i)P(s_k). \tag{3.38}$$

where $P(\mathbf{r}_k|s_j, s_i)$ is given by

[8] The MAP equalizer also has approximately twice the computational complexity of the MLSE equalizer.

$$P(\mathbf{r}_k|s_j, s_i) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(\frac{-\left|r_k - \sum_{n=0}^{L-1} h_n s_{k-n}\right|^2}{2\sigma^2}\right). \tag{3.39}$$

By considering the the relevant transition value[9] s_ξ, $P(s_k)$ can be written as [12]

$$P(s_k) = exp\left(\frac{1}{2} s_\xi L(\hat{s_k})\right), \tag{3.40}$$

where $L(.)$ denotes the log-likelihood ratio (LLR) operation and $\hat{s_k}$ is an estimate of s_k. Therefore, substituting (3.39) and (3.40) in (3.38), yields

$$\omega_k(s_j, s_i) = \frac{exp\left(\frac{1}{2} s_\xi L(\hat{s_k})\right)}{\sqrt{2\pi\sigma^2}} exp\left(\frac{-\left|r_k - \sum_{n=0}^{L-1} h_n s_{k-n}\right|^2}{2\sigma^2}\right), \tag{3.41}$$

which completely describes the probability of a transition from s_j at time $k-1$ to s_i at time k, based on all available information. Thus, for a transition labeled $s_\xi = 1$ and $L_e^D(\hat{s_k})$ equal to any large positive value,[10] $P(s_k)$ will be large, confirming the transition. Similarly, for a transition labeled $\xi = -1$ and $L_e^D(\hat{s_k})$ equal to any large negative value $P(s_k)$ will be large, also confirming the transition. However, for a transition labeled $s_\xi = 1$ and $L(s_\xi)$ equal to any large negative number, or for a transition labeled $s_\xi = -1$ and $L_e^D(\hat{s_k})$ equal to any large positive number, $P(s_k)$ will be small. Thus, if the prior information $L_e^D(\hat{s_k})$ is in agreement with the transition value s_ξ, the probability of that transition will increase, confirming that transition. Otherwise, if the prior information $L_e^D(\hat{s_k})$ contradicts the transition value s_ξ, the probability of that transition will be decreased.

Propagating the transition probabilities $\omega_k(s_j, s_i)$ across the whole sequence and marginalizing according to (3.37) will produce the posterior probabilities of each estimated s_k. The sum-product algorithm, also called the forward-bakward algorithm [21], achieves exactly this.

3.7.2.1 The Sum-Product Algorithm

The Sum-Product algorithm, also known as the Forward-Backward algorithm, uses the trellis in Fig. 3.24 to perform marginalization. This algorithm follows three steps [21]:

1. Determine the forward pass messages from left to right on the trellis.

2. Determine the backward pass messages from right to left on the trellis.

[9] $s_\xi \in$ {-1,1} for BPSK.
[10] The magnitude of $L_e^D(\hat{s_k})$ gives an indication of the confidence of that estimate.

3. Multiply, scale and accumulate (marginalize) probabilities at each stage of the trellis.

To determine the forward pass messages on the trellis, let a counter k run from left to right (from 1 to N) on the trellis and compute for each state in the trellis

$$\alpha_{i,k} = \sum_{j \epsilon Parent(i)} \omega_k(s_j, s_i)\alpha_{j,k-1}$$

where j represents the parent states of the current state at stage i of the trellis and $\omega_k(s_j, s_i)$ is the probability associated with the transition from s_j to s_i. Note that $\alpha_{j,0} = 1$. Similarly, to determine the backward pass messages on the trellis, let a counter k run from right to left (from $N-1$ to 1) on the trellis and compute for each state in the trellis

$$\beta_{j,k} = \sum_{j \epsilon Parent(i)} \omega_k(s_j, s_i)\beta_{i,k+1}$$

where again j represents the parent states of the current state at stage i of the trellis and $\omega_k(s_j, s_i)$ is the probability associated with the transition from s_j to s_i. Note that $\beta_{i,k} = 1$.

Finally, the exact marginalized symbol probability of each detected symbol is determined by summing over all nodes in each stage corresponding to a transition of either $s_\xi = 1$ or $s_\xi = -1$ such that

$$P(s_k = 1|\mathbf{r}) = \sum_{j \epsilon Parent(i), s_\xi = 1} \alpha_{j,k-1}\omega_k(s_j, s_i)\beta_{i,k} \tag{3.42}$$

$$P(s_k = -1|\mathbf{r}) = \sum_{j \epsilon Parent(i), s_\xi = -1} \alpha_{j,k-1}\omega_k(s_j, s_i)\beta_{i,k} \tag{3.43}$$

The MAP algorithm can also produce soft bits. The soft bits, also called LLRs, can be determined by

$$L(\hat{s_k}) = log\left(\frac{P(s_k = 1|\mathbf{r})}{P(s_k = -1|\mathbf{h})}\right). \tag{3.44}$$

where the sign of $L(\hat{s_k})$ indicates whether $\hat{s_k} = -1$ or $\hat{s_k} = 1$, and the magnitude of $|L(\hat{s_k})|$ is a measure of the confidence of that estimate.

3.7.3 MMSE Equalizer

Minimum Mean Square Error (MMSE) equalizers are of the class of linear equalizers. The MMSE equalizer aims to determine a set of coefficients $\mathbf{w} = \{w_1, w_2, \cdots, w_I\}$, where I is

the number of filter coefficients, based on the mean square error (MSE) criterion, in order to mitigate the effect of ISI on the transmitted symbols. The filter coefficients are chosen so as to minimize

$$J = E\left|s_k - \hat{s}_k\right|^2, \tag{3.45}$$

where s_k and $\hat{s}_k$ are the kth transmitted and estimated symbols respectively [10]. Because the MMSE equalizer is a linear equalizer, the estimate $\hat{s}_k$ is a linear combination of the ISI-corrupted received symbols r_k to r_{k-I}. Therefore $\hat{s}_k$ is determined by

$$\hat{s}_k = \sum_{i=1}^{i=I} w_i r_{k-i}, \tag{3.46}$$

which can be written in vector form as

$$\hat{s}_k = \mathbf{w}^T \mathbf{r}_{k-I:k} = \mathbf{r}_{k-I:k}^T \mathbf{w}, \tag{3.47}$$

where the subscripts indicate the segment of elements considered in $\mathbf{r}$. Assuming that the additive noise is white, the filter coefficients $\mathbf{w}$ can be determined. (3.45) can be rewritten as

$$J = E\left[\mathbf{w}^T \mathbf{r}\mathbf{r}^H \mathbf{w}^* - Re\left\{\mathbf{r}^H \mathbf{w}^* s_k\right\} + \left|s_k\right|^2\right| \tag{3.48}$$

[10], where T, H and $*$ are the transpose, Hermitian and complex conjugate operations respectively. Now we set $\mathbf{M_r} = E\left[\mathbf{r}\mathbf{r}^H\right]$ and $\mathbf{r_d} = E\left[\mathbf{r}^H s_k\right]$, where $\mathbf{M_r}$ is an $I \times I$ Hermitian matrix and r_d is a row vector length I. Assuming $\left[E|s_k^2|\right] = 1$, (3.48) can be rewritten as

$$J = \mathbf{w}^T \mathbf{M_r} \mathbf{w}^* - Re\left\{\mathbf{r_d} \mathbf{w}^*\right\} + 1. \tag{3.49}$$

In order to obtain the optimal weights $\mathbf{w}$, the gradient of J must be set to zero. Thus,

$$\frac{\partial J}{\partial \mathbf{w}} = \left(\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, ..., \frac{\partial J}{\partial w_I}\right) = 0. \tag{3.50}$$

Differentiating (3.49) gives

$$\frac{\partial J}{\partial \mathbf{w}} = 2\mathbf{w}^T \mathbf{M_r} - 2\mathbf{r_d}, \tag{3.51}$$

and setting (3.51) equal to zero gives

$$\mathbf{w} = (\mathbf{M_r})^{-1}\,\mathbf{r_d}^H, \tag{3.52}$$

which produces $\mathbf{w}$ so as to minimize (3.45) [10]. After the coefficients are determined, the transmitted symbol estimates are determined by (3.46).

The MMSE equalizer can be modified as in [14] to incorporate prior probabilistic information in the estimation process, thus making it fit for use in iterative equalization schemes, of which Turbo Equalization is discussed next.

3.7.4 RAKE Demodulator

In direct sequence spread spectrum (DSSS) systems the signal bandwidth is increased beyond what is necessary for effective communication, by modulating the transmitted signal with a spreading sequence [10]. Given a system with symbol period T_s, a spreading sequence with chip duration $T_c << T_s$, is used to transform each symbol into T_s/T_c chips, thus increasing the bandwidth by T_s/T_c. All users' chip sequences are summed together to form a composite signal, which is transmitted. See *Appendix A* for more information on spread spectrum principles.

As explained in *Section 3.6.2*, there will be a corresponding increase in the CIR length as the signal bandwidth increases. This is due to interchip interference, analogous to ISI, in DSSS systems. The RAKE demodulator is used in a DSSS system to coherently combine the chips of the received composite signal in order to reconstruct the composite transmitted signal, which is then correlated with each user's spreading code in order to retrieve the corresponding transmitted information [10]. A RAKE demodulator recombines the energy spread across the channel such that

$$s_k = \sum_{l=0}^{L-1} h_l^* r_{k-l}, \tag{3.53}$$

where s_k is the kth estimated chip of the composite signal, h_l^* is the complex conjugate of the lth estimated CIR coefficient, and r_{k-l} is the $(k-l)$th received chip.

The RAKE demodulator is suboptimal in that it does not attempt to estimate the maximum likelihood (ML) solution as explained in *Section 3.7.1*. This is an naive approach to detection. The HNN MLSE equalizer is used to replace the RAKE demodulator in a DSSS system, due to its ability to equalize signals in systems with long CIR lengths. Simulation results are presented in *Chapter 5*.

3.7.5 Turbo Equalizer

Turbo Equalization is a fairly new concept, based on the concept of Turbo Codes which have proven to be extremely effective in increasing system performance. In 1993 the *Turbo Principle* was introduced in [33] as applied to error-correction decoding. In 1995 the concept was applied to equalization in [11] with great success. Although very computationally complex to execute, Turbo Equalization has proven to be a very powerful iterative equalization/decoding technique, allowing for extraordinary gains in system performance.

A Turbo Decoder uses two MAP decoders to iteratively decode convolutional coded concatenated codes. Like the MAP equalizer, the MAP decoder produces posterior probabilistic information on the source symbols. The output of each decoder is therefore used to produce prior probabilistic information about the input symbols of the other decoder, thus allowing this scheme to exploit the inherent structure of the code to correct errors with each turbo iteration [4], achieving near Shannon limit performance in AWGN channels [11, 12, 13, 14].

Since the communication channel can be viewed as a non-systematic non-binary encoder [12], the channel can be viewed as an *outer* code while a convolutional encoder is used as an *inner* code in much the same way as in Turbo Coding. As such, one of the MAP decoders in the Turbo Decoder is substituted with a MAP equalizer to mitigate the effect of the channel on the transmitted symbols (to "decode" the ISI-corrupted received symbols) [12]. The output of the MAP equalizer is used to produce prior probabilistic information on the encoded symbols, which is exploited by the MAP decoder. In turn, the output of the MAP decoder is used to produce prior probabilistic information on the *un*-equalized received symbols, which is again exploited by the MAP equalizer. By iterating this system a number of times, the performance of the system can be enhanced greatly [11, 12, 13, 14].

Fig. 3.25 shows the structure of the Turbo Equalizer. The MAP equalizer takes the ISI-corrupted received symbols $\mathbf{r}$ and determines $L(\mathbf{r})$. The equalizer uses $L(\mathbf{r})$ together with the extrinsic information[11] $L_e^D(\hat{\mathbf{s}})$ as input and produces a sequence of posterior transmitted symbol estimates $L^E(\hat{\mathbf{s}})$. Extrinsic information $L_e^E(\hat{\mathbf{s}})$ is determined by

$$L_e^E(\hat{\mathbf{s}}) = L^E(\hat{\mathbf{s}}) - L_e^D(\hat{\mathbf{s}}), \tag{3.54}$$

which is then deinterleaved to produce $L_e^E(\hat{\mathbf{s}}')$. $L^E(\hat{\mathbf{s}}')$ and $L_e^E(\hat{\mathbf{s}}')$ are used as input to the MAP decoder, where $L_e^E(\hat{\mathbf{s}}')$ is used to provide prior information on the coded symbols, which produces a sequence of posterior source symbol estimates $L(\hat{\mathbf{u}})$. $L(\hat{\mathbf{u}})$ is encoded using a soft convolutional encoder[12] to give $L^D(\hat{\mathbf{s}}')$, which is used together with $L_e^E(\hat{\mathbf{s}}')$ to determine the extrinsic information

[11] The extrinsic information $L_e^D(\hat{\mathbf{s}})$ is zero during the first iteration.

[12] It was noted that none of the authors include this crucial step to produce N coded estimates from $N.R_c$ decoded estimates.

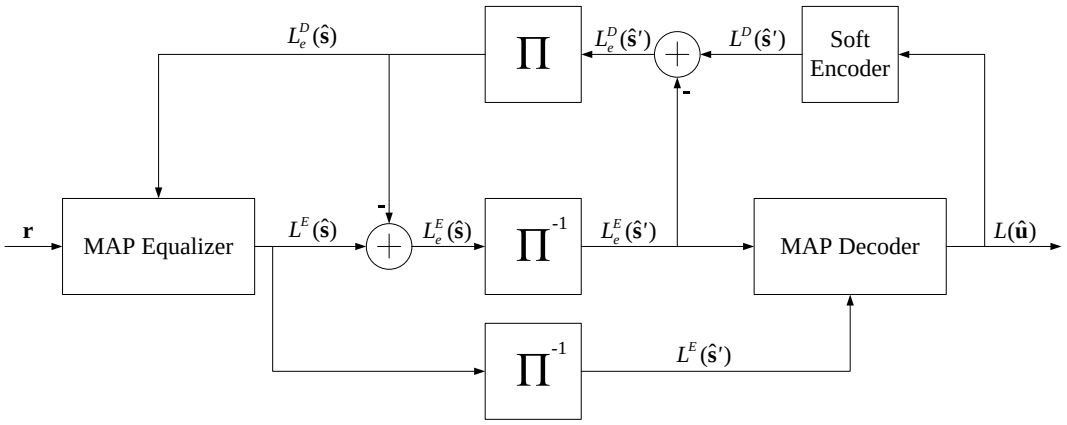

FIGURE 3.25: Turbo Equalizer.

$$L_e^D(\hat{\mathbf{s}}') = L^D(\hat{\mathbf{s}}') - L_e^E(\hat{\mathbf{s}}'). \tag{3.55}$$

$L_e^D(\hat{\mathbf{s}}')$ is then interleaved to produce $L_e^D(\hat{\mathbf{s}})$. The interleaved extrinsic information $L_e^D(\hat{\mathbf{s}})$ is used together with the received symbol LLRs $L(\mathbf{r})$ in the MAP equalizer, with $L_e^D(\hat{\mathbf{s}})$ serving to provide prior information on the received symbol LLRs.

The equalizer again produces posterior information $L^E(\hat{\mathbf{s}})$ of the interleaved coded symbols. This process continues until the outputs of the decoder settle, or until a predefined stop-criterion is met [12]. After termination, the output $L(\hat{\mathbf{u}})$ of the decoder gives an estimate of the source symbols $\mathbf{u}$.

The Turbo Equalizer in Fig. 3.25 does not exactly correspond to those in pioneering work [11, 12, 13, 14], but the structure of the Turbo Equalizer in Fig. 3.25 is implied in the work. The output of the MAP equalizer after deinterleaving, $L^E(\hat{\mathbf{s}})$, is used in the MAP decoder, together with the extrinsic information $L_e^E(\hat{\mathbf{s}}')$ wich serves as prior information on $L^E(\hat{\mathbf{s}})$. Hence connection between the equalizer and the decoder is made via a deinterleaver. Also, in order to produce N estimates $L^D(\hat{\mathbf{s}}')$ to be used in the calculation of extrinsic information $L_e^D(\hat{\mathbf{s}}')$, the output $L(\hat{\mathbf{u}})$ of the MAP decoder, which is a sequence of length $N.R_c$, must be encoded by a soft encoder corresponding to the modulo-two hard output convolutional encoder used in the transmitter (See Section 3.3.1).

3.7.5.1 The effect of priors on transition probabilities

The power of Turbo Equalization lies in the exchange of extrinsic information $L_e^E(\hat{\mathbf{s}})$ and $L_e^D(\hat{\mathbf{s}}')$ between the equalizer and the decoder. By feeding back the extrinsic information, without creating self-feedback loops, the correlation between prior information and output information is minimized, allowing the system to converge to an optimal state in the solution space [13, 14]. If information is exchanged directly between the equalizer and the decoder by

ingoring interleaving and/or extrinsic information, self-feedback loops will be formed. This will cause minimal performance gains, since the equalizer and the decoder will inform each other about information already attained in previous iterations [13].

Since SISO MAP algorithms are used for the equalizer and the decoder, prior probabilistic information can be exploited to produce more reliable posterior probabilities on the respective equalized and decoded symbols, allowing for substantial performance gains.

3.7.5.2 Reduced Complexity Turbo Equalization

As the CIR length increases, Turbo Equalization becomes exceedingly complex in terms of the number of computations. It is therefore instructive to consider using less complex SISO equalizers in an attempt to alleviate the computational strain due to long CIR lengths. MMSE equalizers have successfully been used in [13] and [14] to replace the MAP equalizer, and a soft DFE was developed in [6]. The subject of this thesis, the HNN MLSE equalizer, will also be used in the place of the MAP equalizer to enable near-optimal Turbo Equalization in extremely long channels. In *Section 5.5.1* it will be shown how the HNN MLSE equalizer is modified to produce soft outputs and to use prior probabilistic information. Simulation results are presented in *Chapter 5*.

3.8 Decoding

In a mobile communication system the transmitted signal is subject to energy losses due to multipath and fading, as well as interference due to thermal noise, resulting in unreliable estimates of source information in the receiver. In order to correct errors in the receiver, ECC is used to introduce redundancy to the source information, in a controlled fashion. The decoder exploits the structure introduced by the encoder in the encoded symbol sequence to reconstruct the source information from the estimated coded symbols, correcting errors while doing so. The importance of error-correction coding was discussed in *Section 3.3*.

The decoding of convolutional codes is closely related to equalization discussed in *Section 3.7*, in that the min-sum (Viterbi MLSE) and sum-product (MAP/BCJR) algorithms are also used for decoding. The MAP decoding algorithm is discussed here because it is an attractive choice for use in iterative equalization/decoding algorithms, of which Turbo Equalization (*Section 3.7.5*) is considered in this thesis. It is attractive for two reasons: firstly because it includes prior probabilistic information in the estimation, and secondly, it provides posterior soft estimates of individual symbols.

3.8.1 MAP Decoding

Bayes' theorem can be used to express the posterior probability of the codewords in a sequence [4],

$$P(\mathbf{C}|\mathbf{s}) = \frac{P(\mathbf{s}|\mathbf{C})P(\mathbf{C})}{P(\mathbf{s})}. \tag{3.56}$$

Assuming that a rate 1/3 systematic convolutional encoder is used, $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_{N_c}\}$ is the sequence of $N_c = NR_c = N/3$ codewords $\mathbf{c}_k = \{c_1^{(k)}, c_2^{(k)}, c_3^{(k)}\}$ where $k = 1, 2, 3, ..., N_c$, and $\mathbf{s}$ is the estimated symbol sequence of length N. The likelihood $P(\mathbf{s}|\mathbf{C})$ that a sequence $\mathbf{s}$ was observed given a sequence of codewords $\mathbf{C}$, is expressed as

$$P(\mathbf{s}|\mathbf{C}) = \prod_{k=1}^{N_c} P(\mathbf{s}_k|\mathbf{c}_k), \tag{3.57}$$

where the sequence $\mathbf{s}$ is segmented into N_c segments of $1/R_c = 3$ symbols such that each segment $\mathbf{s}_k = \{s_1^{(k)}, s_2^{(k)}, s_3^{(k)}\}$ corresponds to a codeword $\mathbf{c}_k$. Analogous to MAP equalization discussed in *Section 3.7.2*, the MAP decoder uses prior probabilities to produce posterior information on each decoded source symbol by calculating the marginalized probability for each estimated codeword, from which the originally transmitted symbols can be derived using the decoder state diagram as discussed in *Section 3.3.1*. The probability of a transition from codeword $\mathbf{c}_j$ at time $k-1$ to codeword $\mathbf{c}_i$ at time k is given

$$\omega_k(\mathbf{c}_j, \mathbf{c}_i) = P(\mathbf{s}_k|\mathbf{c}_j, \mathbf{c}_i)P(\mathbf{c}_k). \tag{3.58}$$

where $P(\mathbf{s}_k|\mathbf{c}_j, \mathbf{c}_i)$ is given by

$$P(\mathbf{s}_k|\mathbf{c}_j, \mathbf{c}_i) = \delta.exp\left(-\sum_{n=1}^{3}\left|c_n^{(k)} - s_n^{(k)}\right|^2\right), \tag{3.59}$$

where δ is a normalization constant and $c_n^{(k)}$ can only assume values as dictated by the coding rule.[13] By considering the relevant transition value $c_1^{(\xi)}$, at time instance ξ, as permitted by the coding rule, $P(\mathbf{c}_k)$ can be written as[14]

$$P(\mathbf{c}_k) = exp\left(\frac{1}{2}c_1^{(\xi)}L(\hat{c}_1^{(k)})\right), \tag{3.60}$$

[13] The state diagram of a convolutional encoder does not allow all codewords to precede and/or follow all other codewords.

[14] For a systematic code the first bit of the codeword is equal to the source bit.

where $\hat{c}_1^{(k)}$ is an estimate of $c_1^{(k)}$. By substituting (3.59) and (3.60) in (3.58), $\omega_k(\mathbf{c}_j, \mathbf{c}_i)$ can be rewritten as

$$\omega_k(\mathbf{c}_j, \mathbf{c}_i) = \delta.exp\left(-\sum_{n=1}^{3}\left|c_n^{(k)} - s_n^{(k)}\right|^2 + \frac{1}{2}c_1^{(\xi)}L(\hat{c}_1^{(k)})\right). \tag{3.61}$$

which completely describes the probability of a transition from $\mathbf{c}_j$ at time $k-1$ to $\mathbf{c}_i$ at time k, based on all available information. Using the sum-product algorithm as explained in *Section 3.7.2.1*, posterior probabilities of each estimated $\mathbf{c}_k$ can be determined, from which the source symbols can be derived using the state diagram of the encoder.

3.9 Concluding Remarks

In this chapter the most important concepts of digital communication systems were discussed. It was also explained how the communication system considered in this thesis will be simulated. Although the discussion is not exhaustive, the concepts relevant to this thesis were discussed in length and will serve as a basis for the discussions that follow.

Chapter 4

The HNN MLSE Equalizer

In this chapter the HNN MLSE equalizer is developed. To refresh the memory of the reader, the concept of MLSE equalization is discussed briefly, followed by the systematic derivation of the HNN MLSE equalizer for BPSK, 4-QAM and 16-QAM modulation. The iterative nature of the equalizer is then discussed, after which the realization of the relevant decision functions is discussed. Optimization techniques are discussed next, followed by a computational complexity analysis of the HNN MLSE equalizer. This chapter is concluded with a few general remarks.

4.1 MLSE Equalization

For systems employing single-carrier modulation in a multipath fading environment, the received symbols are described by [1, 4]

$$r_k = \sum_{j=0}^{L-1} h_j s_{k-j} + n_k, \tag{4.1}$$

where s_k denotes the kth complex symbol in the transmitted sequence of N symbols chosen from an alphabet $\mathcal{D}$ containing M complex symbols. r_k is the kth received symbol, n_k is the kth Gaussian noise sample $\mathcal{N}(0, \sigma^2)$ and h_j is the jth coefficient of the estimated CIR [43].

To find the most likely transmitted sequence $\mathbf{s} = \{s_1, s_2, \cdots, s_N\}^T$, the cost function [1]

$$\mathcal{L} = \sum_{k=1}^{N} \left| r_k - \sum_{j=0}^{L-1} h_j s_{k-j} \right|^2, \tag{4.2}$$

must be minimized, in order to maximize the probability $P(\mathbf{s}|\mathbf{r})$ to produce the estimated transmitted symbols with maximum confidence. The Viterbi MLSE equalizer is able to solve this problem exactly, with computational complexity linear in N and exponential in L [1]. The

HNN MLSE equalizer is also able to minimizes the cost function in (4.2), with computational complexity quadratic in N, but approximately independent of L, thus enabling it to perform near-optimal sequence estimation in systems with extremely long CIR lengths with very low computational cost.

4.2 Systematic Derivation

It was observed [15, 16, 17, 29, 30, 31, 32] that (4.2) can be written as

$$\mathcal{L} = -\frac{1}{2}\mathbf{s}^{\dagger}\mathbf{T}\mathbf{s} - \mathbf{I}^{\dagger}\mathbf{s}, \tag{4.3}$$

where $\mathbf{I}$ is a column vector with N elements, $\mathbf{T}$ is an $N \times N$ matrix, and † implies the Hermitian transpose, where (4.3) corresponds to the HNN energy function in (2.4). In order to use the HNN to perform MLSE equalization, the cost function (4.2) that is minimized by the Viterbi MLSE equalizer must be mapped to the energy function (4.3) of the HNN. This mapping is performed by expanding (4.2) for a given block length N and a number of CIR lengths L, starting at $L = 2$ and increasing L until a definite pattern emerges in $\mathbf{T}$ and $\mathbf{I}$ in (4.3). The emergence of a pattern in $\mathbf{T}$ and $\mathbf{I}$ will enable the realization of an MLSE equalizer for the general case, that is, for systems with any N and L, yielding a generalized HNN MLSE equalizer that can be used in a single-carrier communication system. The transmitted and received data block structure is shown in Fig. 4.1, where it is assumed that $L-1$ known tail symbols are appended and prepended to the block of payload symbols.

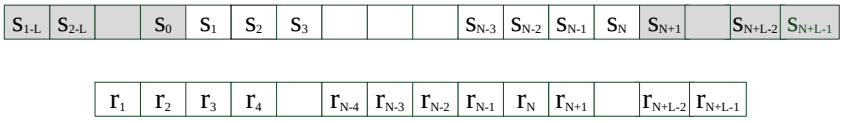

FIGURE 4.1: Transmitted (top) and received (bottom) data block structures. The shaded blocks contain known tail symbols

4.2.1 The BPSK HNN MLSE equalizer

The derivation of a general MLSE equalizer based on the HNN will follow from the separate derivations of independent MLSE equalizers for systems transmitting N symbols through channels with increasing CIR lengths, starting from $L = 2$. It is expected that a pattern will emerge in the matrix $\mathbf{T}$ and the vector $\mathbf{I}$ in (4.3) as L is increased. This pattern will then be used to derive a model for a generalized BPSK HNN MLSE equalizer. For BPSK modulation the tail symbols s_{1-L} to s_0 and s_N to s_{N+L-1} are equal to 1.

4.2.1.1 $L = 2$

The first step towards deriving an all-purpose BPSK HNN MLSE equalizer is to derive a model for the simplest case, that is, a communication system transmitting N BPSK modulated payload symbols through a multipath channel with a CIR length of $L = 2$. Expanding (4.2) for a data block length N and CIR length $L = 2$ yields

$$\begin{aligned}\mathcal{L} = \quad & r_1^2 - 2r_1h_0s_1 - 2r_1h_1s_0 + 2h_0h_1s_0s_1 + h_0^2s_1^2 + h_1^2s_0^2 \\ & r_2^2 - 2r_2h_0s_2 - 2r_2h_1s_1 + 2h_0h_1s_1s_2 + h_0^2s_2^2 + h_1^2s_1^2 \\ & r_3^2 - 2r_3h_0s_3 - 2r_3h_1s_2 + 2h_0h_1s_2s_3 + h_0^2s_3^2 + h_1^2s_2^2 \\ & \quad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ & r_N^2 - 2r_Nh_0s_N - 2r_Nh_1s_{N-1} + 2h_0h_1s_{N-1}s_N + h_0^2s_{N-1}^2 + h_1^2s_{N-1}^2\end{aligned} \tag{4.4}$$

which can be written as

$$\mathcal{L} = \sum_{k=1}^{N} r_k^2 - 2\sum_{k=1}^{N}\sum_{l=0}^{1} r_k h_l s_{k-l} + 2\sum_{k=1}^{N} h_0h_1s_{k-1}s_k + \sum_{k=1}^{N}\sum_{l=0}^{1} h_l^2 s_{k-l}^2. \tag{4.5}$$

By noting that the quadratic terms can be ignored, as they will only provide a positive offset to the total cost, or *energy*, in the network, (4.5) can be written as

$$\mathcal{L} = 2\sum_{k=1}^{N} h_0h_1s_{k-1}s_k - 2\sum_{k=1}^{N}\sum_{l=0}^{1} r_k h_l s_{k-l}, \tag{4.6}$$

which is in the form of (4.3). By inspection it is easy to determine $\mathbf{T}$ and $\mathbf{I}$ in (4.3). $\mathbf{T}$ is determined by

$$\mathbf{T} = -4\begin{bmatrix} 0 & \alpha_1 & 0 & \dots & 0 & 0 \\ \alpha_1 & 0 & \alpha_1 & 0 & \ddots & 0 \\ 0 & \alpha_1 & 0 & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \alpha_1 & 0 \\ 0 & \ddots & \ddots & \alpha_1 & 0 & \alpha_1 \\ 0 & 0 & \dots & 0 & \alpha_1 & 0 \end{bmatrix}, \tag{4.7}$$

where $\mathbf{T}$ is an $N \times N$ matrix, diagonally symmetric and banded with a band of non-zero elements. $\mathbf{T}$ is a correlation matrix, with T_{ij} describing the correlation between symbols i and j.[1] For $L = 2$ there is only one non-zero element, determined by

[1] T_{ij} is the weight between neurons i and j in the context of the HNN.

$$\alpha_1 = h_0 h_1. \tag{4.8}$$

I is determined by

$$\mathbf{I} = 4 \begin{bmatrix} r_1 h_0 + r_2 h_1 - \alpha_1 \\ r_2 h_0 + r_3 h_1 \\ r_3 h_0 + r_4 h_1 \\ \vdots \quad \vdots \quad \vdots \\ r_{N-2} h_0 + r_{N-1} h_1 \\ r_{N-1} h_0 + r_N h_1 \\ r_N h_0 + r_{N+1} h_1 - \alpha_1 \end{bmatrix}, \tag{4.9}$$

where **I** is a column vector of size N.

The derivation of **T** and **I** fully describes the BPSK HNN MLSE equalizer model for a data block length of N and CIR length of $L = 2$. The HNN can now be used to minimize (4.3) to determine the MLSE estimates for the transmitted symbols **s**. The iterative process during which MLSE estimation is performed, will be discussed in *Section 4.3*.

This concludes the derivation of the model for the BPSK HNN MLSE equalizer for systems transmitting N symbols through a channel with a CIR length of $L = 2$.

4.2.1.2 $L = 3$

As before, the first step in the derivation is to expand (4.2). Therefore, expanding (4.2) for a data block length N and CIR length $L = 3$ yields

$$\begin{aligned} \mathcal{L} = \quad & r_1^2 - 2r_1 h_0 s_1 - 2r_1 h_1 s_0 - 2r_1 h_2 s_{-1} + 2h_0 h_1 s_0 s_1 + \\ & 2h_1 h_2 s_{-1} s_0 + 2h_0 h_2 s_{-1} s_1 + h_0^2 s_2^2 + h_1^2 s_0^2 + h_2^2 s_{-1}^2 + \\ & r_2^2 - 2r_2 h_0 s_2 - 2r_2 h_1 s_1 - 2r_1 h_2 s_0 + 2h_0 h_1 s_1 s_2 + \\ & 2h_1 h_2 s_0 s_1 + 2h_0 h_2 s_0 s_2 + h_0^2 s_1^2 + h_1^2 s_1^2 + h_2^2 s_0^2 + \\ & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ & r_N^2 - 2r_N h_0 s_N - 2r_N h_1 s_{N-1} - 2r_N h_2 s_{N-2} + 2h_0 h_1 s_{N-1} s_N \\ & + 2h_1 h_2 s_{N-2} s_{N-1} + 2h_0 h_2 s_{N-2} s_N + h_0^2 s_N^2 + h_1^2 s_{N-1}^2 + h_2^2 s_{N-2}^2 \end{aligned} \tag{4.10}$$

which can be written as

$$\begin{aligned}\mathcal{L} &= \sum_{k=1}^{N} r_k^2 - 2\sum_{k=1}^{N}\sum_{l=0}^{2} r_k h_l s_{k-l} + 2\sum_{k=1}^{N} h_0 h_1 s_{k-1} s_k + \\ & 2\sum_{k=1}^{N} h_1 h_2 s_{k-2} s_{k-1} + 2\sum_{k=1}^{N} h_0 h_2 s_{k-2} s_k + \sum_{k=1}^{N}\sum_{l=0}^{2} h_l^2 s_{k-l}^2.\end{aligned} \tag{4.11}$$

Again, by ignoring the quadratic terms, (4.11) can be written as

$$\begin{aligned}\mathcal{L} &= 2\sum_{k=1}^{N} h_0 h_1 s_{k-1} s_k + 2\sum_{k=1}^{N} h_1 h_2 s_{k-2} s_{k-1} + 2\sum_{k=1}^{N} h_0 h_2 s_{k-2} s_k \\ & -2\sum_{k=1}^{N}\sum_{l=0}^{2} r_k h_l s_{k-l}.\end{aligned} \tag{4.12}$$

which again is in the form of (4.3). By comparing (4.12) to (4.3), term by term, $\mathbf{T}$ and $\mathbf{I}$ are determined as

$$\mathbf{T} = -4\begin{bmatrix} 0 & \alpha_1 & \alpha_2 & 0 & \dots & 0 & 0 \\ \alpha_1 & 0 & \alpha_1 & \alpha_2 & \ddots & \ddots & 0 \\ \alpha_2 & \alpha_1 & 0 & \ddots & \dots & \ddots & \vdots \\ 0 & \alpha_2 & \ddots & \ddots & \ddots & \alpha_2 & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 & \alpha_1 & \alpha_2 \\ 0 & \ddots & \ddots & \alpha_2 & \alpha_1 & 0 & \alpha_1 \\ 0 & 0 & \dots & 0 & \alpha_2 & \alpha_1 & 0 \end{bmatrix} \tag{4.13}$$

where again $\mathbf{T}$ is an $N \times N$ matrix. Here, for $L = 3$, there are two non-zero elements in $\mathbf{T}$, determined by

$$\begin{aligned}\alpha_1 &= h_0 h_1 + h_1 h_2 \\ \alpha_2 &= h_0 h_2\end{aligned} \tag{4.14}$$

$$\mathbf{I} = 4\begin{bmatrix} r_1 h_0 + r_2 h_1 + r_3 h_2 - (\alpha_1 + \alpha_2) \\ r_2 h_0 + r_3 h_1 + r_4 h_2 - \alpha_2 \\ r_3 h_0 + r_4 h_1 + r_5 h_2 \\ r_4 h_0 + r_5 h_1 + r_6 h_2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ r_{N-3} h_0 + r_{N-2} h_1 + r_{N-1} h_2 \\ r_{N-2} h_0 + r_{N-1} h_1 + r_N h_2 \\ r_{N-1} h_0 + r_N h_1 + r_{N+1} h_2 - \alpha_2 \\ r_N h_0 + r_{N+1} h_1 + r_{N+2} h_2 - (\alpha_1 + \alpha_2) \end{bmatrix} \tag{4.15}$$

where again $\mathbf{I}$ is a column vector of size N.

This concludes the derivation for systems transmitting N symbols through a channel with a CIR length of $L = 3$. It is already clear that a pattern emerges in $\mathbf{T}$ and $\mathbf{I}$ as L is increased. However, to confirm this observation, the derivation for $L = 4$ is in order.

4.2.1.3 $L = 4$

By expanding (4.2) for a data block length N and CIR length $L = 4$,

$$\begin{aligned}\mathcal{L} \;=\; & r_1^2 - 2r_1h_0s_1 - 2r_1h_1s_0 - 2r_1h_2s_{-1} - 2r_1h_3s_{-2} + 2h_0h_1s_0s_1 + \\ & 2h_1h_2s_{-1}s_0 + 2h_2h_3s_{-2}s_{-1} + 2h_0h_2s_{-1}s_1 + 2h_1h_3s_{-2}s_0 + 2h_0h_3s_{-3}s_0 + \\ & r_2^2 - 2r_2h_0s_2 - 2r_2h_1s_1 - 2r_2h_2s_0 - 2r_2h_3s_{-1} + 2h_0h_1s_1s_2 + \\ & 2h_1h_2s_0s_1 + 2h_2h_3s_{-1}s_0 + 2h_0h_2s_0s_2 + 2h_1h_3s_{-1}s_1 + 2h_0h_3s_{-2}s_1 + \\ & \;\;\vdots \quad\; \vdots \quad\; \vdots \quad\; \vdots \quad\; \vdots \quad\; \vdots \\ & r_N^2 - 2r_Nh_0s_N - 2r_Nh_1s_{N-1} - 2r_Nh_2s_{N-2} - 2r_Nh_3s_{N-2} + 2h_0h_1s_{N-1}s_N + \\ & 2h_1h_2s_{N-2}s_{N-1} + 2h_2h_3s_{N-3}s_{N-2} + 2h_0h_2s_{N-2}s_N + 2h_1h_3s_{N-3}s_{N-1} + 2h_0h_3s_{N-4}s_N\end{aligned} \tag{4.16}$$

which can be written as

$$\begin{aligned}\mathcal{L} \;=\; & \sum_{k=1}^{N} r_k^2 + 2\sum_{k=1}^{N} h_0h_1s_{k-1}s_k + 2\sum_{k=1}^{N} h_1h_2s_{k-2}s_{k-1} \\ & +2\sum_{k=1}^{N} h_0h_2s_{k-2}s_k + 2\sum_{k=1}^{N} h_1h_3s_{k-3}s_{k-1} + 2\sum_{k=1}^{N} h_0h_3s_{k-3}s_k \\ & -2\sum_{k=1}^{N}\sum_{l=0}^{2} r_kh_ls_{k-l} + \sum_{k=1}^{N}\sum_{l=0}^{3} h_l^2s_{k-l}^2\end{aligned} \tag{4.17}$$

By ignoring the exponential terms as before, (4.17) can be written as

$$\begin{aligned}\mathcal{L} \;=\; & 2\sum_{k=1}^{N} h_0h_1s_{k-1}s_k + 2\sum_{k=1}^{N} h_1h_2s_{k-2}s_{k-1} + 2\sum_{k=1}^{N} h_0h_2s_{k-2}s_k \\ & +2\sum_{k=1}^{N} h_1h_3s_{k-3}s_{k-1} + 2\sum_{k=1}^{N} h_0h_3s_{k-3}s_k - 2\sum_{k=1}^{N}\sum_{l=0}^{3} r_kh_ls_{k-l},\end{aligned} \tag{4.18}$$

which is in the form of (4.3). As before, by comparing (4.18) to (4.3), $\mathbf{T}$ and $\mathbf{I}$ are determined as

$$\mathbf{T} = -4\begin{bmatrix} 0 & \alpha_1 & \alpha_2 & \alpha_3 & 0 & \ddots & \dots & \dots & 0 & 0 \\ \alpha_1 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & 0 & \ddots & \ddots & \ddots & 0 \\ \alpha_2 & \alpha_1 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \ddots & \ddots & \ddots & \vdots \\ \alpha_3 & \alpha_2 & \alpha_1 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \alpha_3 & \alpha_2 & \ddots & \ddots & \ddots & \alpha_2 & \alpha_3 & 0 & \ddots \\ \ddots & 0 & \alpha_3 & \ddots & \ddots & \ddots & \alpha_1 & \alpha_2 & \alpha_3 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \alpha_1 & 0 & \alpha_1 & \alpha_2 & \alpha_3 \\ \vdots & \ddots & \ddots & \ddots & \alpha_3 & \alpha_2 & \alpha_1 & 0 & \alpha_1 & \alpha_2 \\ 0 & \ddots & \ddots & \ddots & 0 & \alpha_3 & \alpha_2 & \alpha_1 & 0 & \alpha_1 \\ 0 & 0 & \dots & \dots & \ddots & 0 & \alpha_3 & \alpha_2 & \alpha_1 & 0 \end{bmatrix} \tag{4.19}$$

where $\mathbf{T}$ is an $N \times N$ matrix. For $L = 4$, there are three non-zero elements in $\mathbf{T}$, given as

$$\begin{aligned} \alpha_1 &= h_0h_1 + h_1h_2 + h_2h_3 \\ \alpha_2 &= h_0h_2 + h_1h_3 \\ \alpha_3 &= h_0h_3 \end{aligned} \tag{4.20}$$

and

$$\mathbf{I} = 4\begin{bmatrix} r_1h_0 + r_2h_1 + r_3h_2 + r_4h_3 - (\alpha_1 + \alpha_2 + \alpha_3) \\ r_2h_0 + r_3h_1 + r_4h_2 + r_5h_3 - (\alpha_2 + \alpha_3) \\ r_3h_0 + r_4h_1 + r_5h_2 + r_6h_3 - \alpha_3 \\ r_4h_0 + r_5h_1 + r_6h_2 + r_7h_3 \\ r_5h_0 + r_6h_1 + r_7h_2 + r_8h_3 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ r_{N-4}h_0 + r_{N-3}h_1 + r_{N-2}h_2 + r_{N-1}h_3 \\ r_{N-3}h_0 + r_{N-2}h_1 + r_{N-1}h_2 + r_Nh_3 \\ r_{N-2}h_0 + r_{N-1}h_1 + r_Nh_1 + r_{N+1}h_3 - \alpha_3 \\ r_{N-1}h_0 + r_Nh_1 + r_{N+1}h_1 + r_{N+2}h_3 - (\alpha_2 + \alpha_3) \\ r_Nh_0 + r_{N+1}h_1 + r_{N+2}h_1 + r_{N+3}h_3 - (\alpha_1 + \alpha_2 + \alpha_3) \end{bmatrix} \tag{4.21}$$

where $\mathbf{I}$ is a column vector of size N. By exploiting the patterns that appear in $\mathbf{T}$ and $\mathbf{I}$ for $L = 2$, $L = 3$ and $L = 4$, a generalized model for a BPSK HNN MLSE equalizer can be derived.

4.2.1.4 A General Model for the BPSK HNN MLSE Equalizer

By comparing $\mathbf{T}$ and $\mathbf{I}$ for $L = 2$, $L = 3$ and $L = 4$ respectively, the following observations are made:

1. The number of diagonally symmetric bands in $\mathbf{T}$ increases.
   (a) For $L = 2$ there is only one symmetric band around the diagonal.
   (b) For $L = 3$ there are two symmetric bands around the diagonal.
   (c) For $L = 4$ there are three symmetric bands around the diagonal.

   Rule 1: For $L = l$ there are $l - 1$ symmetric bands around the diagonal.

2. The number of distinct α-values in $\mathbf{T}$ and $\mathbf{I}$ increases.
   (a) For $L = 2$, there is one α-value. α_1 contains only one term. This term is the summation of the product of adjacent CIR coefficients.
   (b) For $L = 3$, there are two α-values.
      i. α_1 is the summation of the product of adjacent CIR coefficients.
      ii. α_2 is the summation of the product of every second CIR coefficient.
   (c) For $L = 4$, there are three α-values.
      i. α_1 is the summation of the product of the adjacent CIR coefficients.
      ii. α_2 is the summation of the product of every second CIR coefficient.
      iii. α_3 is the summation of the product of every third CIR coefficient.

   Rule 2: For $L = l$, there are $l - 1$ α-values. α_n is the summation of the product of every nth CIR coefficient, where $n = 1, 2, \ldots, l - 1$.

3. The number of $r_n h_m$ terms in each element of $\mathbf{I}$ increases.
   (a) For $L = 2$, there are two $r_n h_m$ terms.
   (b) For $L = 3$, there are three $r_n h_m$ terms.
   (c) For $L = 4$, there are four $r_n h_m$ terms.

   Rule 3: For $L = l$, there are l $r_n h_m$ terms in each element of $\mathbf{I}$. All are positive.

4. The number of α-values in $\mathbf{I}$ increases.
   (a) For $L = 2$ there is one α-value in the first and the last elements of $\mathbf{I}$ (α_1).
   (b) For $L = 3$ there are two α-values in the first and the last elements of $\mathbf{I}$ (α_1 and α_2), and one α-value in the second and the second-to-last element of $\mathbf{I}$ (α_2).
   (c) For $L = 4$ there are three α-values in the first and last elements of $\mathbf{I}$ (α_1, α_2 and α_3), two α-values in the second and second-to-last element of $\mathbf{I}$ (α_2 and α_3) and one α-value in the third and third last elements of $\mathbf{I}$ (α_3).

   Rule 4: For $L = l$ there are $l - 1$ α-values in the first and last elements of $\mathbf{I}$ (α_1 to α_{l-1}) and $l - n$ α-values in the nth and nth last element of $\mathbf{I}$ (α_n to α_{l-1}).

By examining the energy functions derived for $L = 2$, $L = 3$ and $L = 4$ in (4.6), (4.12) and (4.18), and by following the rules derived above, a general model for the BPSK HNN MLSE equalizer is derived. For a BPSK system with a data block length of N, transmitting information through a multipath fading channel with a CIR length of L, where $L-1$ known symbols are added to both ends of the data block, $\mathbf{T}$ and $\mathbf{I}$ are respectively determined by

$$\mathbf{T} = -4\begin{bmatrix} 0 & \alpha_1 & \alpha_2 & \dots & \alpha_{L-1} & \dots & 0 \\ \alpha_1 & 0 & \alpha_1 & \alpha_2 & \dots & \alpha_{L-1} & \vdots \\ \alpha_2 & \alpha_1 & 0 & \alpha_1 & \ddots & \vdots & \alpha_{L-1} \\ \vdots & \alpha_2 & \alpha_1 & \ddots & \ddots & \alpha_2 & \vdots \\ \alpha_{L-1} & \vdots & \ddots & \ddots & 0 & \alpha_1 & \alpha_2 \\ \vdots & \alpha_{L-1} & \dots & \alpha_2 & \alpha_1 & 0 & \alpha_1 \\ 0 & \dots & \alpha_{L-1} & \dots & \alpha_2 & \alpha_1 & 0 \end{bmatrix} \tag{4.22}$$

and

$$\mathbf{I} = 4\begin{bmatrix} r_1h_0 + \ldots + r_Lh_{L-1} - \alpha_1 - \alpha_2 - \ldots - \alpha_{L-1} \\ r_2h_0 + \ldots + r_{L+1}h_{L-1} - \alpha_2 - \ldots - \alpha_{L-1} \\ r_3h_0 + \ldots + r_{L+2}h_{L-1} - \ldots - \alpha_{L-1} \\ \vdots \\ r_{L-1}h_0 + \ldots + r_{2L-2}h_{L-1} - \alpha_{L-1} \\ r_Lh_0 + \ldots + r_{2L-1}h_{L-1} \\ \vdots \\ r_{N-L+1}h_0 + \ldots + r_Nh_{L-1} \\ r_{N-L+2}h_0 + \ldots + r_{N+1}h_{L-1} - \alpha_{L-1} \\ \vdots \\ r_{N-2}h_0 + \ldots + r_{N+L-3}h_{L-1} - \ldots - \alpha_{L-1} \\ r_{N-1}h_0 + \ldots + r_{N+L-2}h_{L-1} - \alpha_2 - \ldots - \alpha_{L-1} \\ r_Nh_0 + \ldots + r_{N+L-1}h_{L-1} - \alpha_1 - \alpha_2 - \ldots - \alpha_{L-1} \end{bmatrix}, \tag{4.23}$$

where $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_{L-1}\}$ is determined by

$$\alpha_k = \sum_{j=0}^{L-k-1} h_j h_{j+k}. \tag{4.24}$$

This model can now be used to perform MLSE equalization in BPSK systems with a data block length of N and CIR length of L. The next step is to derive a similar model for M-QAM modulation.

4.2.2 The M-QAM HNN MLSE equalizer

The derivation of a model for the BPSK equalizer was trivial, since (4.2) could be mapped to (4.3) exactly. However, when the equalization of M-QAM modulated signals is considered, the elements in the received symbol vector $\mathbf{r}$, the estimated CIR $\mathbf{h}$ and the estimated symbols $\mathbf{s}$ are complex. This poses a problem in that (4.2) cannot be directly mapped to (4.3). By retaining the same general form, (4.3) must be modified to account for the complex terms.

In a communication system that employs M-QAM modulation, transmitting N complex payload symbols in a multipath fading environment with a complex CIR of length L, (4.2) is minimized in order to produce the transmitted sequence estimates with maximum confidence. For convenience and for the purpose of derivation of the M-QAM HNN MLSE equalizer, (4.2) is rewritten as

$$\mathcal{L} = \sum_{k=1}^{N} \left| \left(r_k^{(i)} + j r_k^{(q)} \right) - \sum_{l=0}^{L-1} \left(h_l^{(i)} + j h_l^{(q)} \right) \left(s_{k-l}^{(i)} + j s_{k-l}^{(q)} \right) \right|^2 \tag{4.25}$$

to explicitly show the in-phase and quadrature components. Here $r_k^{(i)}$ and $r_k^{(q)}$ denote the in-phase and quadrature components of the kth received symbol, $h_l^{(i)}$ and $h_l^{(q)}$ denote the in-phase and quadrature components of the lth estimated CIR coefficient, and $s_k^{(i)}$ and $s_k^{(q)}$ denote the in-phase and quadrature components of the kth estimated symbol.

For convenience (4.3) is presented here again:

$$\mathcal{L} = -\frac{1}{2}\mathbf{s}^{\dagger}\mathbf{T}\mathbf{s} - \mathbf{I}^{\dagger}\mathbf{s}. \tag{4.26}$$

Assuming that $\mathbf{s}$, $\mathbf{I}$ and $\mathbf{T}$ contain complex values, substituting $\mathbf{T}$ with $\mathbf{X}$, these variables can be written as [29, 30, 31, 32]

$$\begin{aligned} \mathbf{s} &= \mathbf{s}_i + j\mathbf{s}_q, \\ \mathbf{I} &= \mathbf{I}_i + j\mathbf{I}_q, \\ \mathbf{X} &= \mathbf{X}_i + j\mathbf{X}_q, \end{aligned} \tag{4.27}$$

where $\mathbf{s}$ and $\mathbf{I}$ are column vectors of length N, and $\mathbf{X}$ is an $N \times N$ matrix, where subscripts i and q are used to denote the respective in-phase and quadrature components. $\mathbf{X}$ is a correlation matrix for complex symbols, implying that it is Hermitian. That is,

$$\mathbf{X}^H = \mathbf{X}_i^T - j\mathbf{X}_q^T = \mathbf{X}_i + j\mathbf{X}_q, \tag{4.28}$$

hence $\mathbf{X}_i^T = \mathbf{X}_i$ is symmetric and $\mathbf{X}_q^T = -\mathbf{X}_q$ is skew symmetric [29, 30]. By using the symmetric properties of $\mathbf{X}$, (4.26) can be expanded and rewritten as

$$\mathcal{L} = -\frac{1}{2}[\mathbf{s}_i^T\mathbf{X}_i\mathbf{s}_i + \mathbf{s}_q^T\mathbf{X}_q\mathbf{s}_q + 2\mathbf{s}_q^T\mathbf{X}_q\mathbf{s}_i] - [\mathbf{s}_i^T\mathbf{I}_i + \mathbf{s}_q^T\mathbf{I}_q]$$

which in turn can be rewritten as [29, 30, 31, 32]

$$\mathcal{L} = -\frac{1}{2}[\mathbf{s}_i^T|\mathbf{s}_q^T]\begin{bmatrix}\mathbf{X}_i & \mathbf{X}_q^T\\ \mathbf{X}_q & \mathbf{X}_i\end{bmatrix}\begin{bmatrix}\mathbf{s}_i\\ \mathbf{s}_q\end{bmatrix} - [\mathbf{I}_i^T|\mathbf{I}_q^T]\begin{bmatrix}\mathbf{s}_i\\ \mathbf{s}_q\end{bmatrix}. \tag{4.29}$$

It is clear that (4.29) is in the form of (4.26), where the variables in (4.26) are substituted as follows:

$$\begin{aligned}\mathbf{s}^\dagger &= [\mathbf{s}_i^T|\mathbf{s}_q^T],\\ \mathbf{I}^\dagger &= [\mathbf{I}_i^T|\mathbf{I}_q^T],\\ \mathbf{X} &= \begin{bmatrix}\mathbf{X}_i & \mathbf{X}_q^T\\ \mathbf{X}_q & \mathbf{X}_i\end{bmatrix}.\end{aligned} \tag{4.30}$$

Equation (4.29) will be used to derive a general model for M-QAM equalization. As for the derivation of the M-QAM model, we derive the model for a system with a data block length of N and increasing CIR lengths starting at $L = 2$. Again it is assumed that $L-1$ tail symbols are added to the ends of the block of N payload symbols such that s_{1-L} to s_0 and s_N to s_{N+L-1} are equal to $(1/\sqrt{2}) + j(1/\sqrt{2})$.

4.2.2.1 $L = 2$

Here we follow the same procedure as before. To derive the model for a M-QAM equalizer with a data block length of N and a CIR length of $L = 2$, (4.25) is written as

$$\begin{aligned}\mathcal{L} = \sum_{k=1}^{N}\Big|&\left(r_k^{(i)} + jr_k^{(q)}\right) - \left(h_0^{(i)} + jh_0^{(q)}\right)\left(s_1^{(i)} + js_1^{(q)}\right) -\\ &\left(h_1^{(i)} + jh_1^{(q)}\right)\left(s_0^{(i)} + js_0^{(q)}\right)\Big|^2.\end{aligned} \tag{4.31}$$

Before expanding (4.31), the real and imaginary components must be separated. Therefore, by multiplying the terms in brackets and separating the real and imaginary terms we get

$$\mathcal{L} = \sum_{k=1}^{N}\Big|\left(r_k^{(i)} - s_k^{(i)}h_0^{(i)} + s_k^{(q)}h_0^{(q)} - s_{k-1}^{(i)}h_1^{(i)} + s_{k-1}^{(q)}h_1^{(q)}\right) + \tag{4.32}$$

$$j\left(r_k^{(q)} - s_k^{(i)}h_0^{(q)} - s_k^{(q)}h_0^{(i)} - s_{k-1}^{(i)}h_1^{(q)} - s_{k-1}^{(q)}h_1^{(i)}\right)\Big|^2. \tag{4.33}$$

By expanding (4.32) and ignoring the exponential terms as before, we get

$$\begin{aligned}
\mathcal{L} \;=\; & -2r_1^{(i)}h_0^{(i)}s_1^{(i)} - 2r_1^{(i)}h_1^{(i)}s_0^{(i)} + 2r_1^{(i)}h_0^{(q)}s_1^{(q)} + 2r_1^{(q)}h_1^{(q)}s_0^{(i)} \\
& -2r_1^{(q)}h_0^{(q)}s_1^{(i)} - 2r_1^{(q)}h_1^{(q)}s_0^{(i)} - 2r_1^{(q)}h_0^{(i)}s_1^{(q)} - 2r_1^{(q)}h_1^{(i)}s_0^{(q)} \\
& -2s_1^{(i)}s_1^{(q)}h_0^{(i)}h_0^{(q)} - 2s_0^{(i)}s_1^{(q)}h_1^{(i)}h_0^{(q)} - 2s_0^{(i)}s_0^{(q)}h_1^{(i)}h_1^{(q)} \\
& +2s_0^{(i)}s_1^{(i)}h_0^{(i)}h_1^{(i)} + 2s_0^{(q)}s_1^{(q)}h_0^{(q)}h_1^{(q)} + 2s_0^{(q)}s_1^{(i)}h_0^{(i)}h_1^{(q)} \\
& +2s_1^{(i)}s_1^{(q)}h_0^{(i)}h_0^{(q)} + 2s_0^{(i)}s_1^{(q)}h_0^{(i)}h_1^{(q)} + 2s_0^{(i)}s_0^{(q)}h_1^{(i)}h_1^{(q)} \\
& +2s_0^{(i)}s_1^{(i)}h_0^{(q)}h_1^{(q)} + 2s_0^{(q)}s_1^{(q)}h_0^{(i)}h_1^{(i)} + 2s_0^{(q)}s_1^{(i)}h_1^{(i)}h_0^{(q)} \\
& \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
& \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
& -2r_N^{(i)}h_0^{(i)}s_N^{(i)} - 2r_N^{(i)}h_1^{(i)}s_{N-1}^{(i)} + 2r_N^{(i)}h_0^{(q)}s_N^{(q)} + 2r_N^{(q)}h_1^{(q)}s_{N-1}^{(i)} \\
& -2r_N^{(q)}h_0^{(q)}s_N^{(i)} - 2r_N^{(q)}h_1^{(q)}s_{N-1}^{(i)} - 2r_N^{(q)}h_0^{(i)}s_N^{(q)} - 2r_N^{(q)}h_1^{(i)}s_{N-1}^{(q)} \\
& -2s_N^{(i)}s_N^{(q)}h_0^{(i)}h_0^{(q)} - 2s_{N-1}^{(i)}s_N^{(q)}h_1^{(i)}h_0^{(q)} - 2s_{N-1}^{(i)}s_{N-1}^{(q)}h_1^{(i)}h_1^{(q)} \\
& +2s_{N-1}^{(i)}s_N^{(i)}h_0^{(i)}h_1^{(i)} + 2s_{N-1}^{(q)}s_N^{(q)}h_0^{(q)}h_1^{(q)} + 2s_{N-1}^{(q)}s_N^{(i)}h_0^{(i)}h_1^{(q)} \\
& +2s_N^{(i)}s_N^{(q)}h_0^{(i)}h_0^{(q)} + 2s_{N-1}^{(i)}s_N^{(q)}h_0^{(i)}h_1^{(q)} + 2s_{N-1}^{(i)}s_{N-1}^{(q)}h_1^{(i)}h_1^{(q)} \\
& +2s_{N-1}^{(i)}s_N^{(i)}h_0^{(q)}h_1^{(q)} + 2s_{N-1}^{(q)}s_N^{(q)}h_0^{(i)}h_1^{(i)} + 2s_{N-1}^{(q)}s_N^{(i)}h_1^{(i)}h_0^{(q)}
\end{aligned} \tag{4.34}$$

Since is (4.34) is very long and almost illegible, the in-phase and quadrature components of the estimated symbol vector are factorized. The result is

$$\begin{aligned}
\mathcal{L} \;=\; & -2s_0^{(i)}\left(r_1^{(i)}h_1^{(i)} + r_1^{(q)}h_1^{(q)}\right) - 2s_1^{(i)}\left(r_1^{(i)}h_0^{(i)} + r_1^{(q)}h_0^{(q)}\right) \\
& -2s_0^{(q)}\left(r_1^{(q)}h_1^{(i)} + r_1^{(i)}h_1^{(q)}\right) - 2s_1^{(q)}\left(r_1^{(q)}h_0^{(i)} + r_1^{(i)}h_0^{(q)}\right) \\
& +2s_0^{(i)}s_1^{(i)}\left(h_0^{(i)}h_1^{(i)} + h_0^{(q)}h_1^{(q)}\right) + 2s_0^{(q)}s_1^{(q)}\left(h_0^{(i)}h_1^{(i)} + h_0^{(i)}h_1^{(i)}\right) \\
& -2s_0^{(i)}s_1^{(q)}\left(h_0^{(q)}h_1^{(i)} + h_0^{(i)}h_1^{(q)}\right) - 2s_0^{(q)}s_1^{(i)}\left(h_0^{(i)}h_1^{(q)} + h_0^{(q)}h_1^{(i)}\right) \\
& \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
& \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
& -2s_{N-1}^{(i)}\left(r_N^{(i)}h_1^{(i)} + r_N^{(q)}h_1^{(q)}\right) - 2s_N^{(i)}\left(r_N^{(i)}h_0^{(i)} + r_N^{(q)}h_0^{(q)}\right) \\
& -2s_{N-1}^{(q)}\left(r_N^{(q)}h_1^{(i)} + r_N^{(i)}h_1^{(q)}\right) - 2s_N^{(q)}\left(r_N^{(q)}h_0^{(i)} + r_N^{(i)}h_0^{(q)}\right) \\
& +2s_{N-1}^{(i)}s_N^{(i)}\left(h_0^{(i)}h_1^{(i)} + h_0^{(q)}h_1^{(q)}\right) + 2s_{N-1}^{(q)}s_N^{(q)}\left(h_0^{(i)}h_1^{(i)} + h_0^{(i)}h_1^{(i)}\right) \\
& -2s_{N-1}^{(i)}s_N^{(q)}\left(h_0^{(q)}h_1^{(i)} + h_0^{(i)}h_1^{(q)}\right) - 2s_{N-1}^{(q)}s_N^{(i)}\left(h_0^{(i)}h_1^{(q)} + h_0^{(q)}h_1^{(i)}\right)
\end{aligned} \tag{4.35}$$

which can be written as

$$\begin{aligned}\mathcal{L} \;=\; & -2\sum_{k=1}^{N}\sum_{l=0}^{1} s_{k-l}^{(i)}\left(r_k^{(i)}h_{k+l-1}^{(i)} + r_k^{(q)}h_{k+l-1}^{(q)}\right) \\ & -2\sum_{k=1}^{N}\sum_{l=0}^{1} s_{k-l}^{(q)}\left(r_k^{(q)}h_{k+l-1}^{(i)} + r_k^{(i)}h_{k+l-1}^{(q)}\right) \\ & +2\sum_{k=1}^{N}\left(s_{k-1}^{(i)}s_k^{(i)}\left(h_0^{(i)}h_1^{(i)} + h_0^{(q)}h_1^{(q)}\right) + s_{k-1}^{(q)}s_k^{(q)}\left(h_0^{(i)}h_1^{(i)} + h_0^{(i)}h_1^{(i)}\right)\right) \\ & -2\sum_{k=1}^{N}\left(s_{k-1}^{(i)}s_k^{(q)}\left(h_0^{(q)}h_1^{(i)} + h_0^{(i)}h_1^{(q)}\right) - s_{k-1}^{(q)}s_k^{(i)}\left(h_0^{(i)}h_1^{(q)} + h_0^{(q)}h_1^{(i)}\right)\right).\end{aligned}$$

Although not apparent at first sight, (4.37) is in the form of (4.29). All unknown variables in (4.29) can therefore be derived by inspection. For $L = 2$ the variables are determined by

$$\mathbf{T}_i = -4\begin{bmatrix} 0 & \alpha_1 & 0 & \dots & 0 & 0 \\ \alpha_1 & 0 & \alpha_1 & 0 & \ddots & 0 \\ 0 & \alpha_1 & 0 & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \alpha_1 & 0 \\ 0 & \ddots & \ddots & \alpha_1 & 0 & \alpha_1 \\ 0 & 0 & \dots & 0 & \alpha_1 & 0 \end{bmatrix}, \tag{4.37}$$

where $\mathbf{T}_i$ is an $N \times N$ matrix and

$$\alpha_1 = h_0^{(i)}h_1^{(i)} + h_0^{(q)}h_1^{(q)}, \tag{4.38}$$

$$\mathbf{T} = -4\begin{bmatrix} 0 & \gamma_1 & 0 & \dots & 0 & 0 \\ \gamma_1 & 0 & \gamma_1 & 0 & \ddots & 0 \\ 0 & \gamma_1 & 0 & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \gamma_1 & 0 \\ 0 & \ddots & \ddots & \gamma_1 & 0 & \gamma_1 \\ 0 & 0 & \dots & 0 & \gamma_1 & 0 \end{bmatrix}, \tag{4.39}$$

where $\mathbf{T}_q$ is also an $N \times N$ matrix and

$$\gamma_1 = h_0^{(q)}h_1^{(i)} - h_0^{(i)}h_1^{(q)}, \tag{4.40}$$

$$\mathbf{I}_i = 4\begin{bmatrix} \left(r_1^{(i)}h_0^{(i)} + r_2^{(i)}h_1^{(i)}\right) + \left(r_1^{(q)}h_0^{(q)} + r_2^{(q)}h_1^{(q)}\right) - \frac{1}{\sqrt{2}}(\alpha_1 + \gamma_1) \\ \left(r_2^{(i)}h_0^{(i)} + r_3^{(i)}h_1^{(i)}\right) + \left(r_2^{(q)}h_0^{(q)} + r_3^{(q)}h_1^{(q)}\right) \\ \left(r_3^{(i)}h_0^{(i)} + r_4^{(i)}h_1^{(i)}\right) + \left(r_3^{(q)}h_0^{(q)} + r_4^{(q)}h_1^{(q)}\right) \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \left(r_{N-2}^{(i)}h_0^{(i)} + r_{N-1}^{(i)}h_1^{(i)}\right) + \left(r_{N-2}^{(q)}h_0^{(q)} + r_{N-1}^{(q)}h_1^{(q)}\right) \\ \left(r_{N-1}^{(i)}h_0^{(i)} + r_N^{(i)}h_1^{(i)}\right) + \left(r_{N-1}^{(q)}h_0^{(q)} + r_N^{(q)}h_1^{(q)}\right) \\ \left(r_N^{(i)}h_0^{(i)} + r_{N+1}^{(i)}h_1^{(i)}\right) + \left(r_N^{(q)}h_0^{(q)} + r_{N+1}^{(q)}h_1^{(q)}\right) - \frac{1}{\sqrt{2}}(\alpha_1 - \gamma_1) \end{bmatrix} \tag{4.41}$$

and

$$\mathbf{I}_q = 4\begin{bmatrix} \left(r_1^{(q)}h_0^{(i)} - r_2^{(q)}h_1^{(i)}\right) + \left(r_1^{(i)}h_0^{(q)} - r_2^{(i)}h_1^{(q)}\right) - \frac{1}{\sqrt{2}}(\alpha_1 - \gamma_1) \\ \left(r_2^{(q)}h_0^{(i)} - r_3^{(q)}h_1^{(i)}\right) + \left(r_2^{(i)}h_0^{(q)} - r_3^{(i)}h_1^{(q)}\right) \\ \left(r_3^{(q)}h_0^{(i)} - r_4^{(q)}h_1^{(i)}\right) + \left(r_3^{(i)}h_0^{(q)} - r_4^{(i)}h_1^{(q)}\right) \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \left(r_{N-2}^{(q)}h_0^{(i)} - r_{N-1}^{(q)}h_1^{(i)}\right) + \left(r_{N-2}^{(i)}h_0^{(q)} - r_{N-1}^{(i)}h_1^{(q)}\right) \\ \left(r_{N-1}^{(q)}h_0^{(i)} - r_N^{(q)}h_1^{(i)}\right) + \left(r_{N-1}^{(i)}h_0^{(q)} - r_N^{(i)}h_1^{(q)}\right) \\ \left(r_N^{(q)}h_0^{(i)} - r_{N+1}^{(q)}h_1^{(i)}\right) + \left(r_N^{(i)}h_0^{(q)} - r_{N+1}^{(i)}h_1^{(q)}\right) - \frac{1}{\sqrt{2}}1(\alpha_1 + \gamma_1) \end{bmatrix}, \tag{4.42}$$

where $\mathbf{I}_i$ and $\mathbf{I}_q$ are both column vectors with N elements.

It is already apparent that the model for the M-QAM equalizer exibits the same patterns in $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$ as the BPSK equalizer model developed in *Section 4.2.1* exhibited in $\mathbf{T}$ and $\mathbf{I}$. To confirm this observation the model for the M-QAM equalizer with a data block length of N and CIR length $L = 3$ must be derived.

4.2.2.2 $L = 3$

To derive the model for a M-QAM equalizer with a data block length of N and a CIR length of $L = 2$, (4.25) is written as

$$\begin{aligned} \mathcal{L} \;=\; & \sum_{k=1}^{N} \Big| \left(r_k^{(i)} + jr_k^{(q)}\right) - \left(h_0^{(i)} + jh_0^{(q)}\right)\left(s_1^{(i)} + js_1^{(q)}\right) - \\ & \left(h_1^{(i)} + jh_1^{(q)}\right)\left(s_0^{(i)} + js_0^{(q)}\right) - \left(h_2^{(i)} + jh_2^{(q)}\right)\left(s_{-1}^{(i)} + js_{-1}^{(q)}\right) \Big|^2. \end{aligned} \tag{4.43}$$

As before, the real and imaginary components are separated. Therefore we get

$$\mathcal{L} = \sum_{k=1}^{N} \Big| \Big(r_k^{(i)} - s_k^{(i)} h_0^{(i)} + s_k^{(q)} h_0^{(q)} - s_{k-1}^{(i)} h_1^{(i)} + s_{k-1}^{(q)} h_1^{(q)} - s_{k-2}^{(i)} h_2^{(i)} + s_{k-2}^{(q)} h_2^{(q)} \Big) \\ + j \Big(r_k^{(q)} - s_k^{(i)} h_0^{(q)} - s_k^{(q)} h_0^{(i)} - s_{k-1}^{(i)} h_1^{(q)} - s_{k-1}^{(q)} h_1^{(i)} - s_{k-2}^{(i)} h_2^{(q)} - s_{k-2}^{(q)} h_2^{(i)} \Big) \Big|^2. \tag{4.44}$$

By expanding (4.44) and ignoring the exponential terms as before, we get

$$\begin{aligned}
\mathcal{L} \;=\; & -2r_1^{(i)} h_0^{(i)} s_1^{(i)} + 2r_1^{(i)} h_0^{(q)} s_1^{(q)} - 2r_1^{(i)} h_1^{(i)} s_0^{(i)} + 2r_1^{(i)} h_1^{(q)} s_0^{(q)} - 2r_1^{(i)} h_2^{(i)} s_{-1}^{(i)} \\
& + 2r_1^{(i)} h_2^{(q)} s_{-1}^{(q)} - 2s_1^{(i)} s_1^{(q)} h_0^{(i)} h_0^{(q)} - 2s_1^{(q)} s_0^{(i)} h_0^{(q)} h_1^{(i)} - 2s_0^{(i)} s_0^{(q)} h_1^{(i)} h_1^{(q)} \\
& - 2s_0^{(q)} s_{-1}^{(i)} h_1^{(q)} h_2^{(i)} - 2s_{-1}^{(i)} s_{-1}^{(q)} h_2^{(i)} h_2^{(q)} + 2s_1^{(i)} s_0^{(i)} h_0^{(i)} h_1^{(i)} + 2s_1^{(q)} s_0^{(q)} h_0^{(q)} h_1^{(q)} \\
& + 2s_0^{(i)} s_{-1}^{(i)} h_0^{(i)} h_2^{(i)} - 2s_0^{(q)} s_{-1}^{(q)} h_0^{(q)} h_2^{(q)} - 2s_1^{(i)} s_0^{(q)} h_0^{(i)} h_1^{(q)} - 2s_1^{(q)} s_{-1}^{(i)} h_0^{(q)} h_2^{(i)} \\
& - 2s_0^{(i)} s_{-1}^{(q)} h_1^{(i)} h_2^{(q)} + 2s_1^{(i)} s_{-1}^{(i)} h_0^{(i)} h_2^{(i)} + 2s_1^{(q)} s_{-1}^{(q)} h_0^{(q)} h_2^{(q)} - 2s_1^{(i)} s_{-1}^{(q)} h_0^{(i)} h_2^{(q)} \\
& - 2r_1^{(q)} h_0^{(q)} s_1^{(i)} + 2r_1^{(q)} h_0^{(i)} s_1^{(q)} - 2r_1^{(q)} h_1^{(q)} s_0^{(i)} + 2r_1^{(q)} h_1^{(i)} s_0^{(q)} - 2r_1^{(q)} h_2^{(q)} s_{-1}^{(i)} \\
& + 2r_1^{(q)} h_2^{(i)} s_{-1}^{(q)} - 2s_1^{(i)} s_1^{(q)} h_0^{(q)} h_0^{(i)} - 2s_1^{(q)} s_0^{(i)} h_0^{(i)} h_1^{(q)} - 2s_0^{(i)} s_0^{(q)} h_1^{(q)} h_1^{(i)} \\
& - 2s_0^{(q)} s_{-1}^{(i)} h_1^{(i)} h_2^{(q)} - 2s_{-1}^{(i)} s_{-1}^{(q)} h_2^{(q)} h_2^{(i)} + 2s_1^{(i)} s_0^{(q)} h_0^{(q)} h_1^{(i)} + 2s_1^{(q)} s_0^{(i)} h_0^{(i)} h_1^{(q)} \\
& + 2s_0^{(i)} s_{-1}^{(q)} h_0^{(q)} h_2^{(i)} - 2s_0^{(q)} s_{-1}^{(i)} h_0^{(i)} h_2^{(q)} - 2s_1^{(i)} s_0^{(q)} h_0^{(q)} h_1^{(i)} - 2s_1^{(q)} s_{-1}^{(i)} h_0^{(i)} h_2^{(q)} \\
& - 2s_0^{(i)} s_{-1}^{(q)} h_1^{(q)} h_2^{(i)} + 2s_1^{(i)} s_{-1}^{(i)} h_0^{(q)} h_2^{(q)} + 2s_1^{(q)} s_{-1}^{(i)} h_0^{(q)} h_2^{(i)} - 2s_1^{(i)} s_{-1}^{(q)} h_0^{(q)} h_2^{(i)} \\
& \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
& \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
& - 2r_N^{(i)} h_0^{(i)} s_N^{(i)} + 2r_N^{(i)} h_0^{(q)} s_N^{(q)} - 2r_N^{(i)} h_1^{(i)} s_{N-1}^{(i)} + 2r_N^{(i)} h_1^{(q)} s_{N-1}^{(q)} - 2r_N^{(i)} h_2^{(i)} s_{N-2}^{(i)} \\
& + 2r_N^{(i)} h_2^{(q)} s_{N-2}^{(q)} - 2s_N^{(i)} s_N^{(q)} h_0^{(i)} h_0^{(q)} - 2s_N^{(q)} s_{N-1}^{(i)} h_0^{(q)} h_1^{(i)} - 2s_{N-1}^{(i)} s_{N-1}^{(q)} h_1^{(i)} h_1^{(q)} \\
& - 2s_{N-1}^{(q)} s_{N-2}^{(i)} h_1^{(q)} h_2^{(i)} - 2s_{N-2}^{(i)} s_{N-2}^{(q)} h_2^{(i)} h_2^{(q)} + 2s_N^{(i)} s_{N-1}^{(i)} h_0^{(i)} h_1^{(i)} + 2s_N^{(q)} s_{N-1}^{(q)} h_0^{(q)} h_1^{(q)} \\
& + 2s_{N-1}^{(i)} s_{N-2}^{(i)} h_0^{(i)} h_2^{(i)} - 2s_{N-1}^{(q)} s_{N-2}^{(q)} h_0^{(q)} h_2^{(q)} - 2s_N^{(i)} s_{N-1}^{(q)} h_0^{(i)} h_1^{(q)} - 2s_N^{(q)} s_{N-2}^{(i)} h_0^{(q)} h_2^{(i)} \\
& - 2s_{N-1}^{(i)} s_{N-2}^{(q)} h_1^{(i)} h_2^{(q)} + 2s_N^{(i)} s_{N-2}^{(i)} h_0^{(i)} h_2^{(i)} + 2s_N^{(q)} s_{N-2}^{(q)} h_0^{(q)} h_2^{(q)} - 2s_N^{(i)} s_{N-2}^{(q)} h_0^{(i)} h_2^{(q)} \\
& - 2r_N^{(q)} h_0^{(q)} s_N^{(i)} + 2r_N^{(q)} h_0^{(i)} s_N^{(q)} - 2r_N^{(q)} h_1^{(q)} s_{N-1}^{(i)} + 2r_N^{(q)} h_1^{(i)} s_{N-1}^{(q)} - 2r_N^{(q)} h_2^{(q)} s_{N-2}^{(i)} \\
& + 2r_N^{(q)} h_2^{(i)} s_{N-2}^{(q)} - 2s_N^{(i)} s_N^{(q)} h_0^{(q)} h_0^{(i)} - 2s_N^{(q)} s_{N-1}^{(i)} h_0^{(i)} h_1^{(q)} - 2s_{N-1}^{(i)} s_{N-1}^{(q)} h_1^{(q)} h_1^{(i)} \\
& - 2s_{N-1}^{(q)} s_{N-2}^{(i)} h_1^{(i)} h_2^{(q)} - 2s_{N-2}^{(i)} s_{N-2}^{(q)} h_2^{(q)} h_2^{(i)} + 2s_N^{(i)} s_{N-1}^{(q)} h_0^{(q)} h_1^{(i)} + 2s_N^{(q)} s_{N-1}^{(i)} h_0^{(i)} h_1^{(q)} \\
& + 2s_{N-1}^{(i)} s_{N-2}^{(i)} h_0^{(q)} h_2^{(q)} - 2s_{N-1}^{(q)} s_{N-2}^{(q)} h_0^{(i)} h_2^{(i)} - 2s_N^{(i)} s_{N-1}^{(q)} h_0^{(q)} h_1^{(i)} - 2s_N^{(q)} s_{N-2}^{(i)} h_0^{(i)} h_2^{(q)} \\
& - 2s_{N-1}^{(i)} s_{N-2}^{(q)} h_1^{(q)} h_2^{(i)} + 2s_N^{(i)} s_{N-2}^{(q)} h_0^{(q)} h_2^{(i)} + 2s_N^{(q)} s_{N-2}^{(i)} h_0^{(q)} h_2^{(i)} - 2s_N^{(i)} s_{N-2}^{(q)} h_0^{(q)} h_2^{(i)},
\end{aligned} \tag{4.45}$$

which is extremely involved. In order to perform the mapping to $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$, the $\mathbf{s}$-terms are factorized, resulting in

$$
\begin{aligned}
\mathcal{L} \;=\; & -2s_1^{(i)}\left(r_1^{(i)}h_0^{(i)}+r_1^{(q)}h_0^{(q)}\right)-2s_1^{(q)}\left(r_1^{(q)}h_0^{(i)}-r_1^{(i)}h_0^{(q)}\right)-2s_0^{(i)}\left(r_1^{(i)}h_1^{(i)}+r_1^{(q)}h_1^{(q)}\right) \\
& -2s_0^{(q)}\left(r_1^{(q)}h_1^{(i)}-r_1^{(i)}h_1^{(q)}\right)-2s_{-1}^{(i)}\left(r_1^{(i)}h_2^{(i)}+r_1^{(q)}h_2^{(q)}\right)-2s_{-1}^{(q)}\left(r_1^{(q)}h_2^{(i)}-r_1^{(i)}h_2^{(q)}\right) \\
& -2s_1^{(q)}s_0^{(i)}\left(h_0^{(q)}h_1^{(i)}-h_0^{(i)}h_1^{(q)}\right)-2s_0^{(q)}s_{-1}^{(i)}\left(h_1^{(q)}h_2^{(i)}-h_1^{(i)}h_2^{(q)}\right) \\
& -2s_1^{(i)}s_0^{(i)}\left(h_0^{(i)}h_1^{(i)}+h_0^{(q)}h_1^{(q)}\right)-2s_1^{(q)}s_0^{(q)}\left(h_0^{(i)}h_1^{(i)}+h_0^{(q)}h_1^{(q)}\right) \\
& +2s_0^{(i)}s_{-1}^{(i)}\left(h_0^{(i)}h_2^{(i)}+h_0^{(q)}h_2^{(q)}\right)-2s_0^{(q)}s_{-1}^{(q)}\left(h_0^{(i)}h_2^{(i)}+h_0^{(q)}h_2^{(q)}\right) \\
& -2s_1^{(i)}s_0^{(q)}\left(h_0^{(i)}h_1^{(q)}-h_0^{(q)}h_1^{(i)}\right)-2s_1^{(q)}s_{-1}^{(i)}\left(h_0^{(i)}h_2^{(q)}-h_0^{(q)}h_2^{(i)}\right) \\
& -2s_0^{(i)}s_{-1}^{(q)}\left(h_1^{(i)}h_2^{(q)}-h_1^{(q)}h_2^{(i)}\right)+2s_1^{(i)}s_{-1}^{(i)}\left(h_0^{(i)}h_2^{(i)}+h_0^{(q)}h_2^{(q)}\right) \\
& +2s_1^{(q)}s_{-1}^{(q)}\left(h_0^{(i)}h_2^{(i)}+h_0^{(q)}h_2^{(q)}\right)-2s_1^{(i)}s_{-1}^{(q)}\left(h_0^{(i)}h_2^{(q)}-h_0^{(q)}h_2^{(i)}\right) \\
& \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
& \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
& -2s_N^{(i)}\left(r_N^{(i)}h_0^{(i)}+r_N^{(q)}h_0^{(q)}\right)-2s_N^{(q)}\left(r_N^{(q)}h_0^{(i)}-r_N^{(i)}h_0^{(q)}\right)-2s_{N-1}^{(i)}\left(r_N^{(i)}h_1^{(i)}+r_N^{(q)}h_1^{(q)}\right) \\
& -2s_{N-1}^{(q)}\left(r_N^{(q)}h_1^{(i)}-r_N^{(i)}h_1^{(q)}\right)-2s_{N-2}^{(i)}\left(r_N^{(i)}h_2^{(i)}+r_N^{(q)}h_2^{(q)}\right)-2s_{N-2}^{(q)}\left(r_N^{(q)}h_2^{(i)}-r_N^{(i)}h_2^{(q)}\right) \\
& -2s_N^{(q)}s_{N-1}^{(i)}\left(h_0^{(q)}h_1^{(i)}-h_0^{(i)}h_1^{(q)}\right)-2s_{N-1}^{(q)}s_{N-2}^{(i)}\left(h_1^{(q)}h_2^{(i)}-h_1^{(i)}h_2^{(q)}\right) \\
& -2s_N^{(i)}s_{N-1}^{(i)}\left(h_0^{(i)}h_1^{(i)}+h_0^{(q)}h_1^{(q)}\right)-2s_N^{(q)}s_{N-1}^{(q)}\left(h_0^{(i)}h_1^{(i)}+h_0^{(q)}h_1^{(q)}\right) \\
& +2s_{N-1}^{(i)}s_{N-2}^{(i)}\left(h_0^{(i)}h_2^{(i)}+h_0^{(q)}h_2^{(q)}\right)-2s_{N-1}^{(q)}s_{N-2}^{(q)}\left(h_0^{(i)}h_2^{(i)}+h_0^{(q)}h_2^{(q)}\right) \\
& -2s_N^{(i)}s_{N-1}^{(q)}\left(h_0^{(i)}h_1^{(q)}-h_0^{(q)}h_1^{(i)}\right)-2s_N^{(q)}s_{N-2}^{(i)}\left(h_0^{(i)}h_2^{(q)}-h_0^{(q)}h_2^{(i)}\right) \\
& -2s_{N-1}^{(i)}s_{N-2}^{(q)}\left(h_1^{(i)}h_2^{(q)}-h_1^{(q)}h_2^{(i)}\right)+2s_N^{(i)}s_{N-2}^{(i)}\left(h_0^{(i)}h_2^{(i)}+h_0^{(q)}h_2^{(q)}\right) \\
& +2s_N^{(q)}s_{N-2}^{(q)}\left(h_0^{(i)}h_2^{(i)}+h_0^{(q)}h_2^{(q)}\right)-2s_N^{(i)}s_{N-2}^{(q)}\left(h_0^{(i)}h_2^{(q)}-h_0^{(q)}h_2^{(i)}\right).
\end{aligned}
\tag{4.46}
$$

From (4.46) it is now easy to perform the mapping. By inspection, mapping (4.46) term by term to (4.29), $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$ can be expressed as follows:

$$
\mathbf{T}_i = -4\begin{bmatrix}
0 & \alpha_1 & \alpha_2 & 0 & \dots & 0 & 0 \\
\alpha_1 & 0 & \alpha_1 & \alpha_2 & \ddots & \ddots & 0 \\
\alpha_2 & \alpha_1 & 0 & \ddots & \dots & \ddots & \vdots \\
0 & \alpha_2 & \ddots & \ddots & \ddots & \alpha_2 & 0 \\
\vdots & \ddots & \ddots & \ddots & 0 & \alpha_1 & \alpha_2 \\
0 & \ddots & \ddots & \alpha_2 & \alpha_1 & 0 & \alpha_1 \\
0 & 0 & \dots & 0 & \alpha_2 & \alpha_1 & 0
\end{bmatrix}
\tag{4.47}
$$

and

$$\mathbf{T}_q = -4\begin{bmatrix} 0 & \gamma_1 & \gamma_2 & 0 & \dots & 0 & 0 \\ \gamma_1 & 0 & \gamma_1 & \gamma_2 & \ddots & \ddots & 0 \\ \gamma_2 & \gamma_1 & 0 & \ddots & \dots & \ddots & \vdots \\ 0 & \gamma_2 & \ddots & \ddots & \ddots & \gamma_2 & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 & \gamma_1 & \gamma_2 \\ 0 & \ddots & \ddots & \gamma_2 & \gamma_1 & 0 & \gamma_1 \\ 0 & 0 & \dots & 0 & \gamma_2 & \gamma_1 & 0 \end{bmatrix}, \tag{4.48}$$

where

$$\begin{aligned} \alpha_1 &= \left(h_0^{(i)}h_1^{(i)} + h_0^{(q)}h_1^{(q)}\right) + \left(h_1^{(i)}h_2^{(i)} + h_1^{(q)}h_2^{(q)}\right) \\ \alpha_2 &= \mathrm{h}_0^{(i)}h_2^{(i)} + h_0^{(q)}h_2^{(q)} \end{aligned} \tag{4.49}$$

and

$$\begin{aligned} \gamma_1 &= \left(h_0^{(q)}h_1^{(i)} - h_0^{(i)}h_1^{(q)}\right) + \left(h_1^{(q)}h_2^{(i)} - h_1^{(i)}h_2^{(q)}\right) \\ \gamma_2 &= \mathrm{h}_0^{(q)}h_2^{(i)} - h_0^{(i)}h_2^{(q)}. \end{aligned} \tag{4.50}$$

Also,

$$\mathbf{I}_i = 4\begin{bmatrix} \left(r_1^{(i)}h_0^{(i)} + r_2^{(i)}h_1^{(i)} + r_3^{(i)}h_2^{(i)}\right) + \left(r_1^{(q)}h_0^{(q)} + r_2^{(q)}h_1^{(q)} + r_3^{(q)}h_2^{(q)}\right) - \rho(\alpha_1+\alpha_2+\gamma_1+\gamma_2) \\ \left(r_2^{(i)}h_0^{(i)} + r_3^{(i)}h_1^{(i)} + r_4^{(i)}h_2^{(i)}\right) + \left(r_2^{(q)}h_0^{(q)} + r_3^{(q)}h_1^{(q)} + +r_4^{(q)}h_2^{(q)}\right) - \rho(\alpha_2+\gamma_2) \\ \left(r_3^{(i)}h_0^{(i)} + r_4^{(i)}h_1^{(i)} + r_5^{(i)}h_2^{(i)}\right) + \left(r_3^{(q)}h_0^{(q)} + r_4^{(q)}h_1^{(q)} + r_5^{(q)}h_2^{(q)}\right) \\ \left(r_4^{(i)}h_0^{(i)} + r_5^{(i)}h_1^{(i)} + r_6^{(i)}h_2^{(i)}\right) + \left(r_4^{(q)}h_0^{(q)} + r_5^{(q)}h_1^{(q)} + r_6^{(q)}h_2^{(q)}\right) \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \left(r_{N-3}^{(i)}h_0^{(i)} + r_{N-2}^{(i)}h_1^{(i)} + r_{N-1}^{(i)}h_2^{(i)}\right) + \left(r_{N-3}^{(q)}h_0^{(q)} + r_{N-2}^{(q)}h_1^{(q)} + r_{N-1}^{(q)}h_2^{(q)}\right) \\ \left(r_{N-2}^{(i)}h_0^{(i)} + r_{N-1}^{(i)}h_1^{(i)} + r_N^{(i)}h_2^{(i)}\right) + \left(r_{N-2}^{(q)}h_0^{(q)} + r_{N-1}^{(q)}h_1^{(q)} + r_N^{(q)}h_2^{(q)}\right) \\ \left(r_{N-1}^{(i)}h_0^{(i)} + r_N^{(i)}h_1^{(i)} + r_{N+1}^{(i)}h_2^{(i)}\right) + \left(r_{N-1}^{(q)}h_0^{(q)} + r_N^{(q)}h_1^{(q)} + +r_{N+1}^{(q)}h_2^{(q)}\right) - \rho(\alpha_2-\gamma_2) \\ \left(r_N^{(i)}h_0^{(i)} + r_{N+1}^{(i)}h_1^{(i)} + r_{N+2}^{(i)}h_2^{(i)}\right) + \left(r_N^{(q)}h_0^{(q)} + r_{N+1}^{(q)}h_1^{(q)} + r_{N+2}^{(q)}h_2^{(q)}\right) - \rho(\alpha_1+\alpha_2-\gamma_1-\gamma_2) \end{bmatrix} \tag{4.51}$$

and

$$\mathbf{I}_q = 4\begin{bmatrix} \left(r_1^{(q)}h_0^{(i)} + r_2^{(q)}h_1^{(i)} + r_3^{(q)}h_2^{(i)}\right) - \left(r_1^{(i)}h_0^{(q)} + r_2^{(i)}h_1^{(q)} + r_3^{(i)}h_2^{(q)}\right) - \rho(\alpha_1+\alpha_2-\gamma_1-\gamma_2) \\ \left(r_2^{(q)}h_0^{(i)} + r_3^{(q)}h_1^{(i)} + r_4^{(q)}h_2^{(i)}\right) - \left(r_2^{(i)}h_0^{(q)} + r_3^{(i)}h_1^{(q)} + +r_4^{(i)}h_2^{(q)}\right) - \rho(\alpha_2-\gamma_2) \\ \left(r_3^{(q)}h_0^{(i)} + r_4^{(q)}h_1^{(i)} + r_5^{(q)}h_2^{(i)}\right) - \left(r_3^{(i)}h_0^{(q)} + r_4^{(i)}h_1^{(q)} + r_5^{(i)}h_2^{(q)}\right) \\ \left(r_4^{(q)}h_0^{(i)} + r_5^{(q)}h_1^{(i)} + r_5^{(q)}h_2^{(i)}\right) - \left(r_4^{(i)}h_0^{(q)} + r_5^{(i)}h_1^{(q)} + r_6^{(i)}h_2^{(q)}\right) \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \left(r_{N-3}^{(q)}h_0^{(i)} + r_{N-2}^{(q)}h_1^{(i)} + r_{N-1}^{(q)}h_2^{(i)}\right) - \left(r_{N-3}^{(i)}h_0^{(q)} + r_{N-2}^{(i)}h_1^{(q)} + r_{N-1}^{(i)}h_2^{(q)}\right) \\ \left(r_{N-2}^{(q)}h_0^{(i)} + r_{N-1}^{(q)}h_1^{(i)} + r_N^{(q)}h_2^{(i)}\right) - \left(r_{N-2}^{(i)}h_0^{(q)} + r_{N-1}^{(i)}h_1^{(q)} + r_N^{(i)}h_2^{(q)}\right) \\ \left(r_{N-1}^{(q)}h_0^{(i)} + r_N^{(q)}h_1^{(i)} + r_{N+1}^{(q)}h_2^{(i)}\right) - \left(r_{N-1}^{(i)}h_0^{(q)} + r_N^{(i)}h_1^{(q)} + +r_{N+1}^{(i)}h_2^{(q)}\right) - \rho(\alpha_2+\gamma_2) \\ \left(r_N^{(q)}h_0^{(i)} + r_{N+1}^{(q)}h_1^{(i)} + r_{N+2}^{(q)}h_2^{(i)}\right) - \left(r_N^{(i)}h_0^{(q)} + r_{N+1}^{(i)}h_1^{(q)} + r_{N+2}^{(i)}h_2^{(q)}\right) - \rho(\alpha_1+\alpha_2+\gamma_1+\gamma_2) \end{bmatrix} \tag{4.52}$$

where $\rho = 1/\sqrt{2}$. It is clear that there are similar patterns emerging in $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$ for the M-QAM model, than those that emerged in $\mathbf{T}$ and $\mathbf{I}$ for the BPSK model. These observations can now be used to create a model for an M-QAM equalizer for the general case.

4.2.2.3 A General Model for the M-QAM MLSE Equalizer

By comparing $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$ for $L = 2$ and $L = 3$, the following observations are made:

1. The number of symmetric bands in $\mathbf{T}_i$ and $\mathbf{T}_q$ expands correspondingly.
   (a) For $L = 2$ there is only one symmetric band around the diagonal
   (b) For $L = 3$ there are two symmetric bands around the diagonal.

   Rule 1: For $L = l$ there are $l - 1$ symmetric bands around the diagonals of $\mathbf{T}_i$ and $\mathbf{T}_q$.

2. The band of non-zero elements in $\mathbf{T}_i$ expands.
   (a) For $L = 2$, α_1 contains two terms. These terms are the summation of the product of the adjacent real and imaginary components of the CIR coefficients.
   $(\alpha_1 = h_0^{(i)}h_1^{(i)} + h_0^{(q)}h_1^{(q)})$
   (b) For $L = 3$, there are two α-terms.
      i. α_1 is the summation of the product of the adjacent real and imaginary components of the CIR coefficients.
      $(\alpha_1 = h_0^{(i)}h_1^{(i)} + h_1^{(i)}h_2^{(i)} + h_0^{(q)}h_1^{(q)} + h_1^{(q)}h_2^{(q)})$
      ii. α_2 is the summation of the product of every second real and imaginary component of the CIR coefficients.
      $(\alpha_2 = h_0^{(i)}h_2^{(i)} + h_0^{(q)}h_2^{(q)})$

   Rule 2: For $L = l$, there are $l - 1$ α-elements. α_{l-1} is the summation of the product of every $(l - 1)$th real and imaginary component of the CIR coefficient. Real components are multiplied with real components, and imaginary components are multiplied with imaginary components.

3. The band of non-zero elements in $\mathbf{T}_q$ expands.
   (a) For $L = 2$, γ_1 contains two terms. These terms are the difference between the product of the adjacent real and imaginary components of the CIR coefficients.
   $(\gamma_1 = h_0^{(q)}h_1^{(i)} - h_0^{(i)}h_1^{(q)})$
   (b) For $L = 3$, there are two α-terms.
      i. γ_1 is the difference between the product of the adjacent real and imaginary components of the CIR coefficients.
      $(\gamma_1 = h_0^{(q)}h_1^{(i)} + h_1^{(q)}h_2^{(i)} - h_0^{(i)}h_1^{(q)} - h_1^{(i)}h_2^{(q)})$
      ii. γ_2 is the difference between the product of every second real and imaginary component of the CIR coefficients.
      $(\gamma_2 = h_0^{(q)}h_2^{(i)} - h_0^{(i)}h_2^{(q)})$

   Rule 3: For $L = l$, there are $l - 1$ γ-elements. γ_{l-1} is the difference between the product of every $(l - 1)$th real and imaginary component of the CIR coefficient. Real components are multiplied with imaginary components

4. The number of $r_n h_m$ terms in each element of $\mathbf{I}_i$ increases.

   (a) For $L = 2$, there are four $r_n h_m$ terms, two corresponding to the real components and two corresponding to the imaginary components of $\mathbf{r}$ and $\mathbf{h}$, respecetively.

   (b) For $L = 3$, there are six $r_n h_m$ terms, three corresponding to the real components and three corresponding to the imaginary components of $\mathbf{r}$ and $\mathbf{h}$, respectively.

   Rule 4: For $L = l$, there are $2l$ $r_n h_m$ terms, l corresponding to the real components and l corresponding to the imaginary components of $\mathbf{r}$ and $\mathbf{h}$, respecetively.

5. The number of $r_n h_m$ terms in each element of $\mathbf{I}_q$ increases.

   (a) For $L = 2$, there are four $r_n h_m$ terms, two negative and two positive. All are mixtures of the imaginary components of $\mathbf{r}$ and $\mathbf{h}$.

   (b) For $L = 3$, there are six $r_n h_m$ terms, three negative and three positive. All are mixtures of the imaginary components of $\mathbf{r}$ and $\mathbf{h}$.

   Rule 5: For $L = l$, there are $2l$ $r_n h_m$ terms, l negative and l positive. All are mixtures of the imaginary components of $\mathbf{r}$ and $\mathbf{h}$.

6. The number of α and γ terms in $\mathbf{I}_i$ increases.

   (a) For $L = 2$ there is one α term and one γ term in the first and the last elements of $\mathbf{I}_i$ (α_1 and γ_1). The γ-term in Nth element of $\mathbf{I}_i$ is positive and all other α and γ terms are negative.

   (b) For $L = 3$ there are two α terms and two γ terms in the first and the last elements of $\mathbf{I}_i$ (α_1, α_2, γ_1 and γ_2), and one α term and one γ term in the second and the second-to-last element of $\mathbf{I}_i$ (α_2 and γ_2). The γ-terms in Nth and $(N-1)$th element of $\mathbf{I}_i$ are positive and all other α and γ terms are negative.

   Rule 6: For $L = l$ there are $l-1$ α terms and $l-1$ γ terms in the first and last elements of $\mathbf{I}_i$ (α_1 to α_{l-1}) and (γ_1 to γ_{l-1}) and $l-m+1$ α terms and $l-m+1$ γ terms in the mth and mth last element of $\mathbf{I}_i$ (α_m to α_{l-1} and γ_m to γ_{l-1}). The γ-terms from the Nth to the mth last element of $\mathbf{I}_i$ are positive and all other α and γ terms are negative.

7. The number of α and γ terms in $\mathbf{I}_q$ increases.

   (a) For $L = 2$ there is one α term and one γ term in the first and the last elements of $\mathbf{I}_q$ (α_1 and γ_1). The γ-term in first element of $\mathbf{I}_q$ is positive and all other α and γ terms are negative.

   (b) For $L = 3$ there are two α terms and two γ terms in the first and the last elements of $\mathbf{I}_q$ (α_1, α_2, γ_1 and γ_2), and one α term and one γ term in the second and the second-to-last element of $\mathbf{I}_q$ (α_2 and γ_2). The γ-terms in first and second element of $\mathbf{I}_q$ are positive and all other α and γ terms are negative.

Rule 7: For $L = l$ there are $l-1$ α terms and $l-1$ γ terms in the first and last elements of $\mathbf{I}_q$ (α_1 to α_{l-1}) and (γ_1 to γ_{l-1}) and $l-m+1$ α terms and $l-m+1$ γ terms in the mth and mth last element of $\mathbf{I}_q$ (α_m to α_{l-1} and γ_m to γ_{l-1}). The γ-terms from the first to the mth element of $\mathbf{I}_q$ are positive and all other α and γ terms are negative.

For the model of the M-QAM equalizer it is clear that patterns in $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$ exibit the same behavior as the patterns in $\mathbf{T}$ and $\mathbf{I}$ for the the BPSK equalizer model, although more complex. It is therefore not necessary to derive the model for a M-QAM system with N data payload symbols with a CIR length of $L = 4$, since it is expected that the pattern will continue to behave in the same way. Therefore, by following the pattern that emerged in $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$ for $L = 2$ and $L = 3$ and by following the rules above, a general model for the M-QAM HNN MLSE equalizer can be realized. For an M-QAM system with a data block length of N operating in a multipath fading environment where the channel has a CIR length of L, $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$ are respectively given by

$$\mathbf{T}_i = -4\begin{bmatrix} 0 & \alpha_1 & \alpha_2 & \dots & \alpha_{L-1} & \dots & 0 \\ \alpha_1 & 0 & \alpha_1 & \alpha_2 & \dots & \alpha_{L-1} & \vdots \\ \alpha_2 & \alpha_1 & 0 & \alpha_1 & \ddots & \vdots & \alpha_{L-1} \\ \vdots & \alpha_2 & \alpha_1 & \ddots & \ddots & \alpha_2 & \vdots \\ \alpha_{L-1} & \vdots & \ddots & \ddots & 0 & \alpha_1 & \alpha_2 \\ \vdots & \alpha_{L-1} & \dots & \alpha_2 & \alpha_1 & 0 & \alpha_1 \\ 0 & \dots & \alpha_{L-1} & \dots & \alpha_2 & \alpha_1 & 0 \end{bmatrix} \tag{4.53}$$

and

$$\mathbf{T}_q = -4\begin{bmatrix} 0 & \gamma_1 & \gamma_2 & \dots & \gamma_{L-1} & \dots & 0 \\ \gamma_1 & 0 & \gamma_1 & \gamma_2 & \dots & \gamma_{L-1} & \vdots \\ \gamma_2 & \gamma_1 & 0 & \gamma_1 & \ddots & \vdots & \gamma_{L-1} \\ \vdots & \gamma_2 & \gamma_1 & \ddots & \ddots & \gamma_2 & \vdots \\ \gamma_{L-1} & \vdots & \ddots & \ddots & 0 & \gamma_1 & \gamma_2 \\ \vdots & \gamma_{L-1} & \dots & \gamma_2 & \gamma_1 & 0 & \gamma_1 \\ 0 & \dots & \gamma_{L-1} & \dots & \gamma_2 & \gamma_1 & 0 \end{bmatrix}, \tag{4.54}$$

where

$$\alpha_k = \sum_{j=0}^{L-k-1} h_j^{(i)} h_{j+k}^{(i)} + \sum_{j=0}^{L-k-1} h_j^{(q)} h_{j+k}^{(q)}, \tag{4.55}$$

and

$$\gamma_k = \sum_{j=0}^{L-k-1} h_j^{(q)} h_{j+k}^{(i)} - \sum_{j=0}^{L-k-1} h_j^{(i)} h_{j+k}^{(q)}, \tag{4.56}$$

with $k = 1, 2, 3, ..., L-1$.

Also,

$$\mathbf{I}_i = 4 \begin{bmatrix} \lambda_1 - \rho(\alpha_1 + \gamma_1 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \lambda_2 - \rho(\alpha_2 + \gamma_2 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \lambda_3 - \rho(\alpha_3 + \gamma_3 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \vdots \\ \lambda_{L-1} - \rho(\alpha_{L-1} + \gamma_{L-1}) \\ \lambda_L \\ \vdots \\ \lambda_{N-L+1} \\ \lambda_{N-L+2} - \rho(\alpha_{L-1} - \gamma_{L-1}) \\ \vdots \\ \lambda_{N-2} - \rho(\alpha_3 - \gamma_3 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \lambda_{N-1} - \rho(\alpha_2 - \gamma_2 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \lambda_N - \rho(\alpha_1 - \gamma_1 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \end{bmatrix} \tag{4.57}$$

and

$$\mathbf{I}_q = 4 \begin{bmatrix} \omega_1 - \rho(\alpha_1 - \gamma_1 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \omega_2 - \rho(\alpha_2 - \gamma_2 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \omega_3 - \rho(\alpha_3 - \gamma_3 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \vdots \\ \omega_{L-1} - \rho(\alpha_{L-1} - \gamma_{L-1}) \\ \omega_L \\ \vdots \\ \omega_{N-L+1} \\ \omega_{N-L+2} - \rho(\alpha_{L-1} + \gamma_{L-1}) \\ \vdots \\ \omega_{N-2} - \rho(\alpha_3 + \gamma_3 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \omega_{N-1} - \rho(\alpha_2 + \gamma_2 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \omega_N - \rho(\alpha_1 + \gamma_1 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \end{bmatrix}, \tag{4.58}$$

where $\rho = 1/\sqrt{2}$ and $\lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_N\}$ is given by

$$\lambda_k = \sum_{j=0}^{L-1} r_{j+k}^{(i)} h_j^{(i)} + \sum_{j=0}^{L-1} r_{j+k}^{(q)} h_j^{(q)}, \tag{4.59}$$

and $\omega = \{\omega_1, \omega_2, \cdots, \omega_N\}$ is given by

$$\omega_k = \sum_{j=0}^{L-1} r_{j+k}^{(q)} h_j^{(i)} - \sum_{j=0}^{L-1} r_{j+k}^{(i)} h_j^{(q)}, \tag{4.60}$$

where $k = 1, 2, 3, ..., N$.

This concludes the derivation of the model for a M-QAM HNN MLSE equalizer. It will be demonstrated in *Chapter 5* how this model, as well as the model derived for the BPSK MLSE equalizer, is used to equalize signals in wireless communication systems with hundreds of CIR taps.

4.3 The Iterative System

Up to this point the respective models for the BPSK and M-QAM HNN MLSE equalizers have been derived by mapping the cost function (4.2) that is minimized by the Viterbi MLSE algorithm to the energy function (4.3) of the HNN, in order to perform MLSE equalization by using the HNN. In this section the iterative nature of the HNN MLSE equalizer is discussed.

In [5] it was shown that the HNN energy function (4.3) is a monotonically decreasing function, also called a *Lyaponov* function, when the decision function associated with each neuron in the network approaches a *sgn*-function. Hopfield also showed that the system will always converge to a stable state, although not an optimal state, which is associated with a local minimum in the N-dimensional solution space. Because the system is not guaranteed to converge to the global minimum, which will produce the optimal maximum likelihood sequence estimates in this case, optimization techniques have to be employed to allow the system to escape less optimal states in order to produce near-optimal transmitted sequence estimates.

4.3.1 The Solution Space

Before more light can be shed on the iterative nature of the HNN MLSE equalizer, the concept of *solution space* must be explained, since the aim of the HNN MLSE equalizer is to converge to a near-optimal local minimum in the solution space in order to produce near-optimal maximum likelihood transmitted sequence estimates.

In general it is not possible to visualize the solution space for the problem of concern, because the solution space contains N dimensions,[2] where N is the data block length as before. The only viable approach is to divide the solution space into $N - 1$ segments, each of which can be viewed in a three-dimensional space.

Consider a hypothetical BPSK communication system transmitting a sequence of $N = 5$ symbols $\{-1, 1, -1, 1, -1\}$ through a multipath channel with CIR length $L = 3$, where

[2] The complete solution space can be visualized for $N = 2$.

$E_b/N_0 = 5$ dB. By evaluating the HNN energy function in (4.3) for all possible combinations of $\mathbf{s} = \{s_1, s_2, \cdots, s_N\}^T$, with a resolution of $\Delta_s = 0.1$, the solution space can be visualized by segmenting the N-dimensional hypercube into $N-1$ distinct segments. Fig. 4.2 shows four segments of the solution space, where red and blue colors indicate areas of high and low energy respectively.

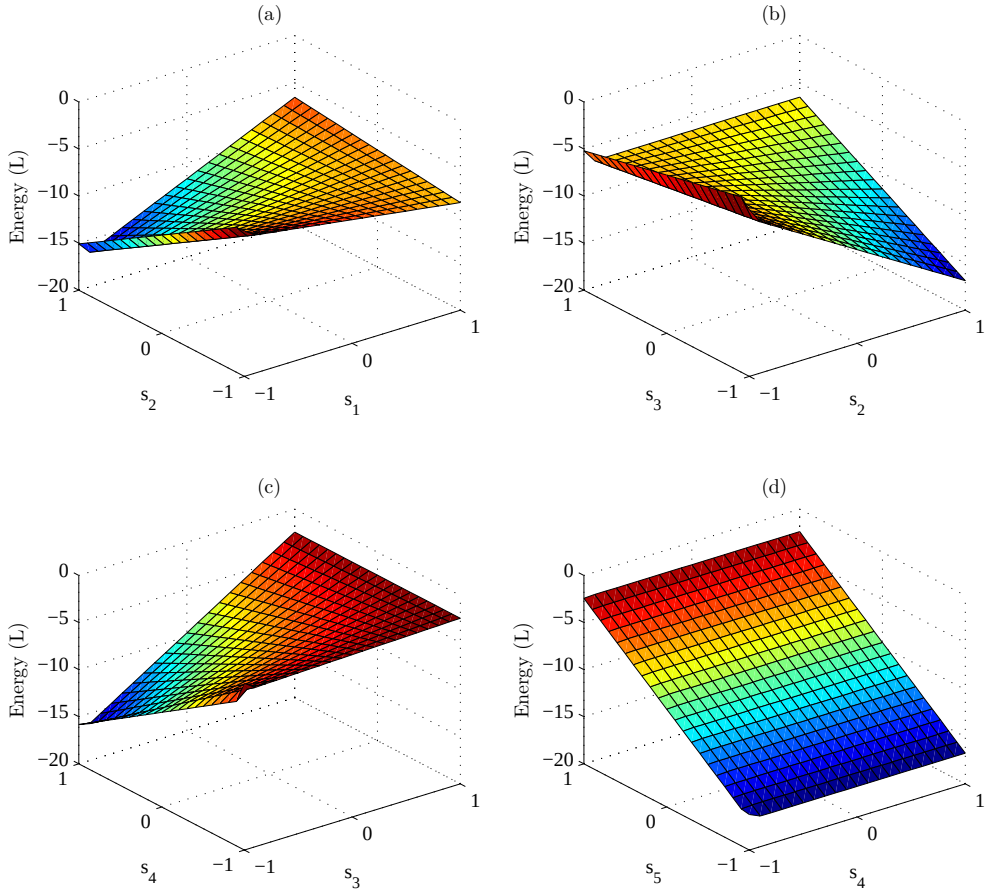

FIGURE 4.2: Segmented solution space for $N = 5$ and $L = 3$ and $\mathbf{s} = \{-1, 1, -1, 1, -1\}^T$.

The following is observed concerning the respective solution space segments in Fig. 4.2:

- (a) shows the solution space for s_1 and s_2. It is clear that the energy will be a minimum for $s_1 = -1$ and $s_2 = 1$.
- (b) shows the solution space for s_2 and s_3. It is clear that the energy will be a minimum for $s_2 = 1$ and $s_3 = -1$.
- (c) shows the solution space for s_3 and s_4. It is clear that the energy will be a minimum for $s_3 = -1$ and $s_4 = 1$.
- (d) shows the solution space for s_4 and s_5. It is clear that the energy will be a minimum for all values of s_4 and for $s_5 = -1$. However, from (c) it is clear that the energy will be a minimum for $s_4 = 1$.

Therefore, the HNN MLSE equalizer must minimize the energy in the network such that the output of the $N = 5$ neurons converge to settle in the corners of the N-dimensional hypercube, which will yield the MLSE estimates $\mathbf{s} = \{-1, 1, -1, 1, -1\}^T$.

4.3.2 Minimizing the HNN Energy Function

In order for the HNN to minimize the energy function (4.3), the following dynamic system is used [5]:

$$\frac{d\mathbf{u}}{dt} = -\frac{\mathbf{u}}{\tau} + \mathbf{Ts} + \mathbf{I}, \tag{4.61}$$

where τ is an arbitrary constant and $\mathbf{u} = \{u_1, u_2, \cdots, u_N\}^T$ is the internal state of the network. An iterative solution for (4.61) is given by

$$\begin{aligned} \mathbf{u}^{(n)} &= \mathbf{Ts}^{(n-1)} + \mathbf{I} \\ \mathbf{s}^{(n)} &= g\left(\beta(n)\mathbf{u}^{(n)}\right), \end{aligned} \tag{4.62}$$

where again $\mathbf{u} = \{u_1, u_2, \cdots, u_N\}^T$ is the internal state of the network, $\mathbf{s} = \{s_1, s_2, \cdots, s_N\}^T$ is the vector of estimated symbols, $g(.)$ is the decision function associated with each neuron and n indicates the iteration number. $\beta(.)$ is a function used for optimization.

To determine the MLSE estimate for a data block of length N with L CIR coefficients for a BPSK system, the following steps are executed:

1. Use the received symbols $\mathbf{r}$ and the estimated CIR $\mathbf{h}$ to calculate $\mathbf{T}$ and $\mathbf{I}$ according to (4.22) and (4.23).
2. Initialize all elements in $\mathbf{s}$ to 0 so that $\mathbf{s} = \{s_1, s_2, \cdots, s_N\}^T = \{0, 0, \cdots, 0\}^T$.
3. Calculate $\mathbf{u}^{(n)} = \mathbf{Ts}^{(n-1)} + \mathbf{I}$, where n indicates the interation number.
4. Calculate $\mathbf{s}^{(n)} = g\left(\beta(n)\mathbf{u}^{(n)}\right)$.
5. Go to step 2 and repeat until $n = Z$, where Z is the predetermined number of iterations.

As is clear from the algorithm, the estimated symbol vector $\mathbf{s}$ is updated with each iteration. $\mathbf{I}$ contains the best linear estimate for $\mathbf{s}$,[3] and is therefore used as input to the network, while $\mathbf{T}$ contains the correlation information of the transmitted symbols. The system solves (4.2) by iteratively mitigating the effect of ISI and produces the MLSE estimates in $\mathbf{s}$ after Z iterations. The BPSK equalizer uses a bipolar decision function.

[3] It can be shown that $\mathbf{I}$ contains the output of a RAKE receiver used in DSSS systems.

Similarly, to determine the MLSE estimate for a data block of length N with L CIR coefficients for a M-QAM system, the following steps are executed:

1. Use the received symbols $\mathbf{r}$ and the estimated CIR $\mathbf{h}$ to calculate $\mathbf{T}_i$, $\mathbf{T}_q$, $\mathbf{I}_i$ and $\mathbf{I}_q$ according to (4.53), (4.54), (4.57) and (4.58).
2. Initialize all elements in $[\mathbf{s}_i^T|\mathbf{s}_q^T]$ to 0.
3. Calculate $[\mathbf{u}_i^T|\mathbf{u}_q^T]^{(n)} = \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i \end{bmatrix} \left[\frac{\mathbf{s}_i}{\mathbf{s}_q}\right]^{(n-1)} + [\mathbf{I}_i^T|\mathbf{I}_q^T]$.
4. Calculate $[\mathbf{s}_i^T|\mathbf{s}_q^T]^{(n)} = g\left(\beta(n)[\mathbf{u}_i^T|\mathbf{u}_q^T]^{(n)}\right)$.
5. Go to step 2 and repeat until $n = Z$, where Z is the predetermined number of iterations.

For the M-QAM system, the bipolar decision function is replaced with a multilevel decision function, where the number of levels are chosen according to the number of signal levels required by the M-QAM modulation scheme.

To demonstrate how the HNN MLSE equalizer iteratively minimizes the energy function in (4.3) to produce MLSE estimates, the hypothetical system of *Section 4.3.1* will be used, but now with $E_b/N_0 = -5$ dB.

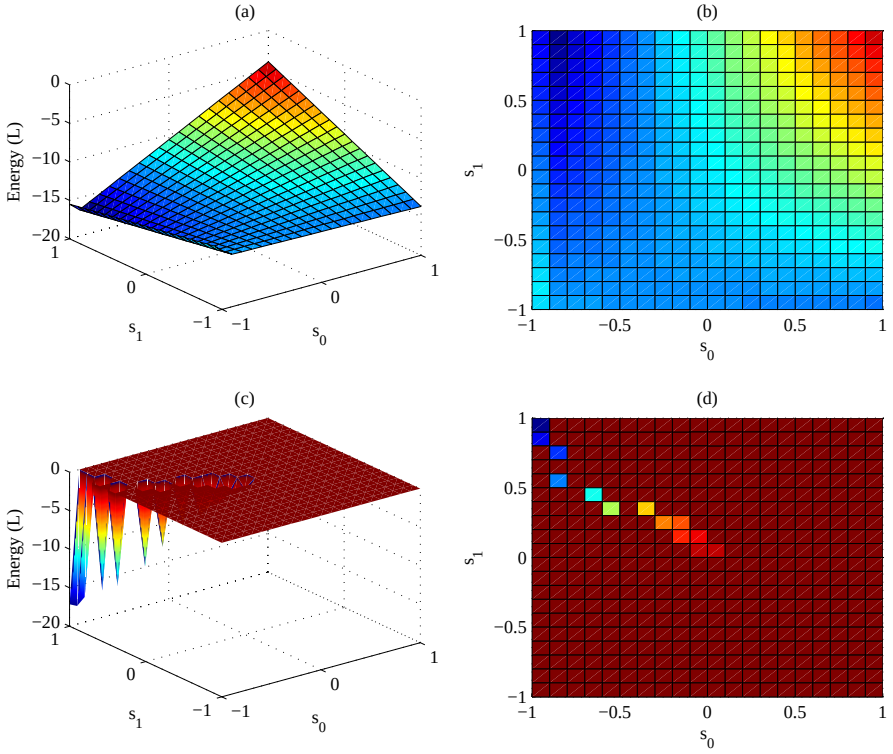

FIGURE 4.3: Convergence of the HNN MLSE equalizer for s_0 and s_1.

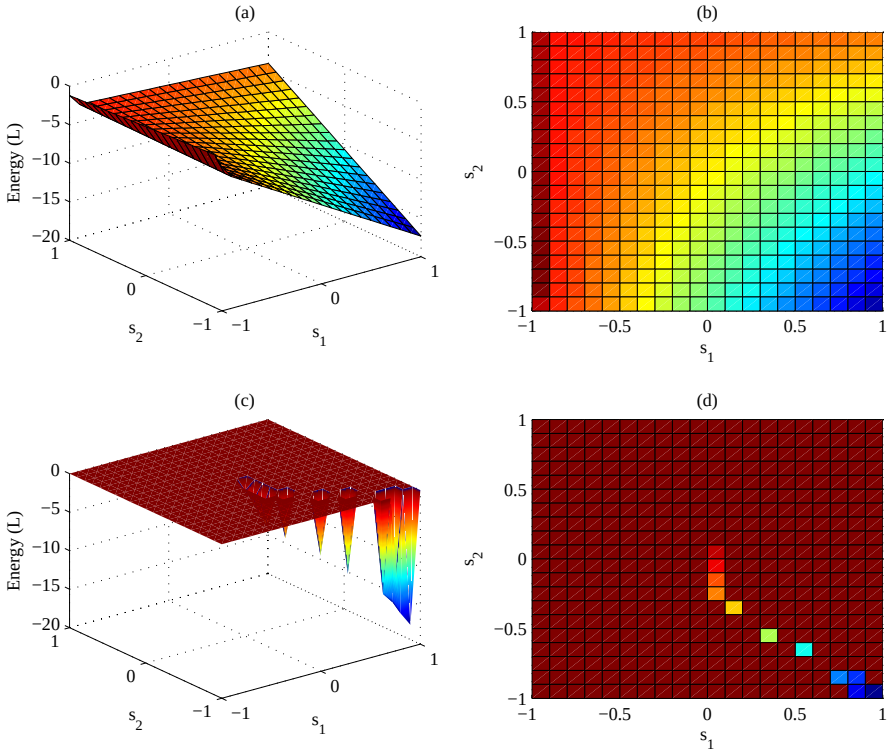

FIGURE 4.4: Convergence of the HNN MLSE equalizer for s_1 and s_2.

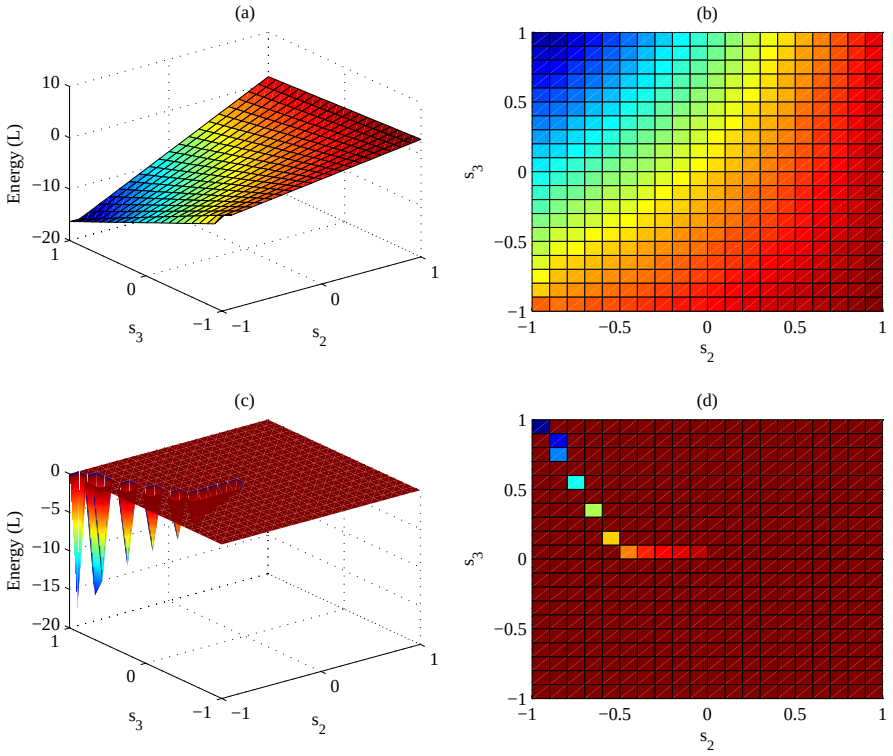

FIGURE 4.5: Convergence of the HNN MLSE equalizer for s_2 and s_3.

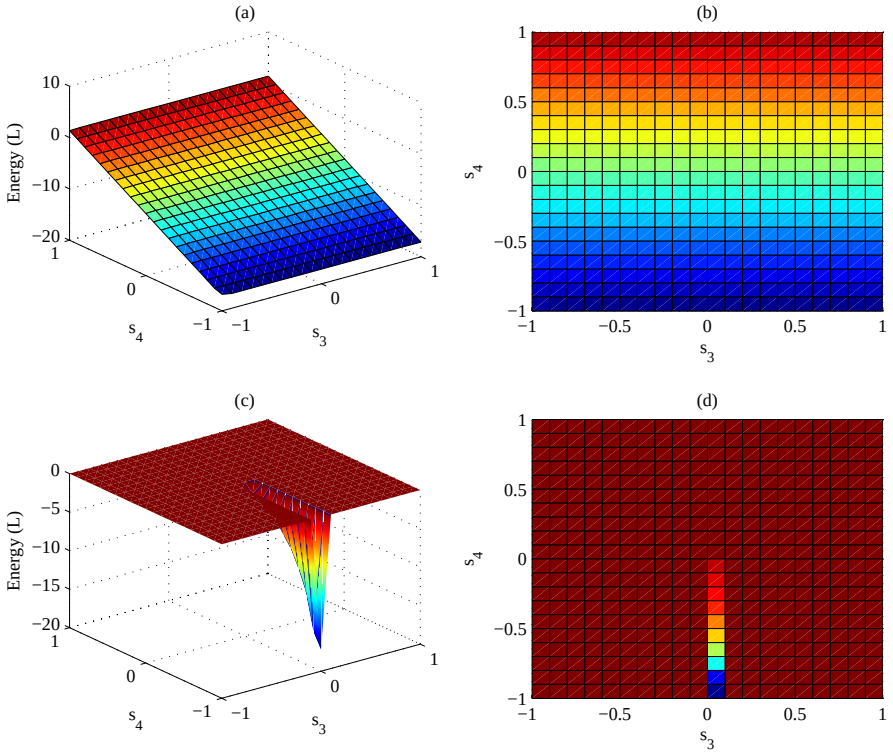

FIGURE 4.6: Convergence of the HNN MLSE equalizer for s_3 and s_4.

Figures 4.3 to 4.6 show the segmented solution space as before as well as the convergence of neuron outputs in the HNN MLSE equalizer for each segment of the N-dimensional solution space, where the red regions correspond with regions of high energy and blue correspond with regions of low energy. Figures 4.3 (a), 4.4 (a), 4.5 (a) and 4.6 (a) show the three-dimensional segments of the solution space. Figures 4.3 (b), 4.4 (b), 4.5 (b) and 4.6 (b) show the three-dimensional segments of the solution space as viewed from above. Figures 4.3 (c), 4.4 (c), 4.5 (c) and 4.6 (c) show the three-dimensional convergence of the outputs of the respective neurons in the HNN MLSE equalizer and figures 4.3 (d), 4.4 (d), 4.5 (d) and 4.6 (d) show the convergence of the respective neurons in the HNN MLSE equalizer as viewed from above.

From these figures it is clear that the outputs of the respective neurons, starting from a zero initial state, traverse the solution space to converge to the values that minimize the HNN energy function in (4.3), hence producing the MLSE estimates of the transmitted symbols.

4.4 Decision Functions

In the previous section reference was made to the decision functions used in the neural network. In a neural network the decision function, also called the activation function or the threshold function, is used to determine the level of activation of a specific neuron, depending on the collective severity of the weighted inputs to the neuron.

4.4.1 Bipolar Decision Function

When BPSK modulation is used, only two signal levels are required to transmit information. Therefore, since only two signal levels are used, a bipolar decision function is used in the BPSK HNN MLSE equalizer. This function, also called a bipolar sigmoid function, is expressed as

$$g(u) = \frac{2}{1 + e^{-u}} - 1, \tag{4.63}$$

and is shown in Fig. 4.7. It must also be noted that the bipolar decision can also be used in

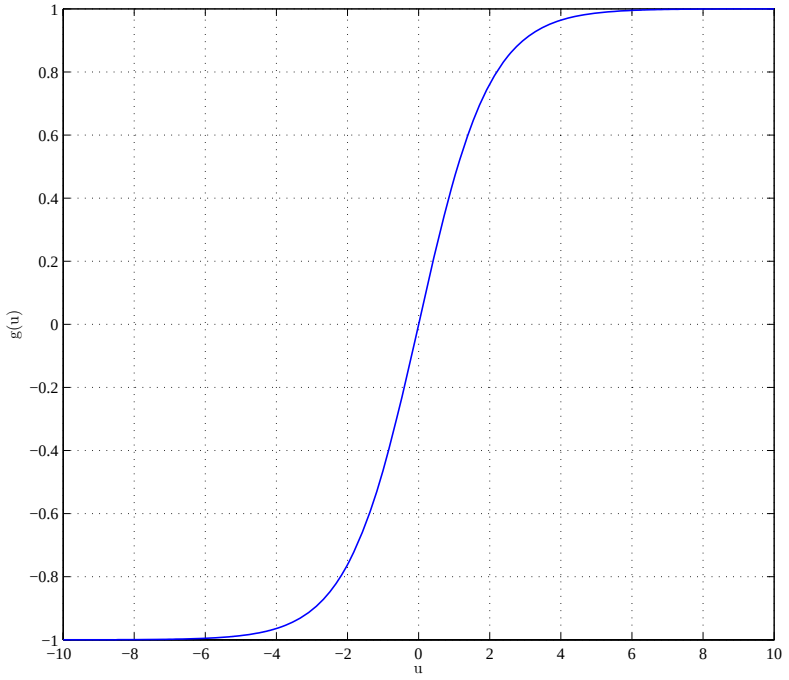

FIGURE 4.7: The bipolar sigmoid function.

the M-QAM model for equalization in 4-QAM systems. This is an exception, since, although 4-QAM modulation uses four signal levels, there are only two signal levels per dimension. By using the model derived for M-QAM modulation, 4-QAM equalization can be performed by using the bipolar decision function in (4.63), with the output scaled by a factor $1/\sqrt{2}$.

4.4.2 Multilevel Decision Function

Apart from 4-QAM modulation, all other M-QAM modulation schemes use multiple amplitude levels to transmit information as the "AM" in the acronym M-QAM implies. A bipolar decision function will therefore not be sufficient; a multilevel decision function with $log_2(M)$ distinct signal levels must be used, where M is the modulation alphabet size.

A multilevel decision function can be realized by adding several bipolar decision functions and shifting each by a predetermined value and scaling the result accordingly [44, 45]. To realize a $log_2(M)$-level decision function for use in an M-QAM HNN MLSE equalizer, the following function can be used:

$$g(u) = \frac{2}{\sqrt{2}\left(log_2(M) - 1\right)} \left(\sum_{k=-((log_2(M)/2)-1)}^{(log_2(M)/2)-1} \frac{1}{1 + e^{-(u+\phi k)}} \right) - \frac{1}{\sqrt{2}}, \tag{4.64}$$

where M is the modulation alphabet size and ϕ is the value by which the respective bipolar decision functions are shifted. ϕ is dependent on the power in the channel.

In this thesis a 16-QAM HNN MLSE equalizer is considered, requiring a four-level decision function, since 16-QAM modulation requires four distinct amplitude levels in each dimension to realize a modulation alphabet containing sixteen symbols. To realize the four-level decision function, (4.64) can be used with $M = 16$ to give

$$g(u) = \frac{2}{3\sqrt{2}} \left(\sum_{k=-1}^{1} \frac{1}{1 + e^{-(u+\phi k)}} \right) - \frac{1}{\sqrt{2}}. \tag{4.65}$$

Fig. 4.8 shows the four-level decision function used in the 16-QAM HNN MLSE equalizer.

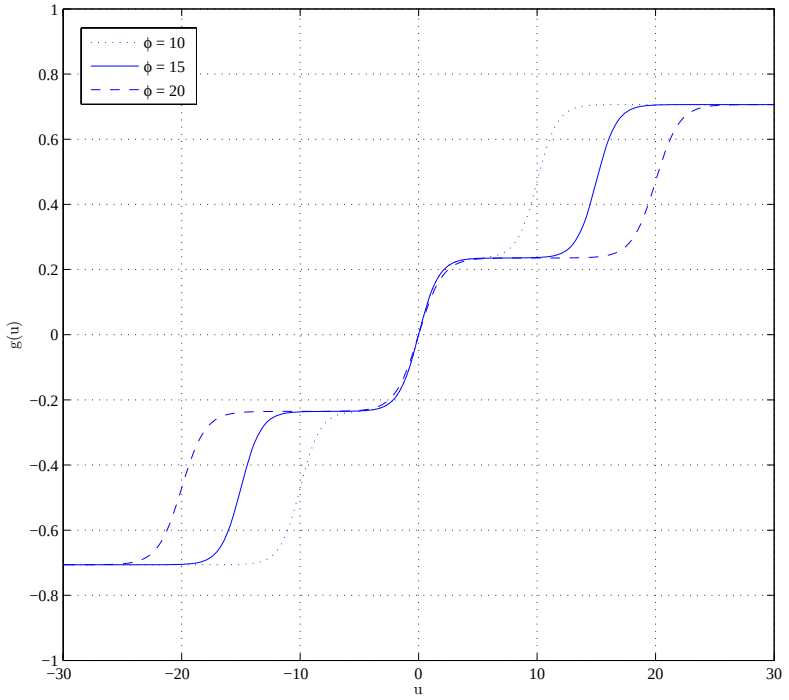

FIGURE 4.8: The four-level decision function.

Due to the time-varying nature of a mobile wireless communication channel and energy losses caused by absorption and scattering, the total power in the received signal is also time-variant. This complicates equalization when using the 16-QAM HNN MLSE equalizer, since the value by which the respective bipolar decision functions are shifted, ϕ, is dependent on the power in the channel and will therefore have a different value for every new data block arriving at the receiver. For this reason the four-level decision function in (4.65) will change slightly for every data block. ϕ is equal to the *Euclidean norm* of the estimated CIR and is determined by

$$\phi = \|\mathbf{h}\| = \sqrt{\left(\sum_{k=0}^{L-1}\left(h_k^{(i)}\right)^2\right)^2 + \left(\sum_{k=0}^{L-1}\left(h_k^{(q)}\right)^2\right)^2}, \tag{4.66}$$

where $h_k^{(i)}$ and $h_k^{(q)}$ are the kth respective in-phase and quadrature components of the estimated CIR of length L as before.

Fig. 4.8 shows the four-level decision function for different values of ϕ to demonstrate the effect of varying power levels in the channel. Higher power in $\mathbf{h}$ will cause the outer neurons to move away from the origin, whereas lower power will cause the outer neurons to move towards the origin. Therefore, upon reception of a complete data block, ϕ is determined according to the power of the CIR after which equalization can commence.

4.5 Optimization

Because MLSE is an NP-complete problem, there are a number of possible "*good*" solutions in the multidimensional solution space. By enumerating every possible solution, it will be possible to find the best solution, that is, the sequence of symbols that minimizes (4.2) and (4.3), but it is not computationally feasible for systems with large N and L. The HNN is used to minimize (4.3) to find a near-optimal solution at very low computational cost. Because the HNN usually gets stuck in suboptimal local minima, it is necessary to employ optimization techniques [46]. To aid the HNN in escaping less optimal basins of attraction, simulated annealing and asynchronous neuron updates are often used.

4.5.1 Simulated Annealing

Simulated annealing has its origin in metallurgy. In metallurgy annealing is the process used to temper steel and glass by heating them to a high temperature and then gradually cooling them, thus allowing the material to coalesce into a low-energy crystalline state [20]. In neural networks this process is imitated to ensure that the neural network escapes less optimal local minima to converge to a near-optimal solution in the solution space. As the neural network starts to iterate, there are many candidate solutions in the solution space, but because the

neural network starts to iterate at a high temperature, it is able to escape the less optimal local minima in the solutions space. As the temperature decreases, the network can still escape less optimal local minima, but it will start to gradually converge to the global minimum in the solution space to minimize the energy. This state of minimum energy corresponds to the optimal solution.

The output of the function $\beta(.)$ in (4.62) is used for simulated annealing. As the system iterates, n is incremented, and $\beta(.)$ produces a value according to an exponential function to ensure that the system converges to a near-optimal local minimum in the solution space. This function is give by

$$\beta(n-1) = 5^{\frac{2(n-Z+1)}{Z}}, \tag{4.67}$$

and shown in Fig. 4.9. This causes the output of $\beta(.)$ to start at a near-zero value and to exponentially converge to 1 with each iteration.

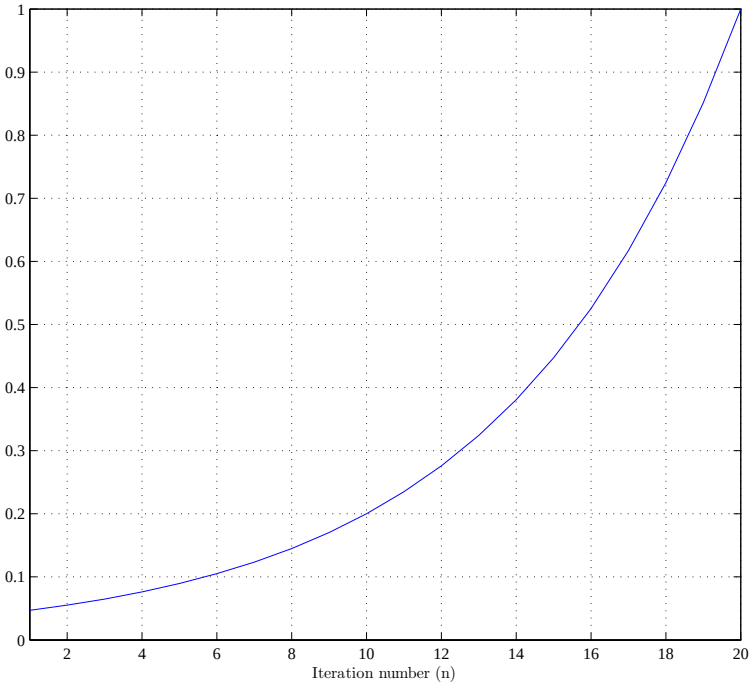

FIGURE 4.9: β-updates for $Z = 20$ iterations.

The effect of annealing on the bipolar and four-level decision function during the iteration cycle is shown in Fig. 4.10 and 4.11 respectively, with the slope of the decision function increasing as $\beta(.)$ is updated with each iteration. Simulated annealing ensures near-optimal sequence estimation by allowing the system to escape less optimal local minima in the solution space, leading to better system performance.

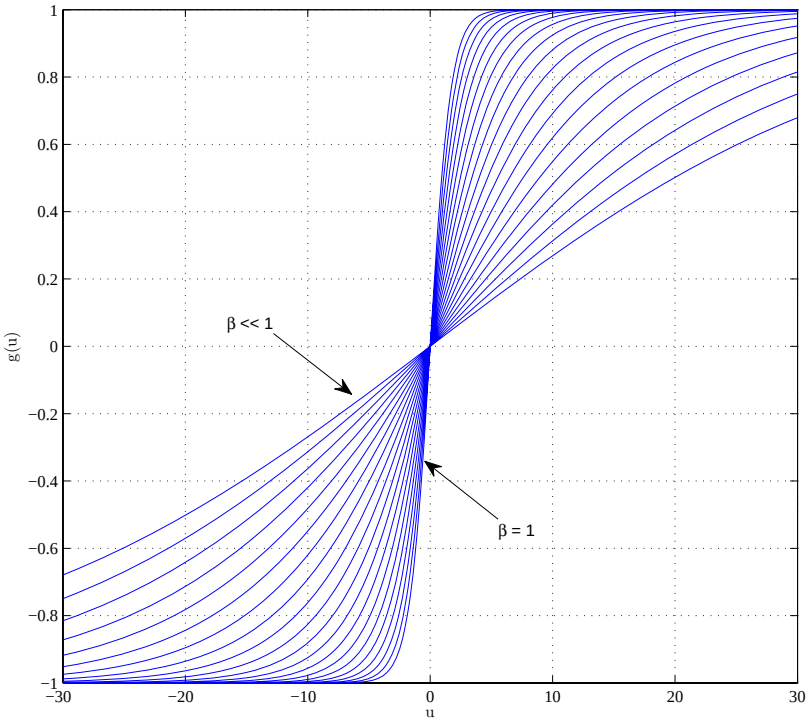

FIGURE 4.10: Simulated annealing on the bipolar decision function for $Z = 20$ iterations.

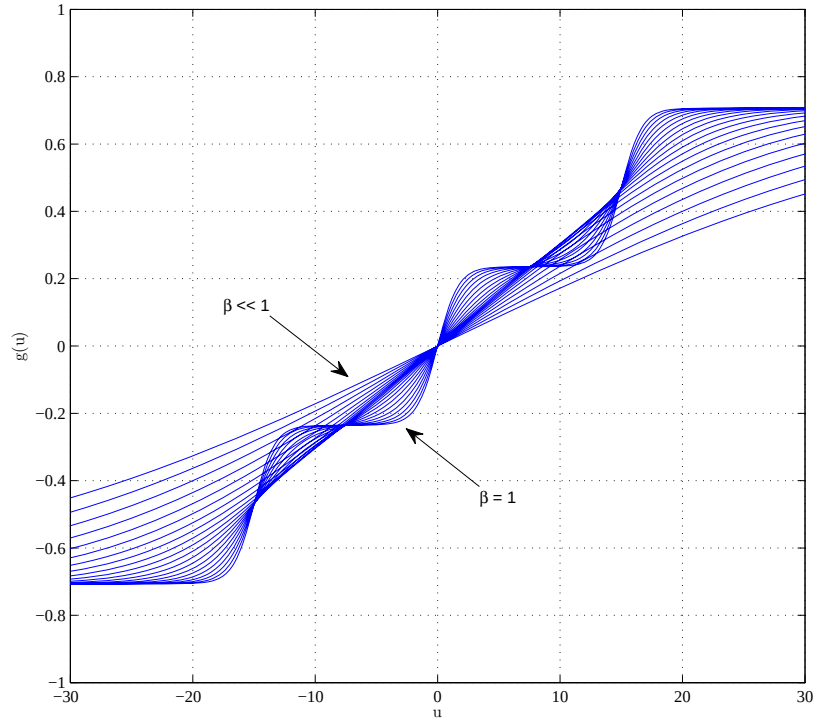

FIGURE 4.11: Simulated annealing on the four-level decision function for $Z = 20$ iterations.

To illustrate the effect of simulated annealing in the HNN MLSE equalizer, the hypothetical system that was used in *Section 4.3.2* is used here again. Figures 4.12 to 4.15 show a comparison between the convergence of the HNN on the segmented solution space for a system that does not employ simulated annealing and a system that does.

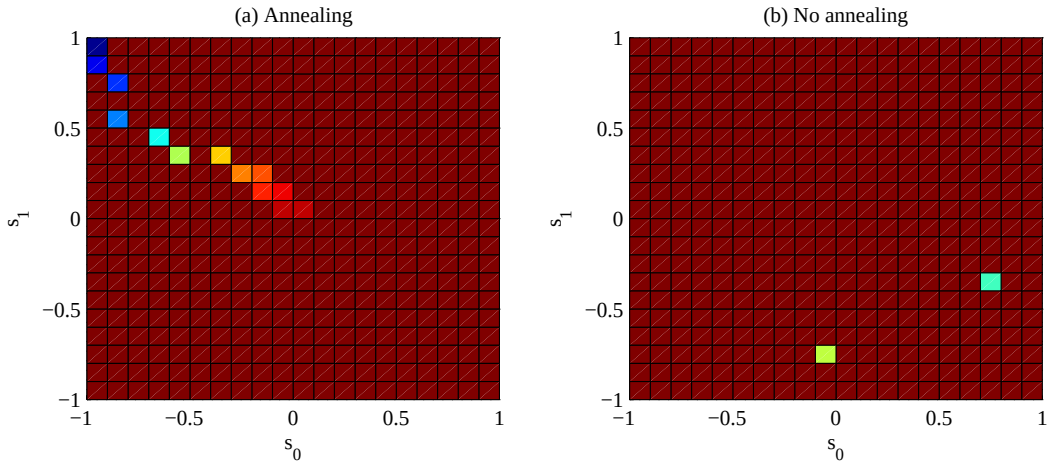

FIGURE 4.12: Convergence of the HNN MLSE equalizer for s_0 and s_1 with and without annealing.

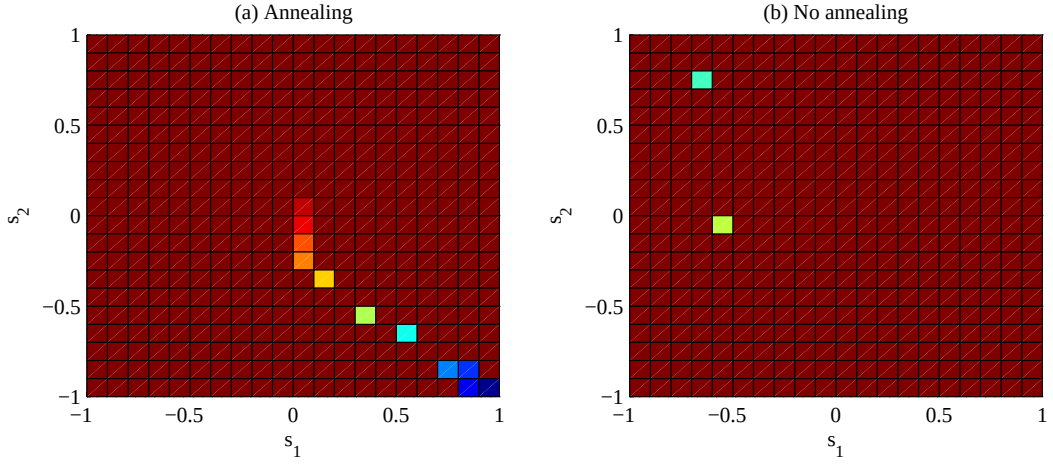

FIGURE 4.13: Convergence of the HNN MLSE equalizer for s_1 and s_2 with and without annealing.

From Fig. 4.12 to 4.15 it should be clear that annealing should improve the systems performance. Only for the solution space segment in Fig. 4.15 did the output of the neurons converge in the same direction for both the annealed and non-annealed case. As stated before, and now supported by evidence in the above mentioned figures, simulated annealing allows the system to gradually converge to a region of minimum energy which will yield reliable MLSE transmitted sequence estimates.

As another illustration of simulated annealing in the HNN MSLE equalizer, consider a BPSK and 16-QAM systems with block length $N = 100$, CIR length $L = 20$, $Z = 20$ iterations and

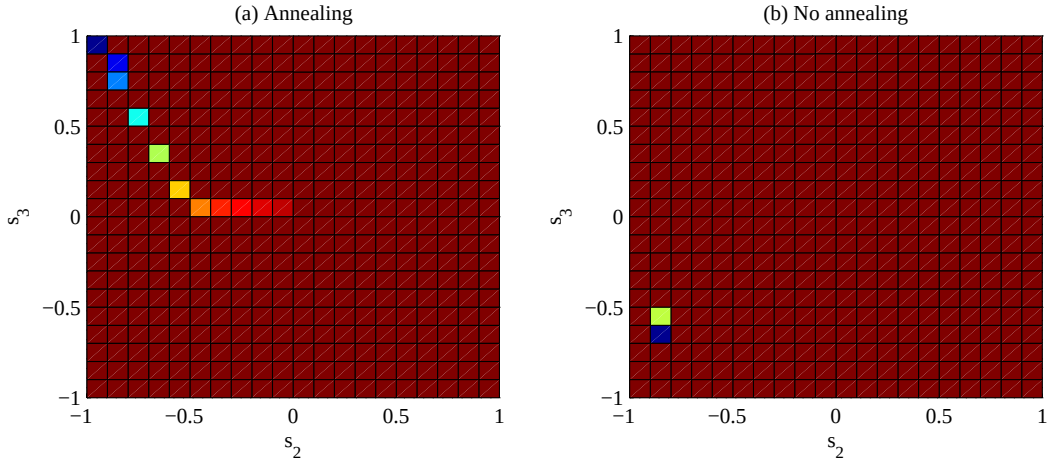

FIGURE 4.14: Convergence of the HNN MLSE equalizer for s_2 and s_3 with and without annealing.

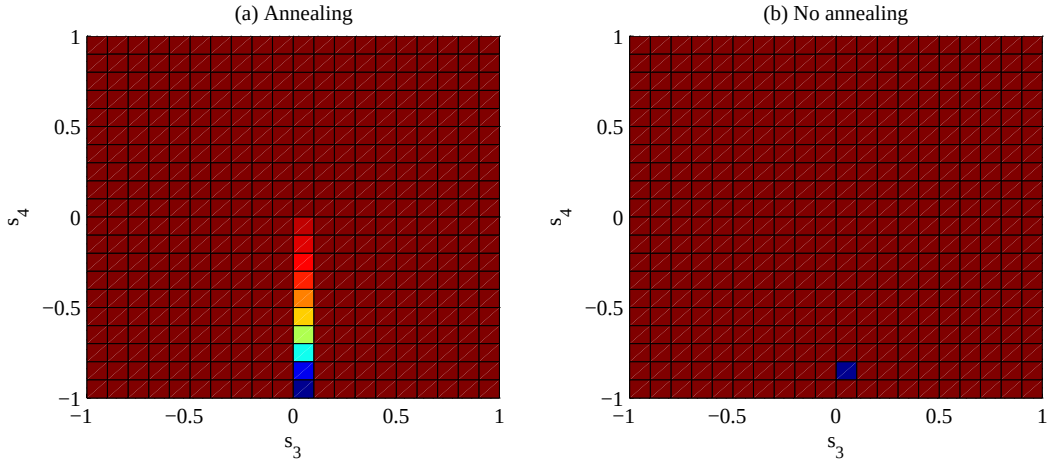

FIGURE 4.15: Convergence of the HNN MLSE equalizer for s_3 and s_4 with and without annealing.

$E_b/N_0 = 10$ dB. Figures 4.16 and 4.17 show the neuron outputs of the BPSK HNN MLSE equalizer for each iteration of the system with and without annealing. It is clear that when no annealing is used the neuron outputs settle after six iterations. Also, figures 4.18 and 4.19 show the neuron outputs, for the real and complex symbol components, of the 16-QAM HNN MLSE equalizer for each iteration of the system with and without annealing. After nine iterations the neuron outputs settle. From these figures it is clear that annealing allows the outputs of the neurons to gradually evolve in order to converge to near-optimal values in the N-dimensional hypercube, yielding near-optimal MLSE estimates.

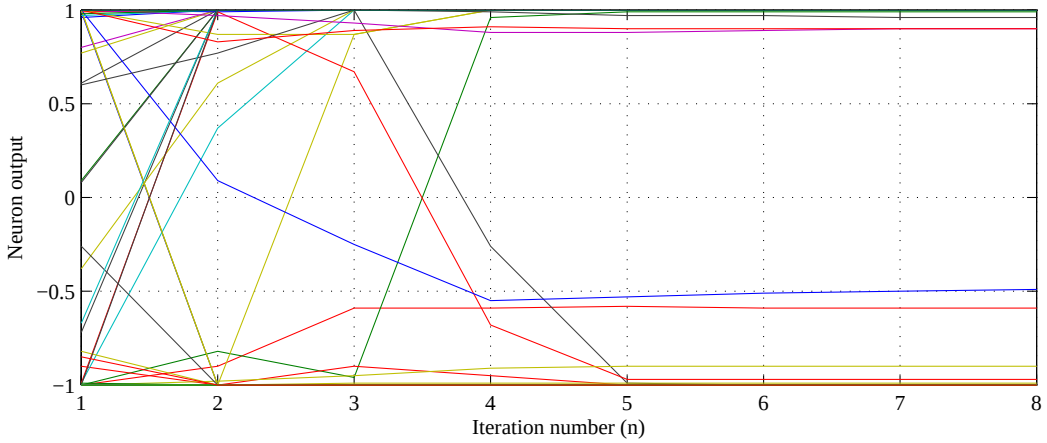

FIGURE 4.16: Convergence of the BPSK HNN MLSE equalizer without annealing.

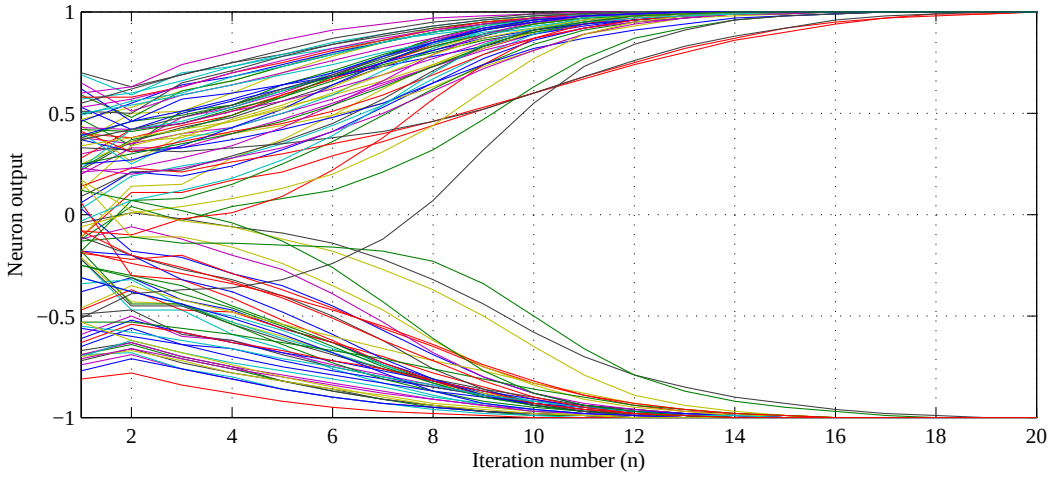

FIGURE 4.17: Convergence of the BPSK HNN MLSE equalizer with annealing.

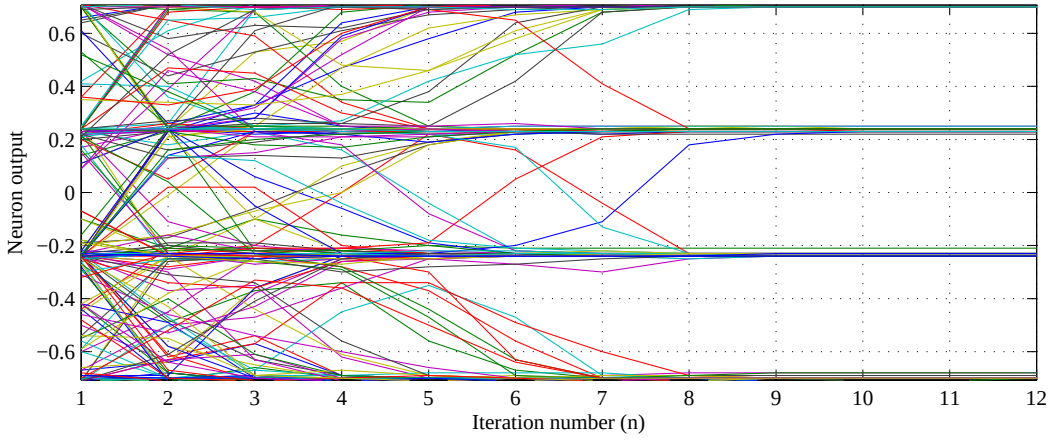

FIGURE 4.18: Convergence of the 16-QAM HNN MLSE equalizer without annealing.

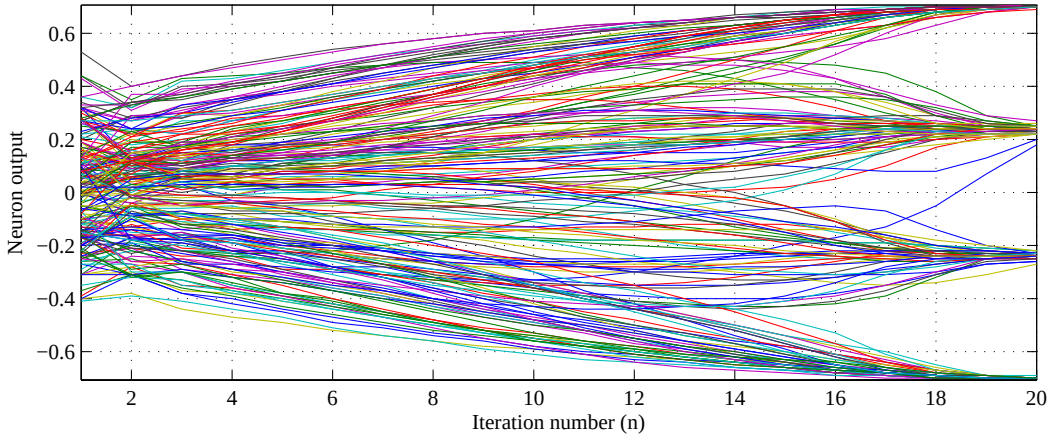

FIGURE 4.19: Convergence of the 16-QAM HNN MLSE equalizer with annealing.

4.5.2 Asynchronous Updates

In artificial neural networks, the neurons in the network can either be updated using parallel or asynchronous updates. Consider the iterative solution of the HNN in (4.62). Assume that $\mathbf{u}$, $\mathbf{s}$ and $\mathbf{I}$ each contain N elements and that $\mathbf{T}$ is an $N \times N$ matrix with Z iterations as before.

When parallel neuron updates are used, N elements in $\mathbf{u}^{(n)}$ are calculated before N elements in $\mathbf{s}^{(n)}$ are determined, for each iteration. This implies that the output of the neurons will only be a function of the neuron outputs from the previous iteration. On the other hand, when using asynchronous neuron updates, one element in $\mathbf{s}^{(n)}$ is determined for every corresponding element in $\mathbf{u}^{(n)}$. This is performed N times per iteration - once for each neuron. Asynchronous updates allow the changes of the neuron outputs to propagate to the other neurons immediately [46], while the output of all of the N neurons will only be propagated to the other neurons after all of them have been updated when parallel updates are used.

With parallel updates the effect of the updates propagates through the network only after one complete iteration cycle. This implies that the energy of the network might change drastically, because all of the neurons are updated together. This will cause the state of the neural network to "jump" around on the solution space, due to the abrupt changes in the internal state of the network. This will lead to degraded performance, since the network is not allowed to gradually evolve towards an optimal, or at least a near-optimal, basin of attraction.

With asynchronous updates the state of the network changes after each element in $\mathbf{u}^{(n)}$ is determined. This means that the state of the network undergoes N gradual changes during each iteration. This ensures that the network traverses the solution space using small steps while searching for the global minimum. The computational complexity is identical for both

parallel and asynchronous updates [46]. Asynchronous updates are therefore used for the HNN MLSE equalizers considered in this thesis.

4.5.3 Neuron Update Schedule

While iterating the HNN MLSE equalizer, the neurons are updated using asynchronous updates. The order in which the neurons are updated is another concern. Generally there are two update schedules that are often used: random and sequential [46]. Both these update schedules however yield identical system performance. Therefore a sequential update schedule is used for the HNN MLSE equalizer considered in this thesis, as it requires fewer computations than a random update schedule. The neurons are therefore updated in the following order during each iteration: $1, 2, 3, \cdots, N$.

4.6 Computational Complexity Analysis

As stated before, the computational complexity of the HNN MLSE equalizer is quadratic in the data block length N and approximately independent of the CIR length L, for practical systems. This is due to the high parallelism of its underlying neural network structure and high level of interconnection between the neurons. The approximate independence of the complexity from the channel memory is significant, as the CIR length is the dominant term in the complexity of all optimal equalizers.

In this section the computational complexity of the HNN MLSE equalizer is analyzed, where it is compared to that of the Viterbi MLSE equalizer. The computational complexities of these algorithms are analyzed by assuming that an addition as well as a multiplication are performed using one machine instruction. It is also assumed that variable initialization does not add to the cost.

4.6.1 HNN MLSE

The M-QAM HNN MLSE equalizer performs the following steps:[4]

1. *Determine α- and γ values using the estimated CIR:* There are $L-1$ distinct α values and $L-1$ distinct γ values. α_1 and γ_1 both contain $L-1$ terms, each consisting of a multiplication between two values. Also, α_{L-1} and γ_{L-1} both contain one term. Therefore the number of computations to determine all α- and γ values can be written

[4]The computational complexity of the BPSK HNN MLSE equalizer is easily derived from that of the M-QAM HNN MLSE equalizer.

as

$$\begin{aligned} 2\sum_{n=1}^{L-1} n + 2\sum_{n=1}^{L-1} n &= 4\sum_{n=1}^{L-1} n \\ &= 4\left(1+2+...+(L-1)\right)) \\ &= 4\left(((L-1)+1)+((L-2)+2)+((L-3)+3)+... \right. \\ &\quad \left. +((L-(L+1))+(L+1)\right) \\ &= 4(L-1)\left(\frac{L-1}{2}\right) \\ &= 2(L-1)^2. \end{aligned}$$

2. *Populate matrices* $\mathbf{T}_i$ *and* $\mathbf{T}_q$ *(of size* $N \times N$*) and vectors* $\mathbf{I}_i$ *and* $\mathbf{I}_q$ *(of size* N*)*: Under the assumption that variable initialization does not add to the total cost, the population of $\mathbf{T}_i$ and $\mathbf{T}_q$ does not add to the cost. However, $\mathbf{I}_i$ and $\mathbf{I}_q$ are not only populated, but some calculations are performed before population. All elements in $\mathbf{I}_i$ and $\mathbf{I}_q$ need L additions of two multiplicative terms. Also, the first and the last $L-1$ elements in $\mathbf{I}_i$ and $\mathbf{I}_q$ together contain $(L-1)^2$ α- and γ addition and/or subtraction terms. Therefore, the cost of populating $\mathbf{I}_i$ and $\mathbf{I}_q$ is given by

$$\sum_{n=1}^{2N}\sum_{m=1}^{L-1} m + 4\left(L-1\right)^2 = 2N(L-1) + 4(L-1)^2.$$

3. *Initialize* $\begin{bmatrix}\mathbf{s}_i \\ \mathbf{s}_q\end{bmatrix}$ *and* $\mathbf{u} = [\mathbf{u}_i^T|\mathbf{u}_q^T]$*, both of length* $2N$: Under the assumption that variable initialization does not add to the total cost, initialization of these variables does not add to the cost.

4. *Iterate the system* Z *times*:

   (a) *Calculate* $[\mathbf{u}_i^T|\mathbf{u}_q^T]^{(n)} = \begin{bmatrix}\mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i\end{bmatrix}\begin{bmatrix}\mathbf{s}_i \\ \mathbf{s}_q\end{bmatrix}^{(n-1)} + [\mathbf{I}_i^T|\mathbf{I}_q^T]$ Z times: The cost of multiplying a matrix of size $2N \times 2N$ with a vector of size $2N$ and adding another vector of length $2N$ to the result, Z times, is given by

   $$Z\left(\sum_{n=1}^{2N}\left(\sum_{m=1}^{2N} m\right) + 2N\right) = Z\left((2N)^2 + 2N\right) = (2N)^2 Z + 2ZN.$$

   (b) *Calculate* $[\mathbf{s}_i^T|\mathbf{s}_q^T]^{(n)} = g\left(\beta(n)[\mathbf{u}_i^T|\mathbf{u}_q^T]^{(n-1)}\right)$ Z times: The cost of calculating the estimation vector $\mathbf{s}$ of length $2N$ by using every value in state vector $\mathbf{u}$, also of length $2N$, assuming that the sigmoid function uses three instructions to execute, Z times, is given by[5]

   $$3 \times 2ZN = 6ZN$$

[5] It is assumed that the values of $\beta(.)$ are stored in a lookup table.

Thus, by adding all of the computations above, the total computational complexity for the M-QAM HNN MLSE equalizer is:

$$\begin{aligned} HNN - CC_{M-QAM} &= (2N)^2 Z + 2ZN + 6ZN + 2(L-1)^2 + \\ &\quad 2N(L-1) + 4(L-1)^2 \\ &= (2N)^2 Z + 8ZN + 2N(L-1) + 6(L-1)^2 \end{aligned} \tag{4.68}$$

The structure of the M-QAM HNN MLSE equalizer is identical to that of the BPSK HNN MLSE equalizer. The only difference is that, for the BPSK HNN MLSE equalizer, all matrices and vectors are of dimension N instead of $2N$, as only the in-phase components of the respective vectors are considered. Therefore it follows that the computational complexity of the BPSK HNN MLSE equalizer will be

$$BPSK - CC_{BPSK} = N^2 Z + 4ZN + N(L-1) + 3(L-1)^2. \tag{4.69}$$

4.6.2 Viterbi MLSE

The Viterbi equalizer performs the following steps:

1. *Setup trellis of length N, where each stage contains M^{L-1} states, where M is the constellation size:* It is assumed that this does not add to the complexity. It must however be noted that the dimensions of the trellis are $N \times M^{L-1}$.

2. *Determine two Euclidean distances for each node*: The *Euclidean distance* is determined by subtracting L addition terms, each containing one multiplication, from the received symbol at instant k. The cost for determining one *Euclidean distance* is therefore given by

$$2NM^{L-1}(2L+1) = 4LNM^{L-1} + 2NM^{L-1}.$$

3. *Eliminate contending paths at each node*: Two path costs are compared using an *if*-statement, assumed to count one instruction. The cost is therefore

$$2NM^{L-1}.$$

4. *Backtrack the trellis to determine the MLSE solution*: Backtracking across the trellis, where each time instant k requires an *if*-statement. The cost is therefore

$$NM^{L-1}.$$

Adding all costs to give the total cost results in

$$Viterbi - CC_{M-QAM} = 4LNM^{L-1} + 5NM^{L-1}.$$

4.6.3 HNN MLSE and Viterbi MLSE Comparison

Fig. 4.20 shows the computational complexities of the HNN MLSE equalizer and the Viterbi MLSE equalizer as a function of the CIR length L, for $L = 2$ to $L = 10$, where the number of iterations is $Z = 20$.

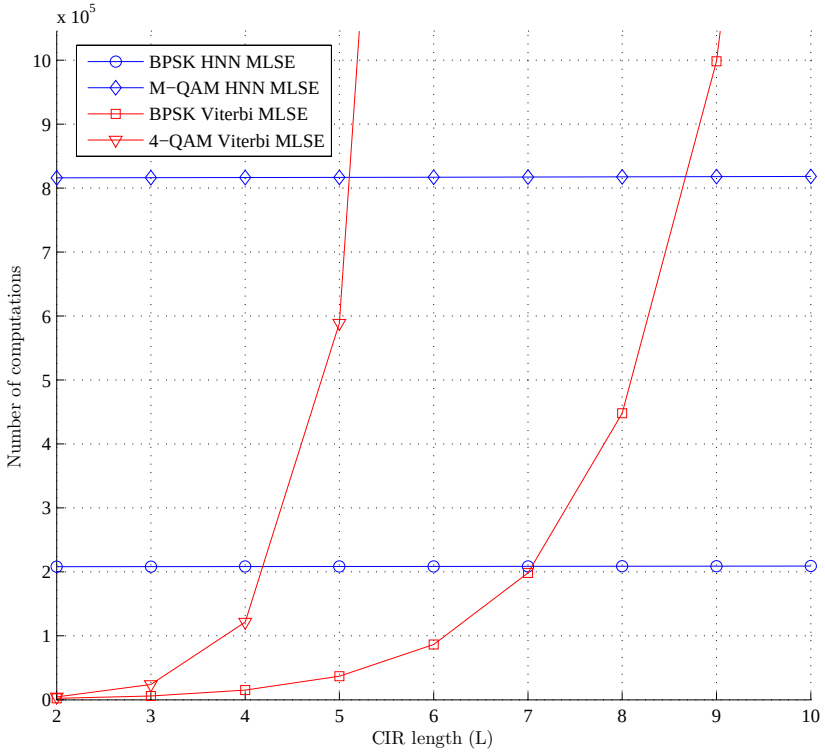

FIGURE 4.20: Computational complexity comparison between the HNN MLSE and Viterbi MLSE equalizers.

For the HNN MLSE equalizer, it is shown for BPSK and M-QAM modulation, since the computational complexities of all M-QAM HNN MLSE equalizers are equal. Also, for the Viterbi MLSE equalizer, it is shown for BPSK and 4-QAM modulation. It is clear that the computational complexity of the HNN MLSE equalizer is superior to that of the Viterbi MLSE equalizer. For BPSK the break-even mark between the HNN MLSE equalizer and the Viterbi MLSE equalizer is at $L = 7$, and for 4-QAM it is at $L = 4.2 \approx 4$.

The complexity of the HNN MLSE equalizer for both BPSK and M-QAM seems constant, whereas that of the Viterbi MLSE equalizer increases exponentially as the CIR length increases. Also note the difference in complexity between the HNN BPSK equalizer and the HNN M-QAM equalizer. This is due to the quadratic relationship between the complexity and the size of the vectors and matrices in the HNN MLSE equalizer. The HNN MLSE equalizer is however not well-suited for systems with short CIRs, as the complexity of the Viterbi MLSE equalizer is less than that of the HNN MLSE equalizer for short CIRs. This

is not a problem, since the focus of this study is on equalization of signals in systems with extremely long memory.

Fig. 4.21 shows the computational complexity of the HNN MLSE equalizer for BPSK and M-QAM modulation, for block lengths of $N = 100$ and $N = 500$ respectively, indicating the quadratic relationship between the computational complexity and the data block length, also requiring larger vectors and matrices in the HNN MLSE equalizer.

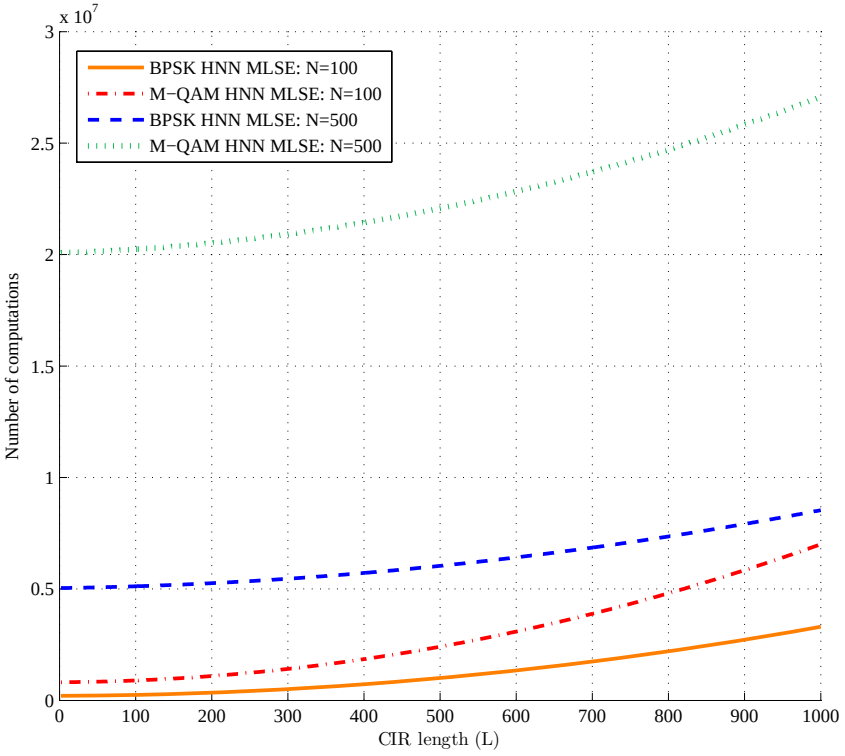

FIGURE 4.21: Computational complexity comparison for the HNN MLSE equalizers for different data block lengths.

It is clear that, as the data block length increases, the data block length N, rather than the CIR length L, is the dominant factor contributing to the computational complexity. However, due to the approximate independence of the complexity on the CIR length, for large data block length, the HNN MLSE equalizer is able to equalize signals in systems with hundreds of CIR elements. This should be clear from Fig. 4.20 and Fig. 4.21.

The scenario in Fig. 4.21 is somewhat unrealistic, since the data block length must at least be as long as the CIR length. However, Fig. 4.22 serves to show the effect of the data block length and the CIR length on the computational complexity of the HNN MLSE equalizer. Fig. 4.22 shows the computational complexity for a more realistic scenario. Here the complexity of the BPSK and M-QAM HNN MLSE equalizer is shown for $N = 1000$, $N = 1500$ and $N = 2000$, for $L = 0$ to $L = 1000$.

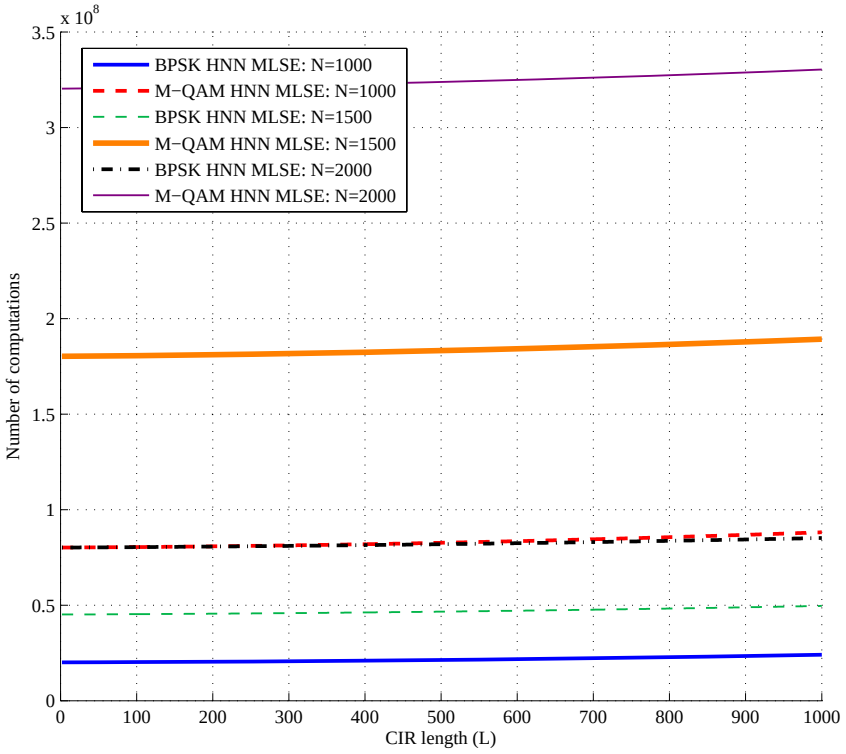

FIGURE 4.22: Computational complexity comparison for the HNN MLSE equalizers for realistic data block lengths.

From Fig. 4.22 it is clear that, as the data block length linearly increases, the computational complexity increases quadratically. It is quite significant that the complexity is nearly independent of the CIR length when the data block length is equal to or greater than the CIR length, which is the case in practical communication systems. It should now be clear why the HNN MLSE equalizer is able to equalize signals in systems, employing BPSK or M-QAM modulation, with hundreds and possibly thousands of resolvable multipath elements.

The complexity advantage of the HNN MLSE equalizer is obvious. Its low complexity makes it suitable for equalization of signals with CIR lengths that are currently beyond the capabilities of optimal equalizers. For conventional optimal equalizers, such as the Viterbi MLSE equalizer and the MAP equalizer, the computational complexity increases exponentially with an increase in the channel memory, and it is also exponentially related to the number of symbols in the modulation alphabet. On the other hand, the computational complexity of the HNN MLSE equalizer is quadratically related to the data block length and almost independent of the CIR length for realistic scenarios, where the data block length is larger than the CIR length. Also, the complexity of the HNN MLSE equalizer is independent of the modulation alphabet size for M-QAM systems, making it suitable for equalization in higher order M-QAM system with even moderate channel memory, where optimal equalizers cannot be applied.

4.7 Concluding Remarks

In this chapter the HNN MLSE equalizer was presented. It was shown how the equalizer is derived from first principles, by mapping the cost function that is minimized by the Viterbi MLSE equalizer to the energy function of the HNN. The iterative nature of the equalizer was explained, where it was shown how the HNN MLSE equalizer minimized the HNN energy function. Following this, the decision functions used in the BPSK and 16-QAM HNN MLSE equalizers were discussed, followed by a discussion on some optimization techniques to allow the HNN MLSE equalizer to produce more reliable source estimates. The chapter was concluded by performing a computational complexity analysis on the HNN MLSE equalizer, where its complexity was compared to that of the Viterbi MLSE equalizer.

The significant computational complexity of the HNN MLSE equalizer is the chief motivation behind this study. The fact that the HNN MLSE equalizer can be applied to almost any single-carrier scenario is profound. However, the quadratic relationship between its complexity and the data block length might limit the size of the data blocks that can be used with this equalizer. Nevertheless, the HNN MLSE equalizer offers a very promising approach to single-carrier MLSE equalization and, as will be shown in *Chapter 5*, it can be used as a low complexity alternative to the MAP equalizer in a Turbo Equalizer configuration.

Chapter 5

Applications and Results

Hopfield and Tank have shown that the HNN can be used to solve NP-complete optimization problems with much less computational complexity than other optimal algorithms [5]. This is due to the high parallelism of the HNN structure as well as the high level of interconnection between the neurons, making the HNN an attractive platform for MLSE equalization in systems with extremely long memory.

In *Section 4.6* it was shown that the computational complexity of the HNN MLSE equalizer is less than that of the Viterbi MLSE equalizer for single-carrier communication systems with moderate to high bandwidth. The HNN MLSE equalizer has computational complexity quadratic in the data block length and almost independent of the channel memory length. The approximate independence of the complexity from the channel memory is significant, as the CIR length is the dominant term in the complexity of all optimal equalizers. The computational complexity of these equalizers grows exponentially with a linear increase in the CIR length, i.e. $O(NM^L)$. Also, the complexity of all M-QAM equalizers realized with the HNN MLSE equalizer is identical, as the complexity is dependent only on the data block length and the number of dimensions of the constellation map, and only weakly dependent on the channel memory length.

In this chapter the HNN MLSE equalizer is evaluated for a number of applications. The low computational complexity of the HNN MLSE equalizer allows it to equalize signals in systems with extremely long memory, well beyond the capabilities of conventional optimal equalizers like the Viterbi MLSE and the MAP equalizers.

The HNN MLSE equalizer will first be simulated for a number of hypothetical scenarios to ensure that it performs as expected. It will be evaluated for short Rayleigh fading channels where its performance will be compared to that of the Viterbi MLSE equalizer. It will become clear that the performance of the HNN MLSE equalizer is worse than that of the Viterbi MLSE equalizer in short channels. The HNN MLSE equalizer will also be evaluated for long Rayleigh fading channels where its performance will be compared to that of an MMSE equalizer. It

will be established that the HNN MLSE equalizer outperforms the MMSE equalizer in long fading channels.

The HNN MLSE equalizer will also be evaluated for a number of applications: Underwater communication systems, CDMA communication systems, a multiple antenna communication system, and Turbo Equalization. It will be shown that the equalizer effectively mitigates the effect of hundreds of interfering symbols in underwater communication systems to yield reliable system performance at moderate to high data rates. It will also be demonstrated that the HNN MLSE equalizer can replace the RAKE demodulator in a CDMA system to perform chip-level equalization, as opposed to the naive approach of coherent energy recombination taken by the RAKE demodulator. The HNN MLSE equalizer will be shown to outperform the RAKE receiver by an order of magnitude, confirming the relevance of this equalizer in a CDMA communication system. The HNN MLSE equalizer will also be evaluated in a multiple transmit antenna configuration, designed to artificially extend the CIR in order to exploit the low complexity equalization ability of the equalizer. Results show a performance gain due to the time- and space diversity provided by this transmit antenna configuration. Finally, the HNN MLSE equalizer will be used as a low complexity alternative to the optimal MAP equalizer in a Turbo Equalizer to enable turbo equalization in systems with extremely long memory. It will be shown that turbo equalization using the HNN MLSE equalizer achieves performance gains, outperforming the MMSE-based Turbo Equalizer for high E_c/N_o values.

All simulations are performed via Monte Carlo simulation, calculating the BER for each E_b/N_0 value in the range of E_b/N_0 values considered. Simulations are terminated after $100,000$ data blocks have been processed or when $100,000$ bit errors are reached. If not otherwise stated, the communication system is simulated for a GSM mobile fading environment, where the carrier frequency is $f_c = 900$ MHz, the symbol period is $T_s = 3.7$ μs and the relative speed between the transmitter and receiver is $v = 3$ km/h. Error-correction coding is mostly used for evaluation of the Turbo Equalizer, using E_c/N_0 to determine the channel noise, where E_c is the energy per coded bit, as opposed to the energy per uncoded bit denoted by E_b used for simulation of uncoded systems. The following parameters are used in all simulations:

- **Modulation:** Indicates the signal constellation used for modulation.
- **Interleaver:** Indicates whether an interleaver is used. Note that an interleaver is mostly used for Turbo Equalization.
- **Uncoded block length (N_u):** This is the number of uncoded payload symbols contained in a data block.
- **Coded block length (N_c):** This is the number of coded payload symbols contained in a data block.
- **CIR length (L):** Indicates the length of the CIR, ie. number of resolvable multipath components plus one.

- **Channel Estimation:** Indicates whether channel estimation, as opposed to perfect channel state information (CSI), is used. When perfect CSI is assumed, the center value of each fading vector is taken as the estimate for the corresponding CIR tap.
- **Number of pilots:** This is the number of pilot symbols contained in each transmitted data block used for channel estimation.
- **Mobile speed (v):** Indicates the relative speed between the transmitter and receiver.[1]
- **Doppler frequency (f_D):** Indicates the Doppler frequency derived from the mobile speed.
- **PDP:** Indicates the power delay profile being used.

5.1 General Evaluation

5.1.1 Rayleigh Fading Simulator

Apart from a few exceptions, all simulations in the following sections are performed for frequency-selective Rayleigh fading channels. As such, it is important that the Rayleigh fading simulator is an accurate model of real-world fading in mobile communication systems. The Rayleigh fading simulator proposed in [37], as described in *Section 3.5.2.1*, is used here. When BPSK modulation is considered, only the real parts of the fading vectors produced by the simulator are used, and when M-QAM modulation is considered, both the real and imaginary components are used independently. The simulation parameters are as follows:

Parameter	**Setting**
Modulation	BPSK, 4-QAM
Interleaver	No
Uncoded block length (N_u)	200
Coded block length (N_c)	-
CIR length (L)	1
Channel Estimation	Perfect CSI
Number of pilots	-
Mobile speed (v)	0 km/h
Doppler frequency (f_D)	0 Hz
PDP	-

Fig. 5.1 shows the simulated BER curves for both BPSK and 4-QAM systems for flat fading, as compared to the theoretical BER curve. From Fig. 5.1 it is clear that the simulated BPSK- and 4-QAM performance is identical to the theoretical performance. Hence it is concluded that the fading simulator works correctly.

[1] Since direction and the angle of incidence are ignored here, speed is used rather than velocity.

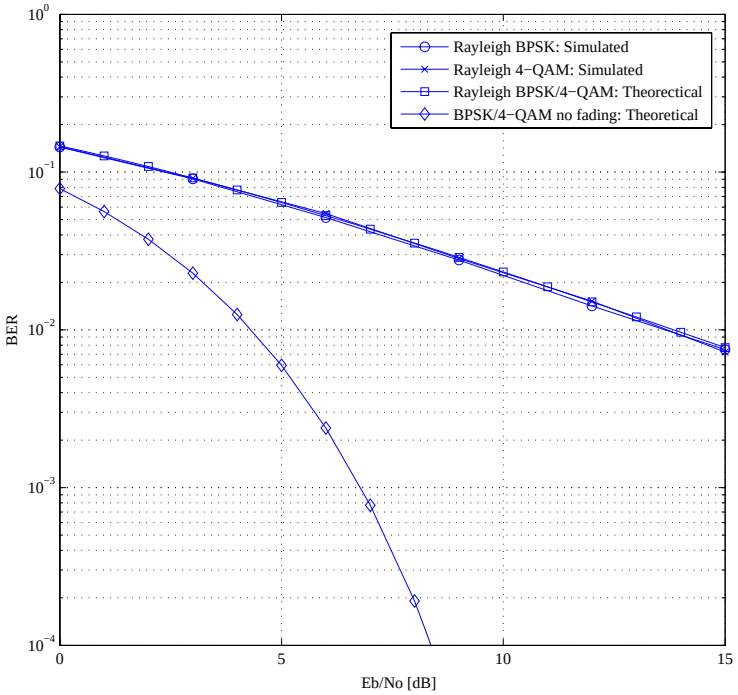

FIGURE 5.1: Rayleigh flat fading simulation results for BPSK and 4-QAM.

5.1.2 Short Channels

The performance of the HNN MLSE equalizer is compared to that of the Viterbi MLSE equalizer for short channels, using BPSK modulation, for CIR lengths from $L = 2$ to $L = 10$ corresponding to channel delays from 7.4 μs to 37 μs. The performance of the 16-QAM MLSE equalizer is also evaluated. The simulation parameters are as follows:

Parameter	**Setting**
Modulation	BPSK, 16-QAM
Interleaver	No
Uncoded block length (N_u)	200
Coded block length (N_c)	-
CIR length (L)	2, 4, 6, 8, 10
Channel Estimation	Yes/No
Number of pilots	$4L$
Mobile speed (v)	3 km/h
Doppler frequency (f_D)	2.5 Hz
PDP	Uniform

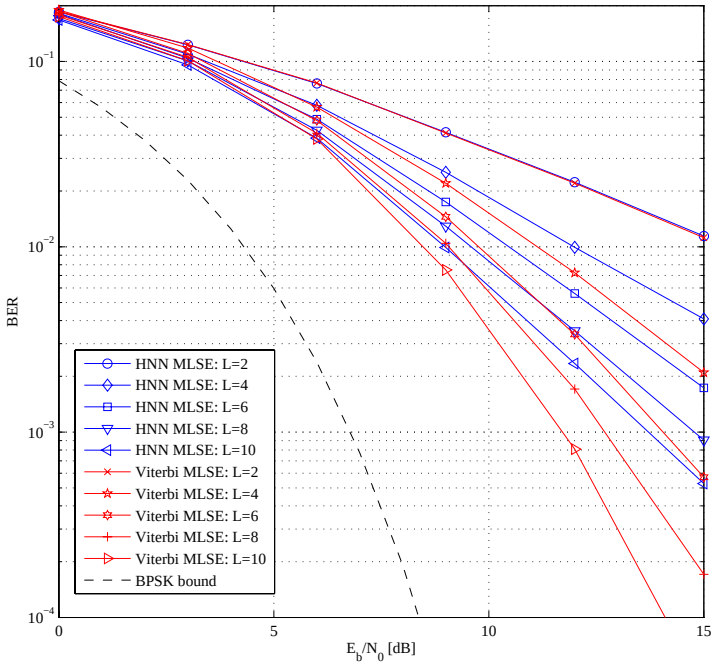

FIGURE 5.2: BPSK HNN MLSE and Viterbi MLSE equalizer performance comparison.

Fig. 5.2 shows the BER performance for the BPSK HNN MLSE equalizer for short channels, as compared to the performance of a BPSK Viterbi MLSE equalizer using channel estimation. The performance of the HNN MLSE equalizer is worse than that of the Viterbi MLSE equalizer for short channels. Also, Fig. 5.3 shows the performance of the BPSK HNN MLSE equalizer for channel estimation and perfect CSI, respectively. There is an approximate 2 dB performance loss due to channel estimation. Fig. 5.4 shows the BER performance of the 16-QAM MLSE equalizer for short channels, where perfect CSI was assumed. Here it is also clear that the performance of the 16-QAM HNN MLSE equalizer is worse than expected for short channels.

From the simulation results presented here it is concluded that the HNN MLSE equalizer is not suitable for use in systems with short channel memory. However, the equalization of signals in short channels is not the focus of this thesis, as the computational complexity of the HNN MLSE equalizer is greater than that of the Viterbi MLSE equalizer for short channels. The HNN MLSE equalizer is therefore not a good option in systems with short memory and for those cases where the Viterbi MLSE equalizer is the clear choice. By utilizing the HNN MLSE equalizer within a Turbo Equalizer configuration, however, BER performance can be increased. This will be demonstrated in *Section 5.5.2.2*.

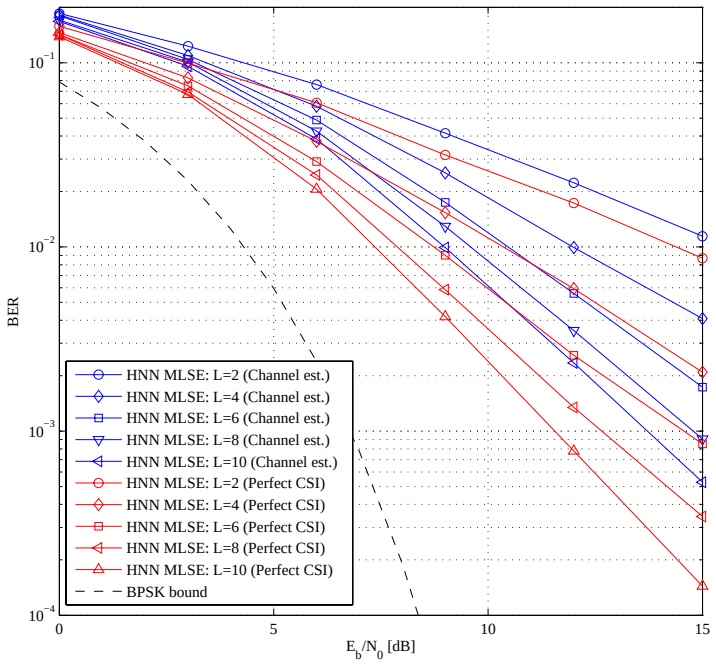

FIGURE 5.3: BPSK HNN MLSE equalizer performance for channel estimation and perfect CSI.

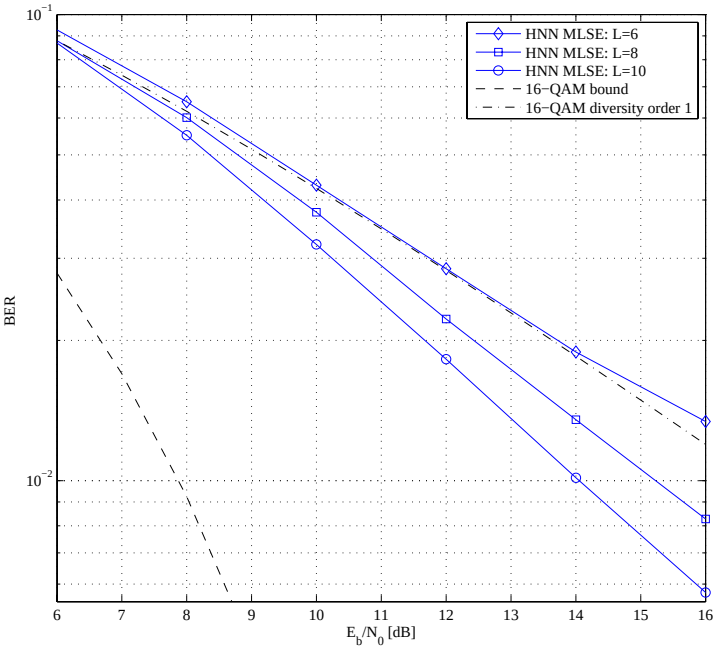

FIGURE 5.4: 16-QAM HNN MLSE equalizer performance in short channels.

5.1.3 Long Channels

Here the performance of the BPSK- and 16-QAM HNN MLSE equalizers is evaluated for long channels, for both perfect and imperfect CSI, with channel delays from 74 μs to 7.4 ms. The performance of the BPSK HNN MLSE equalizer is also compared to that of an MMSE equalizer for imperfect CSI. The simulation parameters are as follows:

Parameter	**Setting**
Modulation	BPSK, 16-QAM
Interleaver	No
Uncoded block length (N_u)	Unique for each case
Coded block length (N_c)	-
CIR length (L)	Moderate to extremly long
Channel Estimation	Yes/No
Number of pilots	Unique for each case
Mobile speed (v)	3 km/h
Doppler frequency (f_D)	2.5 Hz
PDP	Uniform

Fig. 5.5 shows the performance of the BPSK HNN MLSE for perfect CSI. Here the uncoded data block length is $N_u = 500$ and the CIR lengths range from $L = 10$ to $L = 500$, corresponding to channel delay spreads of 37 μs to 1.85 ms. As the channel memory increases, the performance increases correspondingly, approaching unfaded matched filter performance as a result of the effective time diversity provided by the multipath channels. It is clear the BSPK HNN MLSE equalizer performs near-optimal, if not optimal, equalization.

Fig. 5.6 shows the performance of the 16-QAM HNN MLSE equalizer, where again perfect CSI and an uncoded data block length of $N_c = 500$ are assumed. Here the CIR length ranges from $L = 25$ to $L = 400$ which corresponds to channel delay spreads of 92.5 μs to 1.48 ms. Here it is also clear that the performance approaches unfaded matched filter performance as the channel memory increases.

Fig. 5.7 shows the performance of the BPSK HNN MLSE equalizer compared to that of an MMSE equalizer, for moderate to long channels, using channel estimation. It is assumed that $3L$ training symbols are available for channel estimation per data block of length $N_u = 450$. For the HNN MLSE equalizer the channel estimator is used to estimate the CIR, but for the MMSE equalizer, however, channel estimation is performed during equalization, where the *filter smoothing length* is $3L$. From Fig. 5.7 it is clear that the HNN MLSE equalizer outperforms the MMSE equalizer for high E_b/N_0-levels.

From these results it is clear that the HNN MLSE equalizer effectively recombines the dispersed energy of the the transmitted symbols when perfect CSI is assumed. It is also evident that the HNN MLSE equalizer outperforms the MMSE equalizer for long channels.

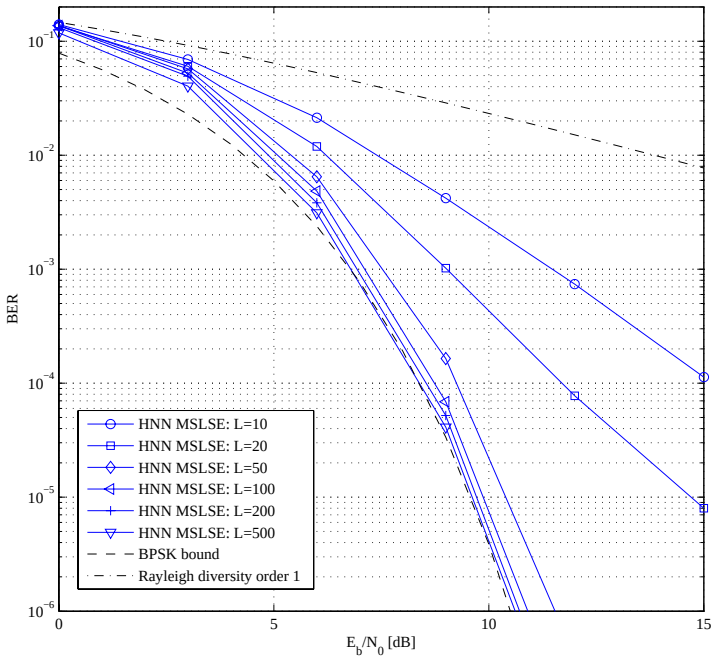

FIGURE 5.5: BPSK HNN MLSE equalizer performance in extremely long channels.

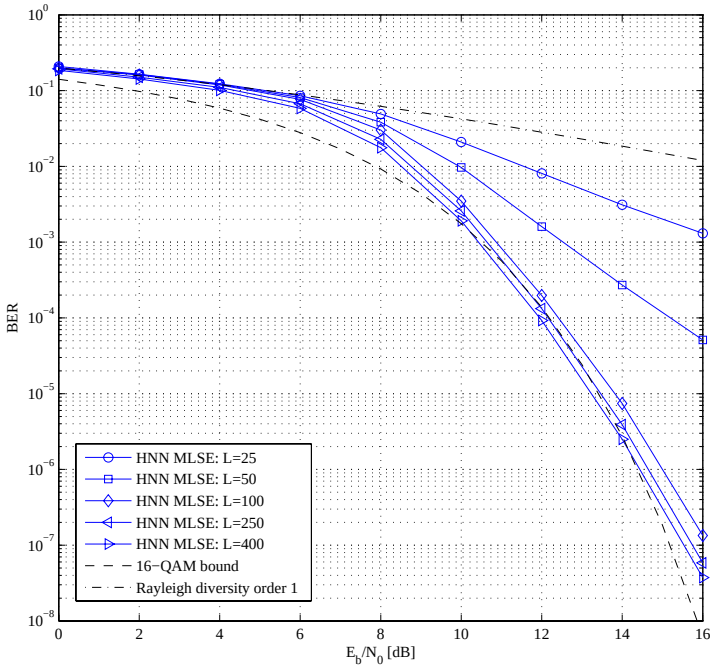

FIGURE 5.6: 16-QAM HNN MLSE equalizer performance in extremely long channels.

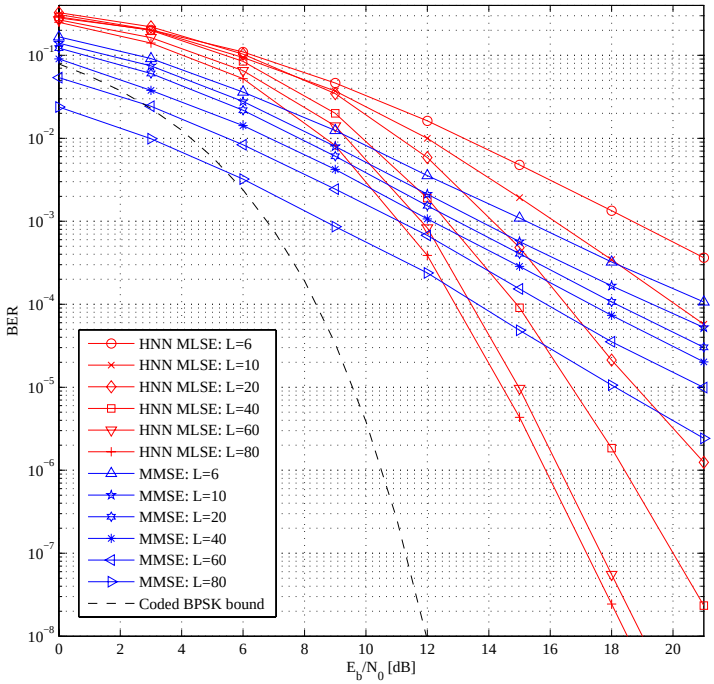

FIGURE 5.7: BPSK HNN MLSE and MMSE equalizer performance comparison.

5.1.4 Different Relative Mobile Speed

The performance of the BPSK- and 16-QAM HNN MLSE equalizers is evaluated for different relative mobile speeds, implying different Doppler frequencies, assuming perfect CSI and the linear PDP.

Parameter	**Setting**
Modulation	BPSK
Interleaver	No
Uncoded block length (N_u)	200
Coded block length (N_c)	-
CIR length (L)	20
Channel Estimation	Perfect CSI
Number of pilots	-
Mobile speed (v)	3, 50, 80, 120, 160, 200, 240 km/h
Doppler frequency (f_D)	2.5, 41.67, 66.67, 100, 133.33, 166.67, 200 Hz
PDP	Linear

Fig. 5.8 shows the performance of the BPSK HNN MLSE equalizer for the different relative speeds between the transmitter and the receiver. It is clear that the performance of the

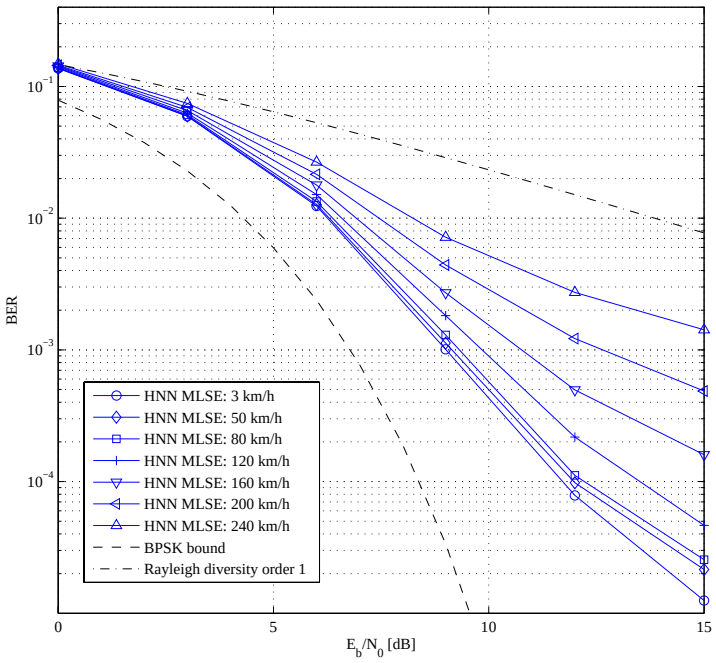

FIGURE 5.8: BPSK HNN MLSE equalizer performance for different Doppler frequencies.

HNN MLSE equalizer decreases as the Doppler frequency increases. This is expected, as the channel estimate becomes exceedingly inaccurate as the Doppler frequency increases, due to the corresponding faster fading rate. By shortening the duration of a data block, the effect of Doppler frequencies can be reduced.

5.2 Underwater Communication systems

5.2.1 Description

In underwater communication systems the information is sent via sound waves, which travel much slower than electromagnetic waves through free space. The losses incurred during transmission are characterized by transmission loss, noise, reverberation and temporal and spatial variability of the channel [47]. The bandwidth and range of such a system are mainly influenced by the transmission loss and noise, as well as the ISI in the systems.

Due to the nature of the UAC, the intersymbol interference caused by multipath may extend over hundreds of symbols for moderate to high data rates, for single carrier systems [7]. The rapidly varying nature of the UAC also introduces Doppler shifts, resulting in a fast varying CIR, complicating equalization even more [7]. To combat the effect of the severe multipath interference, various MMSE and DFE techniques have been proposed [7, 47, 48]. Multi-carrier

modulation techniques, such as orthogonal frequency OFDM [28], have also been proposed for use in the UAC, which is very susceptible to the high Doppler frequency shifts encountered in the UAC. Although the HNN MLSE equalizer does not perform well in fast-varying channels, it will be shown how it can be used to equalize signals in underwater communication systems with multiple hundreds of multipath elements.

Due to the low complexity equalization ability of the HNN MLSE equalizer it can be used to equalize signals in the UAC for relatively low Doppler frequencies. By using the HNN MLSE equalizer, the data rate can be increased dramatically, as the HNN MLSE equalizer is able to take advantage of the increase in the number of interfering symbols that are synonymous with an increase in bandwidth. The HNN MLSE equalizer does not need to be modified for use in a UAC.

5.2.2 Simulation

The HNN MLSE equalizer is evaluated for an underwater communication system using 4-QAM modulation, where different channel delay profiles are used. All simulations where carried out for a carrier frequency $f_c = 15$ kHz with symbol interval of $T_s = 50$ μs, for CIR lengths of $L = 250$ and $L = 1000$, corresponding to delay spreads of 12.5 ms to 50 ms, with an uncoded data block length of $N_u = 1200$ symbols.

Parameter	**Setting**
Modulation	4-QAM
Interleaver	No
Uncoded block length (N_u)	1200
Coded block length (N_c)	-
CIR length (L)	250, 1000
Channel Estimation	Perfect CSI
Number of pilots	-
Mobile speed (v)	1.8 km/h
Doppler frequency (f_D)	5 Hz
PDP	Linear, Random, Exponential

Figures 5.9 and 5.10 show the performance of the HNN MLSE equalizer for different decaying PDPs in an underwater communication system. It is clear that the 4-QAM HNN MLSE equalizer effectively equalizes the ISI-corrupted received symbols for systems with extremely long CIR lengths, recombining the signal energy spread by the UAC among the channels.

The performance of the equalizer is worst for the exponential PDP, since most of the signal energy is concentrated in the first few taps of the CIR. However, when the linear and random PDPs are considered, the performance improves significantly, indicating that the proposed equalizer performs well when the signal energy is spread among the CIR taps.

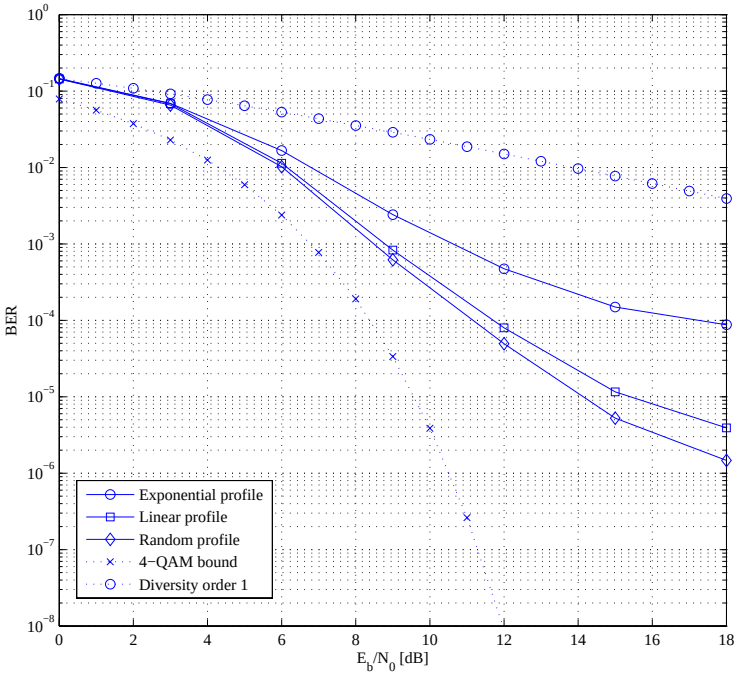

FIGURE 5.9: HNN MLSE equalizer performance for $L = 250$.

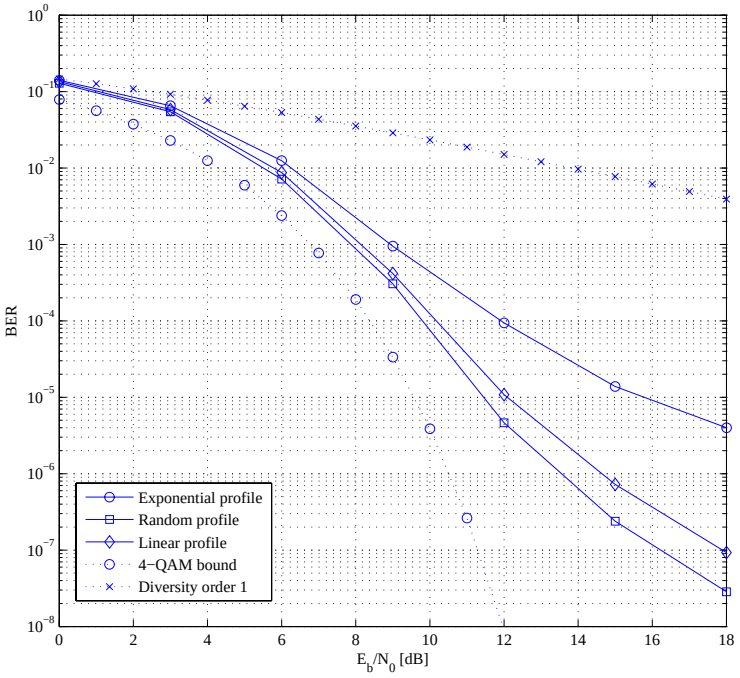

FIGURE 5.10: HNN MLSE equalizer performance for $L = 1000$.

5.3 CDMA Communication Systems

5.3.1 Description

The first use of spread spectrum systems was in the military because of the attractive property of hiding the signal below the noise floor during transmission. Spread spectrum signals are also resistant against narrowband jamming and interference and it also have a low probability of detection and interception. Spread spectrum is currently being used in cellular systems, where its bandwidth sharing and ISI rejection capabilities are desirable [10].

In CDMA systems the signal bandwidth is increased beyond what is necessary for effective communication by modulating the transmitted signal with a spreading sequence [10]. At the receiver the received signal is despreaded by correlating it with a synchronized copy of the spreading sequence. The length of the spreading sequence determines the factor by which signal bandwidth is increased, which also determines the factor by which the number of multipath elements, or the CIR length, increases (See *Section 3.6.2*). For a conventional unspreaded communication system with a CIR length of $L = 5$, the CIR length will increase to $L = 40$ when using a spreading code of length $M = 8$. The current solution is to apply a RAKE demodulator to coherently combine the different multipath components [4]. The argument for the RAKE demodulator is that the bulk of the energy is contained within the leading taps of the CIR. Although this is true, the RAKE demodulator only selects the strongest taps for equalization, therefore ignoring the energy in the remaining taps, however little it may be.

5.3.1.1 Using the HNN MLSE Equalizer for CDMA Equalization

The HNN MLSE equalizer can be used to perform chip-level equalization in CDMA communication systems, due to its low complexity equalization ability. The equalizer works directly on the received chips, producing ML estimates of the composite transmitted symbol sequence, after which the estimated chip sequence is correlated with each user's unique spreading sequence in order to retrieve the source information of each user. The decision function of the HNN MLSE equalizer is however modified slightly to enable it to produce soft outputs of the estimated composite signal.

When considering the composite signal in a CDMA communication system, the amplitude of the demodulated signal can assume different combinations. For instance, for a BPSK CDMA communication system, when the spreaded signals of two users are summed, the composite signal can assume three amplitude levels, namely -2 $(-1-1)$, 0 $(-1+1)$, and 2 $(1+1)$. Similarly, for three users, there are four possible amplitude levels namely -3 $(-1-1-1)$, -1 $(-1-1+1)$, 1 $(1+1-1)$ and 3 $(1+1+1)$.

To make the HNN MLSE equalizer suitable for equalization in CDMA systems, the decision function is scaled according to the maximum amplitude level in order to produce soft outputs

which will range between the minimum and maximum amplitude levels. It was observed that the performance of the system is better for a scaled bipolar decision function than for a scaled multi-amplitude decision function.

For two users, the decision function is therefore given by

$$g(u) = 2\left(\frac{2}{1+e^{-u}} - 1\right), \tag{5.1}$$

and for three users the decision function is given by

$$g(u) = 3\left(\frac{2}{1+e^{-u}} - 1\right). \tag{5.2}$$

The output of the HNN MLSE equalizer is a sequence of N estimates which are despreaded by each user's unique spreading code in order to produce the estimated transmitted sequence for each user.

5.3.2 Simulation

To evaluate the HNN MLSE equalizer in a CDMA communication system, BPSK modulation is used. The exponential power delay profile is employed to characterize the power delay of the channel. The signal energy is not only divided among the CIR taps, but also among the users. The energy is also normalized according to the processing gain. The system is evaluated at a carrier frequency of 900 MHz in a system where the symbol period, without spreading, is $T_s = 9.472$ μs, with channel delay spread of $\tau_{max} = 47.36$ μs, corresponding to a CIR length of $L = 5$. By spreading the symbol sequences of each user with spreading sequences, of length $P = 64$, of which each chip in the sequence has a duration $T_c = 0.148$ μs, (A.7) yields a spreading gain of $G = 64$, corresponding to a CIR of length $L = 320$. The uncoded data block length is $N_u = 640$ chips. The spreading sequences are chosen at random from a set of Walsh-Hadamard codes. Fig. 5.11 shows the performance for the proposed MLSE equalizer and the RAKE demodulator for 1, 2, and 3 users, respectively. It is evident that the HNN MLSE equalizer combines the energy spread by the channel more effectively than the RAKE. This is not surprising, since the RAKE crudely estimates the transmitted chips of the composite signal by recombining the energy of the strongest taps, while the MLSE equalizer uses all the CSI to decorrelate the ISI-corrupted received symbols to produce near-optimal estimates of the transmitted composite chip sequence.

Parameter	**Setting**
Modulation	BPSK
Interleaver	No
Uncoded block length (N_u)	640
Coded block length (N_c)	-
CIR length (L)	320
Channel Estimation	Perfect CSI
Number of pilots	-
Mobile speed (v)	0 km/h
Doppler frequency (f_D)	0 Hz
PDP	Exponential

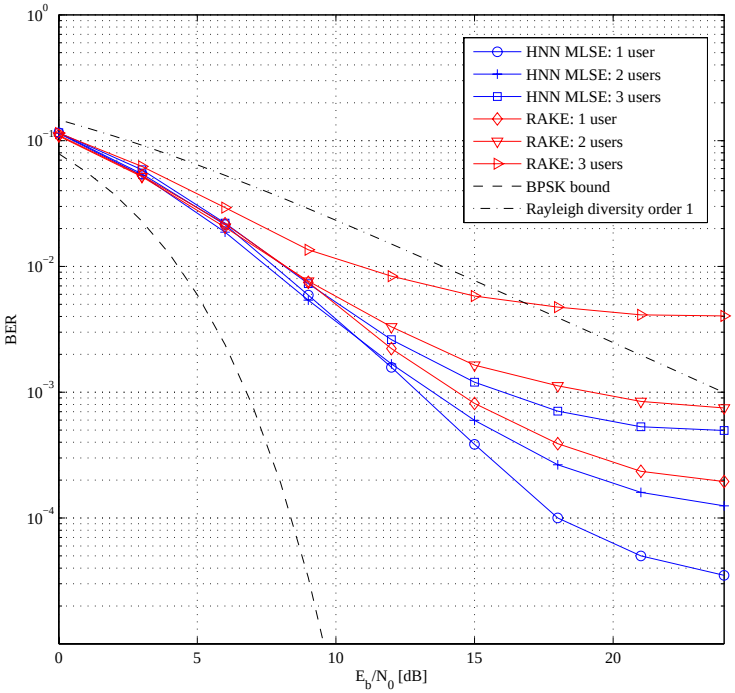

FIGURE 5.11: CDMA performance comparison between the HNN MLSE equalizer and the RAKE for 1, 2, and 3 users.

5.4 Multiple Antenna Scheme

5.4.1 Description

Multiple transmit antennas can be used to exploit the ability of the HNN MLSE equalizer to equalize signals in systems with hundreds of multipath elements. By using M transmit antennas, sufficiently spaced apart, the CIR can be artificially increased by a factor M by delaying the signal transmitted by each antenna by multiples of L symbol periods, corresponding to the duration τ_{max} of the channel delay spread.

Consider a conventional SISO GSM mobile environment, where the symbol period $T_s = 3.7$ μs, and the channel delay spread spans 12 symbols, yielding a total channel delay of $\tau_{max} = 44.4$ μs and a CIR length of $L = 12$ [41]. Given this setup, multiple transmit antennas can be used to artificially increase the CIR by delaying the signal transmitted by each antenna by multiples of $L = 12$ symbol periods. Therefore, without increasing the energy of each transmitted symbol, the ability of the HNN MLSE equalizer to equalize signals in systems with extremely long memory can be exploited to enhance system performance, due to the spatial diversity provided by multiple transmit antennas.

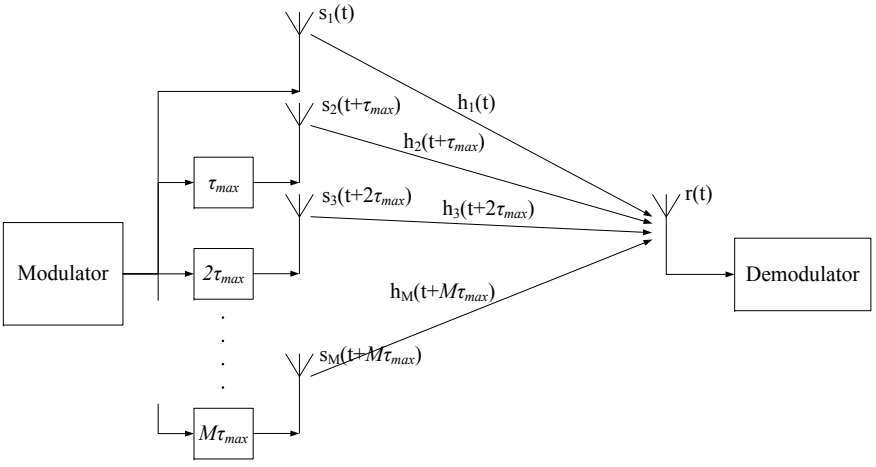

FIGURE 5.12: Proposed multiple transmit antenna setup.

Fig. 5.12 shows the proposed multiple transmit antenna setup, where the energy of the data-carrying signal is divided between M transmit antennas, where the transmitted signal is delayed by multiples of the channel delay spread τ_{max} according to the corresponding antenna. Assuming that the transmit antennas are sufficiently spaced apart each transmitted signal will fade independently. The minimum distance d_{min} between any transmit antenna must be at least as large as the distance the signal travels for the duration of the coherence time T_c of the signal. That is,

$$d_{min} \geq c.T_c, \tag{5.3}$$

where c is the speed of light. Each transmitted signal will therefore propagate through a unique channel, denoted by $h_i(t)$, $i = 1, 2, 3, \ldots, M$. Compared to a system employing only a single transmit antenna, this setup will artificially increase the CIR by a factor M, yielding an effective CIR length of

$$L_{eff} = M\left(\frac{\tau_{max}}{T_s}\right). \tag{5.4}$$

At the receiver, the extended CIR of length L_{eff} is estimated as usual. The HNN MLSE equalizer uses the estimated CIR of length L_{eff} together with the received symbol sequence to estimate the transmitted source symbols. The ability of the HNN MLSE equalizer enables it to equalize signals in a channel with and artificially extended CIR length to increase system performance.

5.4.2 Simulation

By using three multiple transmit antenna configurations, the multiple transmit antenna scheme is evaluated against a system with only one transmit antenna, using the uniform channel delay profile. To artificially extend the CIR, $M = 2$, $M = 5$ and $M = 10$ transmit antennas are used respectively, to yield CIR lengths of $L = 20$, $L = 50$ and $L = 100$.

Parameter	**Setting**
Modulation	4-QAM
Interleaver	No
Uncoded block length (N_u)	500
Coded block length (N_c)	-
CIR length (L)	10 (per antenna)
Channel Estimation	Perfect CSI
Number of pilots	-
Mobile speed (v)	3 km/h
Doppler frequency (f_D)	2.5 Hz
PDP	Uniform

Fig. 5.13 shows the performance for systems employing the multiple transmit antenna scheme compared to a normal single transmit antenna system. It is clear that the addition of transmit antennas achieves spatial diversity to artificially extend the CIR, thus exploiting the low complexity equalization ability of the HNN MLSE equalizer. When $L_{eff} = 100$, the HNN MLSE equalizer recombines the energy spread across the channel to yield near-optimal 4-QAM performance.

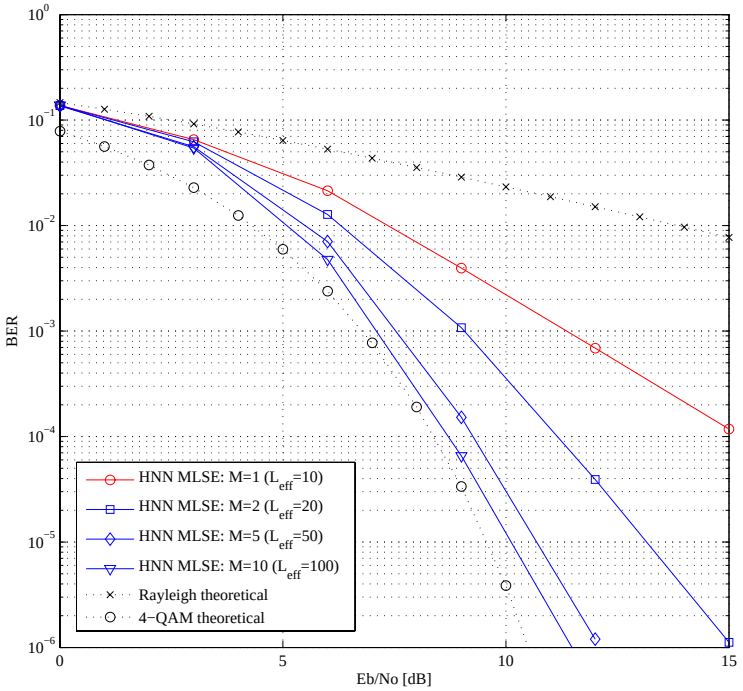

FIGURE 5.13: Multiple transmit antenna performance comparison, for $M = 1$, $M = 2$, $M = 5$, and $M = 10$, using iterative MLSE equalizer.

5.5 Turbo Equalization

5.5.1 Description

Due to the ability of the HNN MLSE equalizer to produce soft outputs, it can be used to replace the MAP equalizer in a Turbo Equalizer. This is an promising prospect, because the superior computational complexity of the HNN MLSE equalizer can be merged with the excellent performance gains achievable by employing Turbo Equalization. The merger will result in substantial performance gains at low computational cost, allowing for the turbo principle to be applied to systems with extremely long memory.

The HNN MLSE equalizer derived in *Chapter 4* is not able to take advantage of prior information as in the case of the MAP equalizer. Also, the HNN MLSE equalizer does not naturally produce marginalized symbol estimates necessary for Turbo Equalization, which will lead to self-feedback loops, ultimately limiting system performance.

5.5.1.1 Soft Outputs

To enable the HNN MLSE equalizer to be used in a Turbo Equalizer configuration, it must be able to produce soft outputs. To allow the HNN MLSE equalizer to produce soft outputs,

the input to the decision function is scaled by a factor $K = 0.5$ (See *Section 4.4*), which will allow the outputs to settle between the discrete decision levels instead of being forced to settle on the decision levels when $K = 1$.

5.5.1.2 Using Priors

To allow the HNN MLSE equalizer to use prior information provided in the form of extrinsic information, the algorithm is slightly modified. Consider the input vector $\mathbf{I}$ of length N for BPSK, and $2N$ for M-QAM, as derived in *Section 4.2*. Only the BPSK MLSE equalizer is considered here. As stated in *Section 4.3.2*, $\mathbf{I}$ contains the best linear estimate for $\mathbf{s}$. Using the intrinsic information as prior information on each coded symbol, each element I_k is biased as follows:

$$I_k.exp\left(s_\xi L(\hat{s_k})\right), \tag{5.5}$$

where $s_\xi = -1$ if I_k is negative and $s_\xi = 1$ if I_k is positive, and $L(\hat{s_k})$ is the prior probability of symbol s_k expressed as an LLR. As explained in *Section 3.7.2* and *Section 3.7.5.1*, the prior will confirm I_k when $L(\hat{s_k})$ is in agreement with I_k, or otherwise reject it. This slight modification to the algorithm allows the HNN MLSE equalizer to take advantage of prior probabilistic information of the symbols to be estimated.

5.5.2 Simulation

In this section the HNN MLSE equalizer is used in a Turbo Equalizer configuration as a low complexity replacement for the optimal MAP equalizer. This Turbo Equalizer will from now on be referred to as the HNN TE. The performance of the proposed Turbo Equalization setup is compared to that of a MMSE-based Turbo Equalizer, from now on referred to as the MMSE TE, where an MMSE equalizer is used to replace the MAP equalizer.

Parameter	**Setting**
Modulation	BPSK
Interleaver	Yes
Uncoded block length (N_u)	Unique for each case
Coded block length (N_c)	Unique for each case
CIR length (L)	Short and long
Channel Estimation	Unique for each case
Number of pilots	-
Mobile speed (v)	3 km/h
Doppler frequency (f_D)	2.5 Hz
PDP	Unique for each case

5.5.2.1 HNN MLSE Equalizer with Hard and Soft Outputs

Fig. 5.14 shows the performance of the HNN MLSE equalizer for hard and soft outputs respectively using BPSK modulation, where the HNN MLSE equalizer is followed by the rate 1/3 convolutional decoder considered in this thesis (See *Section 3.3* and *Section 3.8*). Here the coded data block length is $N_c = 300$, the channel length is $L = 20$, and $4L$ pilot symbols are used for channel estimation. It is clear that the soft output HNN MLSE equalizer achieves a substantial performance gain over the hard output HNN MLSE equalizer. This gain is made possible due to the fact that the soft input decoder exploits the soft input provided by the HNN MLSE equalizer in order to yield more reliable source estimates.

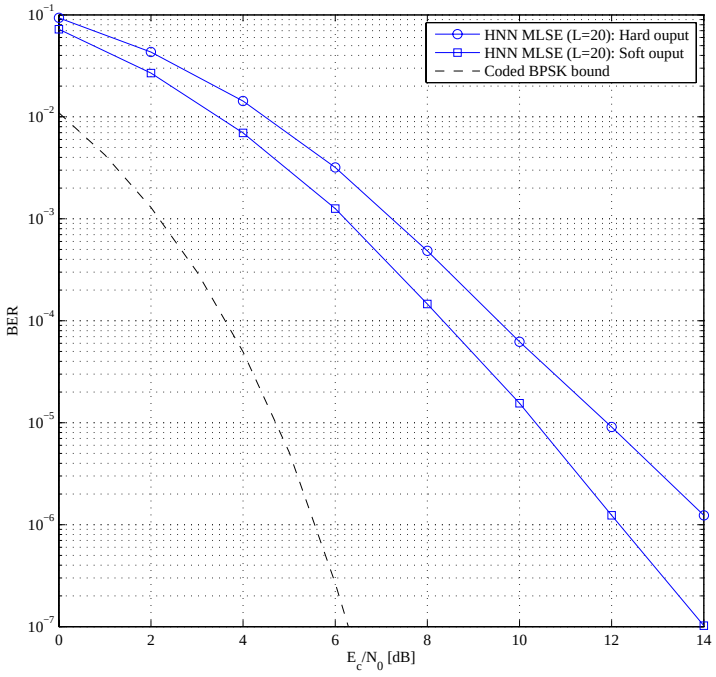

FIGURE 5.14: Hard and soft ouput HNN MLSE equalizer performance comparison.

5.5.2.2 Turbo Equalizer Validation

To ensure that the Turbo Equalizer functions correctly, it is first evaluated using the MAP equalizer in the static channel $\mathbf{h} = \{0.227, 0.460, 0.688, 0.460, 0.277\}$ in [4]. The coded block length is $N_c = 300$ and the system is simulated for a number of iterations. Fig. 5.15 shows the BER graphs for a number of turbo iterations. As expected, the performance is increased with each iteration, approaching perfect AWGN performance as the number of turbo iterations increases.

From Fig. 5.15 it is clear that the Turbo Equalizer, using the MAP algorithm for equalization and decoding, performs as expected. It is therefore concluded that the Turbo Equalizer

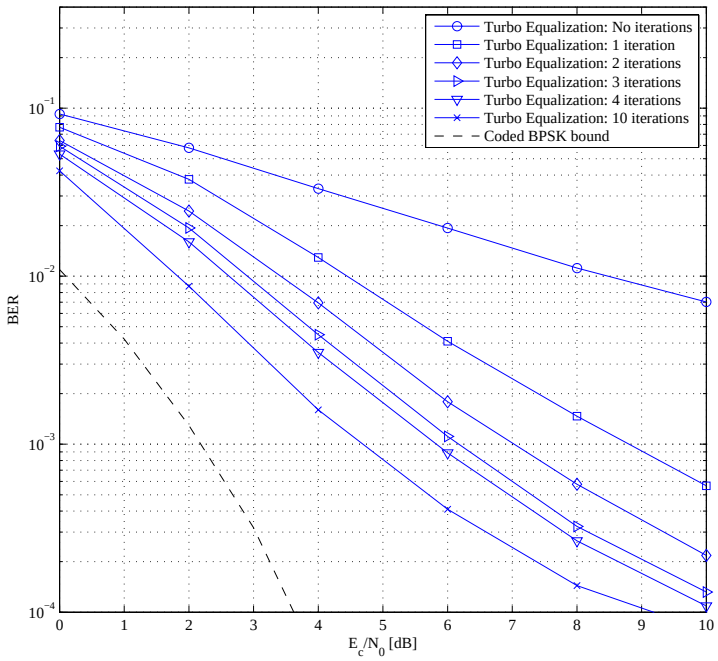

FIGURE 5.15: Turbo Equalizer performance for channel $\mathbf{h} = \{0.227, 0.460, 0.688, 0.460, 0.277\}$.

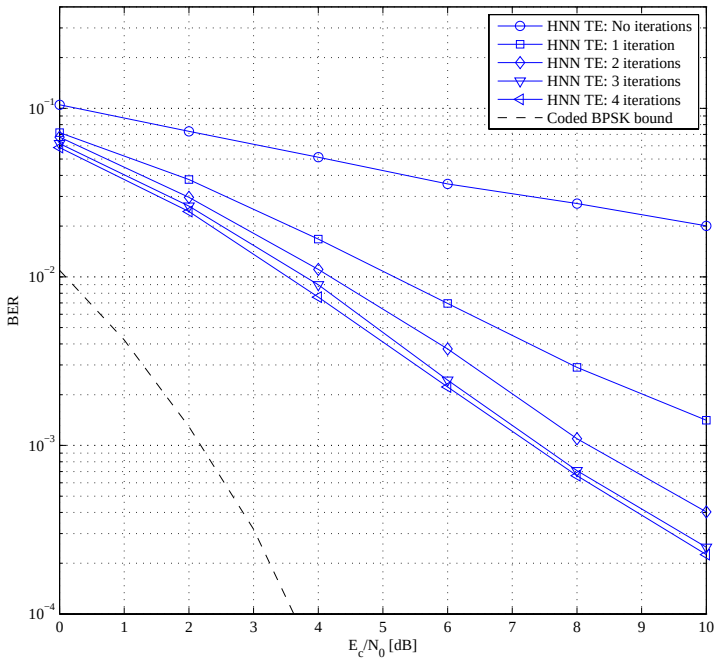

FIGURE 5.16: HNN TE performance for channel $\mathbf{h} = \{0.227, 0.460, 0.688, 0.460, 0.277\}$.

configuration, which will serve as a basis for the HNN TE and the MMSE TE, is correct. Fig. 5.16 shows the performance of the HNN TE for the same static channel considered above. As was established earlier (See *Section 5.1.2*), the HNN MLSE equalizer does not perform well in short channels, but Fig. 5.16 is proof that the HNN TE reduces the BER with each turbo iteration. The lacking performance of the HNN MLSE equalizer in short channels can be compensated for by using the HNN TE.

5.5.2.3 HNN Turbo Equalizer Performance in Rayleigh Fading Channels

The HNN MLSE equalizer is evaluated in long and short Rayleigh fading channels. The performance of the HNN TE is evaluated for fading channels of various lengths, assuming perfect and imperfect CSI. The performance of the HNN TE is compared to that of the MMSE TE for imperfect CSI.

Imperfect CSI

Here the HNN TE is evaluated for channel lengths of $L = 5$, $L = 10$, $L = 20$, $L = 50$, and $L = 100$. The coded block length is $N_c = 450$ and the number of pilot symbols used for channel estimation is $4L$. Here the *Uniform* PDP is used.

Fig. 5.17 shows the performance of the HNN TE for theses channels using no turbo iterations. These results are presented to use as reference for Fig. 5.18, which shows the performance of the HNN TE for one and two turbo iterations respectively. Recalling that no significant performance gain is achieved for more that two turbo iterations in Rayleigh fading channels [12], Fig. 5.18 shows the performance using one and two turbo iterations for each channel respectively, where the dashed line indicates one iterations, and the dashed-and-dotted line indicates two iterations. From Fig. 5.18 it is clear that the performance of the HNN TE increases as the number of turbo iterations are increased. This is a pleasing result, as it confirms that the HNN MLSE equalizer can be used as a low complexity alternative to the MAP equalizer in a Turbo Equalizer, enabling the Turbo Equalizer to equalize signals in systems with extremely long Rayleigh fading channels. It is however noted that the performance gain achieved is less significant for long channels, due to high levels of diversity.

Perfect CSI

The HNN TE is evaluated for channel lengths of $L = 25$, $L = 50$ and $L = 200$, assuming perfect CSI. The coded data block length is $N_c = 1200$ and the *Uniform* PDP is used once again. Fig. 5.19 shows the performance of the HNN TE, where it is clear that the first turbo iteration achieves a substantial performance gain for each channel. The performance

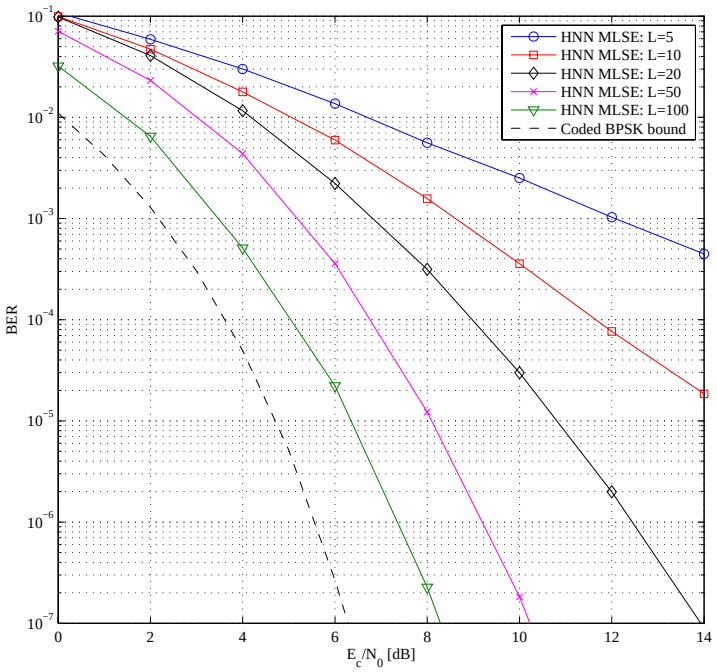

FIGURE 5.17: HNN TE performance for imperfect CSI using no turbo iterations.

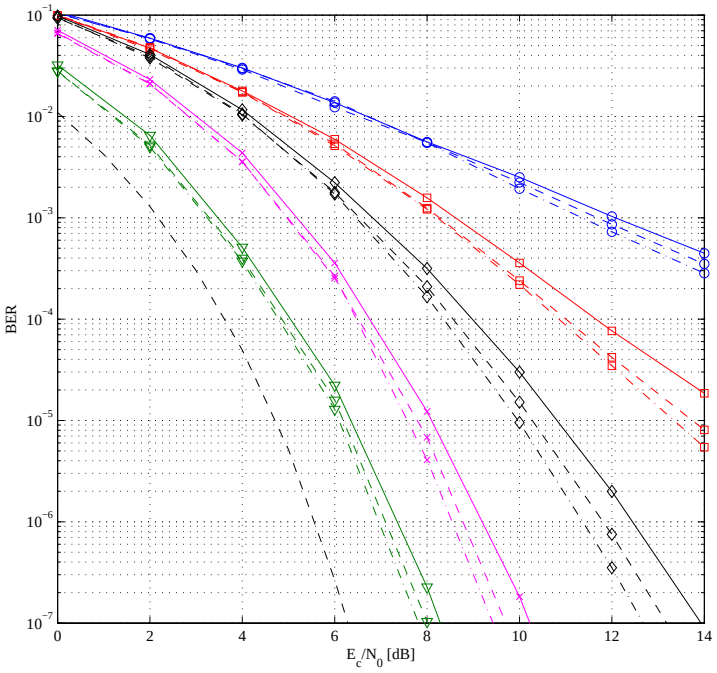

FIGURE 5.18: HNN TE performance for imperfect CSI (See Fig. 5.17 for reference).

increase provided by the second iteration is insignificant. As before, it is clear that performance approaches the uncoded BPSK bound as the number of resolvable multipath elements increases, where Turbo Equalization adds an extra performance advantage.

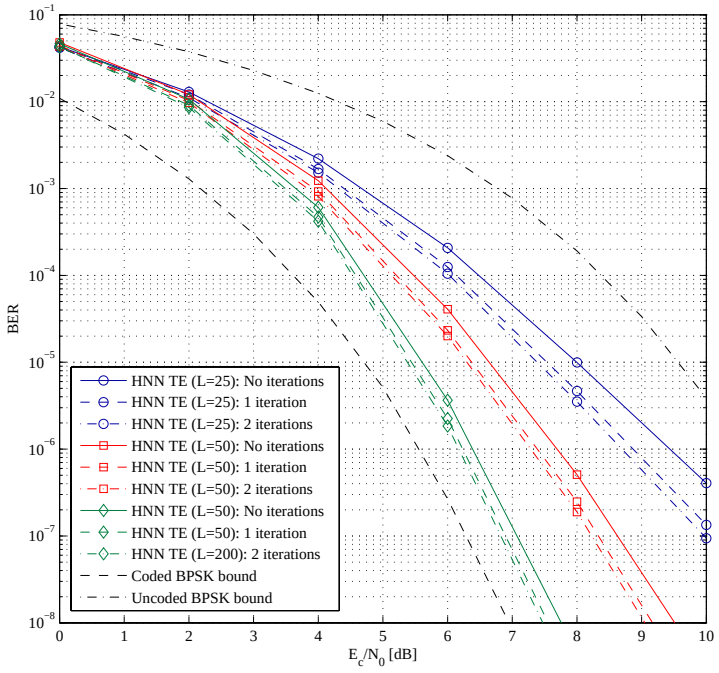

FIGURE 5.19: HNN TE performance of perfect CSI.

HNN TE and MMSE TE Performance Comparison

Here the performance of the HNN TE and the MMSE TE is compared for long channels, where again $4L$ pilot symbols are used for channel estimation. Fig. 5.20 shows the performance of these equalizers for channel lengths of $L = 10$, $L = 20$ and $L = 50$ using no turbo iterations. Again this is to establish a reference for what follows. Here it can be clearly seen that the MMSE TE performs better than the HNN TE for low E_c/N_0 levels. At high E_c/N_0 levels, however, the HNN TE outperforms the MMSE TE. Fig. 5.21 again shows the performance of the HNN TE and the MMSE TE, but now for one and two turbo iterations respectively. It is clear that the MMSE TE reduces the BER with the first two iterations. The HNN also reduces the BER with the first iteration and adds another reduction with the second iteration. From Fig. 5.20 and Fig. 5.21 it is clear that the HNN TE outperforms the MMSE TE for high E_c/N_0 values, as it aims to estimate the ML sequence estimates and does not attempt to linearly recombine the received symbols as in the case of the MMSE TE.

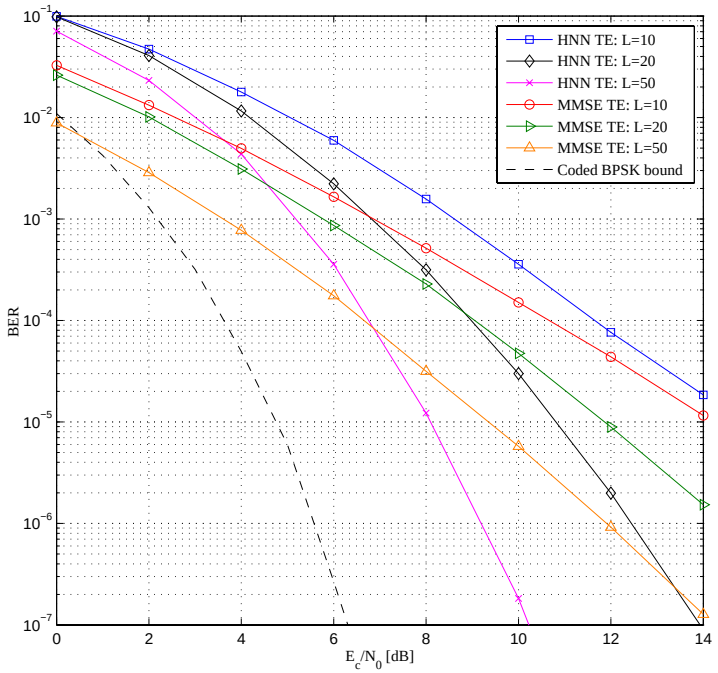

FIGURE 5.20: HNN TE and MMSE TE performance using no turbo iterations.

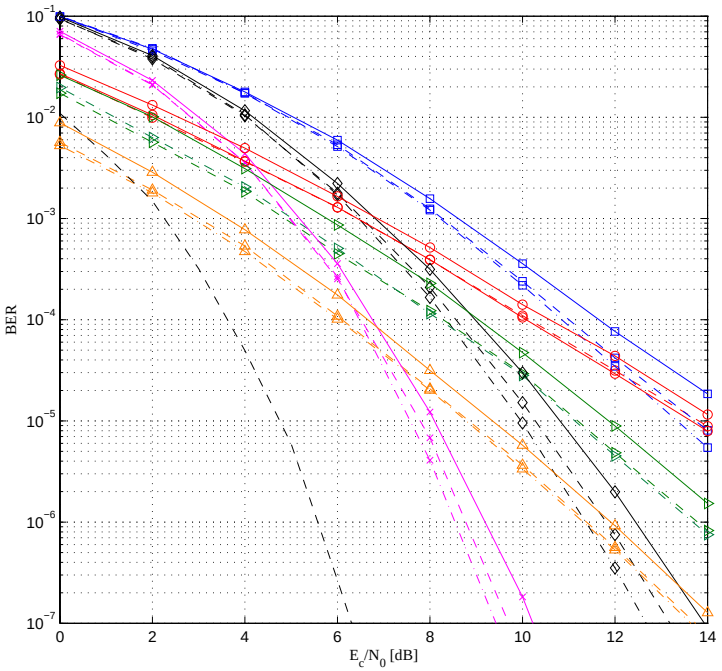

FIGURE 5.21: HNN TE and MMSE TE performance (See Fig. 5.20 for reference).

5.6 Concluding Remarks

In this chapter the HNN MLSE equalizer was subjected to intensive evaluation via computer simulation. It was first evaluated for short fading channels, where it became clear that its performance is not desirable in short channels. The reason for its poor performance in short channels is due to the low level of interconnection of the neurons in the HNN due to sparse population of the **T** matrix, which is also the correlation matrix (See *Section 4.2*). It also became clear that the performance of the HNN MLSE equalizer is affected by an increase in the Doppler frequency, resulting in poor channel estimation due to time-invariant channels.

Three applications of the HNN MLSE equalizer were discussed and evaluated. It was shown that this equalizer is able to equalize signals in the UAC, where there can possibly be hundreds to thousands of interfering symbols. It was seen that, when the energy of the channel is not spread evenly across the channel, the performance of the HNN MLSE equalizer is degraded. It was also demonstrated that the HNN MLSE equalizer outperforms the RAKE demodulator in CDMA communication systems by performing chip-level equalization. A multiple transmit antenna scheme was also proposed to artificially extend the channel length in order to exploit the low complexity equalization ability of the HNN MLSE equalizer to equalize signals in extremely long channels, yielding substantial performance gains.

Finally the HNN MLSE equalizer was used to replace the MAP equalizer in a Turbo Equalizer. It was evident that the HNN TE was able to reduce the BER for all the scenarios considered.

It was also shown that the HNN TE outperforms the MMSE TE for high E_c/N_0 levels. Apart from the fact that the performance of the HNN MLSE equalizer is already very good, its integration with Turbo Equalization allows for added performance gains.

This concludes the evaluation of the HNN MLSE equalizer. The results provided herein are self-evident, confirming the near-optimal low complexity equalization ability of the HNN MLSE equalizer in extremely long channels for SC modulation.

Chapter 6

Conclusion

In this thesis a low complexity MLSE equalizer was developed with computational complexity quadratic in the data block length and approximately independent of the channel memory length. The HNN was used as the foundation due to its ability to solve NP-complete problems with very low computational cost compared to optimal optimization algorithms. Although the outcome of this study might seem to be merely of academic interest, this thesis showed several practical applications where the HNN MLSE equalizer made a solution possible.

Up until now it was thought impossible to optimally solve the problem of MLSE in systems with extremely long channel memory. For that reason DFE and MMSE solutions have been in widespread use, until the recent interest in MC communication systems where severe ISI is not a major concern. These systems however have their own unique challenges, where Doppler frequency shifts and PAPR are among the most challenging problems.

As suggested earlier, the HNN MLSE equalizer may well find its use in MC communication systems in years to come. With ever-increasing data rates the problem of ISI will rear its head once again, which will leave communication system designers with yet another challenge. By employing a HNN MLSE equalizer for each subcarrier, data rates can be increased dramatically by taking advantage of frequency diversity provided by multiple carriers and time diversity due to multipath on each subcarrier.

It has been emphasized that the HNN MLSE equalizer produces near-optimal estimates of the transmitted symbols, and this claim was confirmed when the equalizer was evaluated for short channels. However, when the equalizer was evaluated for extremely long channels for perfect CSI, the performance of the systems matched non-faded AWGN performance, providing enough evidence to assume that the equalizer performs optimally for extremely long channels. It is therefore assumed that the performance of the equalizer approaches optimality as the connection matrix is populated, when the signal energy is evenly spread across the channel.

The unification of the HNN MLSE equalizer with the Turbo Equalizer also allows for performance gains in systems with long memory. The popularity of the turbo principle used for Turbo Coding and Turbo Equalization spurred interest in such a combination, combining the desirable attributes of these algorithms to produce an algorithm that is even more capable.

The low computational complexity of the HNN MLSE equalizer is its most desirable aspect. It is, however, worrying that its computational complexity is quadratically related to the data block length, as opposed to the linear dependence of the complexities of the VA and MAP equalizers on the data block length. This may limit the computational feasibility of the HNN MLSE equalizer in systems transmitting extremely large blocks of data. Despite this, the complexity of the HNN MLSE equalizer is multiple orders less than that of the VA and the MAP equalizers due to the fact that its complexity is almost independent of the channel memory length for many practical systems, as demonstrated in this thesis.

6.1 Further Research

This study only focused on the development of the HNN MLSE equalizer, with little focus on optimization techniques and on the application of the equalizer to practical systems. There are a number of open-ended research questions that must be answered:

- The equalizer does not perform optimally in short channels. Improving the performance of the equalizer is therefore a promising research topic.
- As stated above, application of the equalizer to MC modulated communication systems is a promising prospect and is well worth investigating.
- Integration of the equalizer into current technologies will provide knowledge as to the potential of the equalizer in real-world systems.
- It is suspected that the equalizer can be adapted for use in multiple input multiple output (MIMO) systems, as ISI and multiple access interference (MAI) exhibit the same interference effects. ISI achieves time diversity while MAI achieves spatial diversity.

Nevertheless, the development of the HNN MLSE equalizer is a major step forward in the field of single-carrier MLSE equalization in extremely long channels. This equalizer, as well as the potential research that might emanate from this work, have the potential to contribute greatly to wireless communications in future.

Appendix A

Spread Spectrum Principles

In multi-user spread spectrum communication systems each user is assigned a unique sequence, known to the transmitter and the receiver, which is used to modulate their data. This process is called spreading. In CDMA systems these sequences are chosen from a set of orthogonal codes, among which Walsh-Hadamard codes and Gold codes are often used. All the users' spreaded signals are superimposed to yield a composite signal that is transmitted. Upon reception the composite signal is demodulated and matched filtered, after which coherent combining via a RAKE demodulator mitigates the ISI in the received signal. The resulting signal is despreaded, using each user's spreading sequence and integrated, to yield the original data transmitted by that user.

A.1 Interference Rejection

A set of M linearly independent signals $c_i(t) = c_1(t), c_2(t), \cdots, s_M(t)$ can be written using a basis function representation as [10]

$$c_i(t) = \sum_{j=1}^{M} c_{ij}\varphi_j, \tag{A.1}$$

where $0 \leq t \leq T$ and the basis functions φ_j are orthanormal and span an N-dimensional space, where one of these signals is transmitted every T seconds. At the receiver M correlators, corresponding to each unique $c_i(t)$, are used where the output of the receiver is the signal corresponding to the maximum correlator output [10].

If each $c_i(t)$ is generated using a zero-mean random number generator, the energies of the respective signals will be distributed over the N-dimensional signal space. Assume that the total energy of the combined signal is E_s. Now, if a signal occupies this space, the same signal

space

$$J(t) = \sum_{j=1}^{N} J_j \varphi_j, \tag{A.2}$$

with a total energy of

$$E_J = \sum_{j=1}^{N} J_j^2 = \int_0^T J^2(t), \tag{A.3}$$

is used in an attempt to jam the transmitted signal. It is assumed the signal $c_i(t)$ is transmitted and, neglecting noise, the received signal is

$$r(t) = c_i(t) + J(t). \tag{A.4}$$

Then the output of the ith correlator in the receiver is

$$r_i(t) = \int_0^T r(t)c_i(t)dt = \sum j = 1^N \left(c_{ij}^2 + J_j c_{ij}\right), \tag{A.5}$$

where the first term represents the signal and the second term represents the interference. It can be shown that the signal-to-interference ratio (SIR) is [10]

$$\frac{E_s}{E_J} \times \frac{N}{M}. \tag{A.6}$$

By spreading the interference power over a larger dimension N than the required signalling dimension M, the SIR is increased by $G = N/M$, where G is called the processing or spreading gain. In practice spread spectrum systems have processing gains on the order of $100 - 1000$ [10].

A.2 Spread Spectrum Communication

In order for the source data of multiple users to be multiplexed onto one frequency, each user's source symbols of duration T_s are spread by that user's unique spreading sequence, where the duration of the spreading sequence is equal to the symbol period T_s. However, the spreading sequences are series of pulses, or chips, where each chip has a duration T_c. The ratio between the symbol duration and the chip duration is

$$G = \frac{T_s}{T_c}. \tag{A.7}$$

Assume that K different users are transmitting data on the same frequency, where each user is assigned a unique spreading sequence $c_i^{(j)}$ of length P, where $i = 1, 2, ..., P$ indicates the elements in the sequence and $j = 1, 2, ..., K$ indicates the user number. Assuming that the kth BPSK information symbol of user j is denoted by $b_k^{(j)}$, then the received signal in a Gaussian channel can be expressed as

$$z_k = \sum_{j=1}^{K} \sum_{i=1}^{P} b_k^{(j)} c_i^{(j)} + n_k, \tag{A.8}$$

where n_k is a zero mean σ^2 variance Gaussian noise sample. When the composite CDMA signal is subject to multipath, the received signal is expressed as

$$r_k = \sum_{j=0}^{L-1} h_j z_{k-j} + n_k, \tag{A.9}$$

and to recover the transmitted composite signal from the ISI-corrupted received signal, a RAKE demodulator is used (See *Section 3.7.4*) to coherently recombine the energy that was spread by the channel. The estimated BPSK source symbol k of user j is determined by the sign of

$$b_k^{(j)} = \sum_{i=1}^{P} c_i^{(j)} \left(\sum_{l=0}^{L-1} r_{k+l} h_l \right), \tag{A.10}$$

where, $\mathbf{h}$ is the CIR.

This concludes the discussion on spread spectrum systems.

Bibliography

[1] G.D. Forney. "Maximum Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference". *IEEE Transactions on Information Theory*, IT-18:363–378, May 1972.

[2] A.D. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". *IEEE Transactions on Information Theory*, IT–13(1):260–269, 1967.

[3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. "Optimal decoding of linear codes for minimizing symbol error rate". *IEEE Transactions on Information Theory*, IT-20:284–287, March 1974.

[4] J.G. Proakis. *Digital Communications*. McGraw-Hill, International Edition, New York, 4 th edition, 2001.

[5] J.J. Hopfield and D.W. Tank. "Neural computations of decisions in optimization problems". *Biology and Cybernetics*, 52:1–25, 1985.

[6] R.R. Lopes and J.R. Barry. "The Soft Feedback Equalizer for Turbo Equalization of Highly Dispersive Channels". *IEEE Transactions on Communications*, 54(5):783–788, May 2006.

[7] L. Freitag M. Stojanovic and M. Johnson. "Channel Estimation Based Adaptive Equalization of Underwater Acoustic Signals". *Oceans 1999*, 2:985–990, 1999.

[8] K. Zimmermann and K. Dostert. "A multipath model for the powerline channel". *IEEE Transactions on Communications*, 50(4):553–559, 2002.

[9] J.W.M. Bergmans. "Digital Baseband Transmission and Recording". *Springer*, 1996.

[10] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[11] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux. "Iterative Correction of Intersymbol Intereference: Turbo-Equalization". *European Transactions on Teleommunications*, 6:507–511, 1995.

[12] G. Bauch, H. Khorram, and J. Hagenauer. "Iterative Equalization and Decoding in Mobile Communication Systems". *Proceedings of European Personal Mobile Communications Conference (EPMCC)*, pages 307–312, 1997.

[13] R. Koetter, M. Tuchler, and A.C. Singer. "Turbo Equalization". *IEEE Signal Processing Magazine*, 21(1):67–80, 2004.

[14] R. Koetter, M. Tuchler, and A.C. Singer. "Turbo Equalization: Principles and New Results". *IEEE Transactions on Communications*, 50(5):754–767, 2002.

[15] H.C. Myburgh and J.C. Olivier. "Near-Optimal Low Complexity MLSE Equalization". *Wireless Communications and Networking Conference (WCNC)*, pages 226–230, 2008.

[16] H.C. Myburgh and J.C. Olivier. "Low Complexity Iterative MLSE Equalization in Highly Spread Underwater Acoustic Channels". *OCEANS*, 2009.

[17] H.C. Myburgh and J.C. Olivier. "Low Complexity Iterative MLSE equalization of M-QAM Signals in Extremely long Rayleigh Fading Channels". *EUROCON*, 2009.

[18] H.C. Myburgh and J.C. Olivier. "A Low Complexity Recurrent Neural Network MLSE Equaliser: Applications and Results". *South African Telecommunication, Networks and Applications Conference (SATNAC)*, 2009.

[19] H.C. Myburgh and L.P. Linde. "Reduced Complexity Combined Soft-Decision MLSE Equalization and Decoding". *Australasian Telecommunication, Networks and Applications Conference (ATNAC)*, 2008.

[20] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 2nd edition, 2003.

[21] D.J.C. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[22] J.C. Olivier. *Essential Digital Communication Theory*. ESF320 Lecture notes, 2007.

[23] D.D. Falconer and F.R. Magee. "Adaptive Channel Memory Truncation for Maximum Likelihood Sequence Estimation". *Bell Systems Technology Journal*, 52:1541–1562, November 1973.

[24] C.Y. Beare. "The Choice of the Desired Impulse Response in Combined Linear Viterbi Algorithm Equalizers". *IEEE Transactions on Communications*, 26:1301–1307, August 1978.

[25] W.U. Lee and F.S. Hill. "A Maximum Likelihood Sequence Estimator with Decision Feedback Equalizer". *IEEE Transactions on Communications*, 25:971–979, September 1977.

[26] A. Dual-Hallen and C. Hegaard. "Delayed decision feedback sequence estimation". *IEEE Transactions on Communnications*, 37(5):428–436, May 1989.

[27] S.R. Sauders and A.A. Zavala. *"Antennas and Propagation for Wireless Communication Systems*. Wiley, 2007.

[28] J. Terry and J. Heiskala. *OFDM Wireless LANs: A Theoretical and Practical Guide*. Sams Publishing, Indianapolis, IN, 2001.

[29] S.H. Bang, B.J. Shue, and C.H. Chang. "Maximum likelihood sequence estimation of communication signals by a Hopfield neural network". *IEEE International Conference on Neural Networks*, 5:3369–3374, July 1994.

[30] S.H. Bang, B.J. Shue, and J. Bing. "A Neural Network for Detection of Signals in Communication". *IEEE Transactions on Circuits and Systems*, 42(8):644–655, July 1996.

[31] J.D. Provence. "Neural network implementation for an adaptive maximum-likelihood receiver". *IEEE International Symposium on Circuits and Systems*, 3:2381–2385, June 1988.

[32] D.C. Chen, B.J. Sheu, and E.Y Chou. "A neural network communication equalizer with optimized solution capability". *IEEE International Conference on Neural Networks*, 4: 1957–1962, June 1996.

[33] C. Berrou, A. Glavieux, and P. Thitimajshima. "Near Shannon Limit Error-Correction and Decoding: Turbo-Codes (1)". *International Conference on Communications*, pages 1064–1070, 1993.

[34] B. Wichmann and D. Hill. "Building a random number generator". *Byte*, pages 127–128, 1987.

[35] C.Y. Beare. "An efficient method to calculate the free distance of convolutional codes". *International Conference on Electronics, Hardware, Wireless and Optical Communications (WSEAS)*, February 2007.

[36] C. Heegard and S. Wicker. "Turbo Coding". *Boston: Kluwer Academic Publishing*, 1999.

[37] Y.R. Zheng and C. Xiao. "Improved models for the generation of multiple uncorrelated Rayleigh fading waveforms". *IEEE Commununications Letters*, 6:256–258, June 2002.

[38] R.H. Clarke. "A Statistical Theory of Mobile-Radio Reception". *Bell Systems Technical Journal*, 47:957–1000, 1968.

[39] G. Marsaglia and T.A. Bray. "A convenient method for generating normal variables". *SIAM Rev.*, 6:260–264, 1964.

[40] L. Staphorst. "Viterbi decoded linear block codes for narrowband and wideband wireless communication over mobile fading channels". *Master's thesis, University of Pretoria*, 2005.

[41] R. Steele, C. Lee, and P. Gould. *GSM, cdmaOne and 3G Systems*. John Wiley Sons Ltd, 2001.

[42] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, USA, 1988.

[43] W.H. Gerstacker and R. Schober. "Equalization concepts for EDGE". *IEEE Transactions on Wireless Communications*, 1(1):190 – 199, January 2002.

[44] S.H. Bang, O.T. Chen, J.C. Chang, and B.J. Sheu. "Paralleled hardware annealing in multilevel Hopfield neural networks for optimal solutions". *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 42(1):46–49, January 1995.

[45] S.H. Bang, J.C. Chang, and B.J. Sheu. "Search of optimal solutions in multi-level neural networks". *IEEE International Symposium on Circuits and Systems (ISCAS)*, 6:423–426, June 1994.

[46] U. Halici. *Artificial Neural Networks*. EE 543 Lecture notes, 2004.

[47] M. Stojanovic. *"Acoustic Communications" in Encyclopedia of Telecommunications*. John Wiley and Sons, 2003.

[48] J.C. Preisig L. Weichang. "Estimation and Equalization of Rapidly Varying Sparse Acoustic Communication Channels". *Oceans 2006*, pages 1–6, 2006.