LOW COMPLEXITY TURBO EQUALIZATION USING SUPERSTRUCTURES

by

Hermanus Carel Myburgh

Submitted in partial fulfilment of the requirements for the degree

Philosophiae Doctor (Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology
UNIVERSITY OF PRETORIA

25 June 2013

To Clarise

ACKNOWLEDGEMENTS

The author of this thesis would like to thank the following persons:

- Professor J.C. Olivier, who has been my study leader and mentor since my undergraduate final year project. I want to thank him for his guidance and thoughtful advice over the years, which have shaped me as a young researcher.

- My wife for her continuous loving support and motivation throughout this endeavor. Her persistent encouragement inspired me to see it through, although at times the end seemed very far away. I love her so much.

- God Almighty who has given me a sound mind and the ability to envision innovative solutions to complex problems. Without His presence in my life, this, as well as everything leading up to it, would not have been possible.

SUMMARY

LOW COMPLEXITY TURBO EQUALIZATION USING SUPERSTRUCTURES

by

Hermanus Carel Myburgh

Promoter(s):	Prof J.C. Olivier and Dr A.J. van Zyl
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Philosophiae Doctor (Engineering)
Keywords:	Equalization, decoding, turbo equalization, low complexity, superstructure, iterative, Bayesian network, Hopfield neural network, belief propagation, interleaver, Cholesky, Markov process.

In a wireless communication system the transmitted information is subjected to a number of impairments, among which inter-symbol interference (ISI), thermal noise and fading are the most prevalent. Owing to the dispersive nature of the communication channel, ISI results from the arrival of multiple delayed copies of the transmitted signal at the receiver. Thermal noise is caused by the random fluctuation on electrons in the receiver hardware, while fading is the result of constructive and destructive interference, as well as absorption during transmission. To protect the source information, error-correction coding (ECC) is performed in the transmitter, after which the coded information is interleaved in order to separate the information to be transmitted temporally.

Turbo equalization (TE) is a technique whereby equalization (to correct ISI) and decoding (to correct errors) are iteratively performed by iteratively exchanging extrinsic information formed by optimal posterior probabilistic information produced by each algorithm. The extrinsic information determined from the decoder output is used as prior information by the equalizer, and vice versa, allowing for the bit-error rate (BER) performance to be improved with each iteration. Turbo equalization achieves excellent BER performance, but its computational complexity grows exponentially with an increase in channel memory as well as with encoder memory, and can therefore not be used in dispersive channels where the channel memory is large. A number of low complexity equalizers have consequently been

developed to replace the maximum a posteriori probability (MAP) equalizer in order to reduce the complexity. Some of the resulting low complexity turbo equalizers achieve performance comparable to that of a conventional turbo equalizer that uses a MAP equalizer. In other cases the low complexity turbo equalizers perform much worse than the corresponding conventional turbo equalizer (CTE) because of suboptimal equalization and the inability of the low complexity equalizers to utilize the extrinsic information effectively as prior information.

In this thesis the author develops two novel iterative low complexity turbo equalizers. The turbo equalization problem is modeled on superstructures, where, in the context of this thesis, a superstructure performs the task of the equalizer and the decoder. The resulting low complexity turbo equalizers process all the available information as a whole, so there is no exchange of extrinsic information between different subunits. The first is modeled on a dynamic Bayesian network (DBN) modeling the Turbo Equalization problem as a quasi-directed acyclic graph, by allowing a dominant connection between the observed variables and their corresponding hidden variables, as well as weak connections between the observed variables and past and future hidden variables. The resulting turbo equalizer is named the dynamic Bayesian network turbo equalizer (DBN-TE). The second low complexity turbo equalizer developed in this thesis is modeled on a Hopfield neural network, and is named the Hopfield neural network turbo equalizer (HNN-TE). The HNN-TE is an amalgamation of the HNN maximum likelihood sequence estimation (MLSE) equalizer, developed previously by this author, and an HNN MLSE decoder derived from a single codeword HNN decoder. Both the low complexity turbo equalizers developed in this thesis are able to jointly and iteratively equalize and decode coded, randomly interleaved information transmitted through highly dispersive multipath channels.

The performance of both these low complexity turbo equalizers is comparable to that of the conventional turbo equalizer while their computational complexities are superior for channels with long memory. Their performance is also comparable to that of other low complexity turbo equalizers, but their computational complexities are worse. The computational complexity of both the DBN-TE and the HNN-TE is approximately quadratic at best (and cubic at worst) in the transmitted data block length, exponential in the encoder constraint length and approximately independent of the channel memory length. The approximate quadratic complexity of both the DBN-TE and the HNN-TE is mostly due to interleaver mitigation, requiring matrix multiplication, where the matrices have dimensions equal to the data block length, without which turbo equalization using superstructures is impossible for systems employing random interleavers.

LIST OF ABBREVIATIONS

AWGN	Additive white Gaussian noise
BCJR	Bahl Cocke Jelinek Raviv
BER	Bit-error rate
BDFE	Block decision feedback equalizer
BP	Belief propagation
BPSK	Binary phase shift keying
CDMA	Code division multiple access
CIR	Channel impulse response
CTE	Conventional turbo equalizer
DAG	Directed acyclic graph
DF	Decision feedback
DBN-TE	Dynamic Bayesian network turbo equalizer
DFE	Decision Feedback Equalizer
ECC	Error-correction coding / Error control coding
EEG	Electroencephalograph
EXIT	Extrinsic information transfer
FIR	Finite impulse response
HNN	Hopfield neural network
HNN-TE	Hopfield neural network turbo equalizer
IRR	Infinite impulse response
ISI	Inter-symbol Interference
LCTE	Low complexity turbo equalizer
LDPC	Low density parity check
LLR	Log-likelihood ratio
LS	Least squares
MAP	Maxim a posteriori probability
MRI	Magnetic resonance imaging
M-QAM	M-ary quadrature amplitude modulation
ML	Maximum likelihood

MMSE	Minimum mean squared error
MMSE-LE	Minimum mean squared error linear equalizer
MMSE-DFE	Minimum mean squared error decision feedback equalizer
MSE	Mean squared error
NI-JED	Noniterative joint equalizer and decoder
PDA	Probabilistic data association
PDP	Power delay profile
QAM	Quadrature amplitude modulation
SDFE	Soft decision feedback equalizer
SFE	Soft feedback equalizer
SISO	Soft input soft output
SNR	Signal-to-noise ratio
SOVA	Soft output viteribi algorithm
TE	Turbo equalizer
TSP	Traveling salesman problem
VA	Viterbi algorithm

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context

In a wireless communication system information is transmitted through a multipath communication channel, where multiple delayed copies of the transmitted signal arrive at the receiver. Inter-symbol interference (ISI) is a phenomenon caused by the dispersive nature of a multipath wireless communication channel, which introduces memory to the system. In order to reverse the effect of the multipath channel on the transmitted information, an equalizer is used to detect or estimate the transmitted information with maximum confidence [1, 2].

During transmission the information is subjected to a number of impairments, of which multipath propagation, thermal noise and channel fading are the most prevalent. Fading is caused by the constructive and destructive interference of the signal during transmission, while thermal noise is caused by random fluctuations of electrons in the electronic components in the receiver. In order to mitigate the effect of these impairments on the transmitted symbols, the source information is encoded before transmission. Error-correction coding (ECC) is used to introduce controlled redundancy to the uncoded information to allow for the correction of errors that might have occurred during transmission and reception. Decoding is performed after detection or equalization in the receiver and can greatly improve the system performance compared to a system that does not employ ECC [2–4]. Equalization and decoding of convolutional codes are performed by using an optimal soft-input soft-output (SISO) maximum a posteriori probability (MAP) algorithm, adapted to each task, which ensures optimal performance [2, 3].

Turbo equalization is a technique whereby equalization and decoding are performed in an iterative fashion, where information is exchanged between the equalizer and the decoder, improving the overall bit-error rate (BER) with each iteration [2,5–7]. The computational complexity of turbo equalization, however, becomes an issue with an increase in channel memory and encoder memory. In fact, the computational complexity of turbo equalization grows exponentially with an increase in channel and encoder memory. As a result of the high computational complexity due to channel memory, much research has been devoted to the development of low complexity equalizers, which can be used to replace the MAP equalizer in order to reduce the overall complexity of the turbo equalizer [8–11]. Although these low complexity equalizers result in a significant reduction in computational complexity, the resulting turbo equalizer performance may be suboptimal in some cases.

It is therefore of interest to the field of turbo equalization applied to wireless communication systems to explore different modeling approaches to produce more elegant and efficient algorithms. In this thesis the author develops two new joint equalization and decoding algorithms, which can be used to replace conventional turbo equalizers. These new algorithms are developed by modeling the turbo equalization problem holistically using a single model, and not as separate equalization and decoding solutions that interact iteratively. The thesis shows that this approach leads to elegant solutions with excellent BER performance and favorable computational complexity characteristics.

1.1.2 Research Gap

Whereas the MAP equalizer is based on a structure that allows optimal equalization, the low complexity equalizers developed as a replacement for the MAP equalizer are modeled as minimization problems. Minimum mean squared error (MMSE) and decision feedback (DF) techniques, and combinations thereof, are used to design these low complexity equalizers, at the cost of optimality [7,8,10–12]. The iterative exchange of information in a turbo equalizer also results in suboptimal performance, since suboptimal decisions are made in response to incomplete information during early iterations. This is true even when an optimal MAP equalizer is used, and it is only after many iterations that performance (using MAP detection) will approach levels that may be considered to be acceptable.

There is therefore an opportunity to develop a turbo equalizer that solves the turbo equalization problem using one superstructure,[1] similar to the approach followed for turbo decoding in [13], and not two separate structures that exchange information in order to improve the overall system perform-

[1] In this context a superstructure is a structured algorithm that performs the task of both the equalizer and the decoder.

ance. The thesis instead proposes to use a single superstructure, and two specific cases are proposed to model a turbo equalizer. The first is a dynamic Bayesian network (DBN), and the second a Hopfield neural network (HNN). Both these methods have the potential to use all the available information and process it as a whole, without having to exchange information between constituent parts.

1.2 RESEARCH OBJECTIVES AND QUESTIONS

The objective of the research in this thesis is to stay true to the structured approach that enables processing of all available information as a whole without the need to exchange information between the equalizer and the decoder. By integrating the equalizer and decoder, computational complexity reduction comparable to that of turbo equalizers employing MMSE based equalizers must be achieved, while BER performance must remain comparable to that of a conventional turbo equalizer (CTE). Some research questions to consider are:

- Is it possible to use superstructures to equalize and decode information jointly and achieve the same results as a CTE?
- Will a joint approach result in lower computational complexity while still achieving acceptable performance?
- What will be the effect of an interleaver used in the transmitter? Will there be a limitation on the structure of the interleaver and will the interleaver affect computational complexity?

Throughout this thesis the answers to these questions will become clear as the research objectives are achieved.

1.3 HYPOTHESIS AND APPROACH

The research undertaken was supported by a number of factors that indicated that successful development and implementation of a new kind of turbo equalizer can be achieved, using two different approaches.

1.3.1 Hypothesis

It is hypothesized that a DBN and an HNN can be used to model the turbo equalization problem in order to equalize and decode coded and interleaved transmitted information jointly, and to achieve performance that is comparable to the performance achievable with a CTE, but with reduced complexity. The hypothesis is based on a number of previous works:

1.3.1.1 Dynamic Bayesian Network Turbo Equalizer (DBN-TE)

- It was shown in [14] that optimal, non-iterative joint equalization and decoding can be performed on a super-trellis, when a depth-limited block interleaver is used. The computational complexity grows exponentially with an increase in channel memory, encoder memory and interleaver depth. This implementation is infeasible for systems employing an interleaver with a depth of multiple codeword lengths, and impossible when a random interleaver is used.

- The optimal MAP equalizer and decoder uses belief propagation (BP) on a trellis in order to estimate the posterior distributions of the transmitted symbols [2, 3, 15]. BP is also used in DBNs to infer beliefs regarding particular states of nodes in a solution space, using evidence from neighboring nodes [16].

- It was shown in [13, 17] that turbo decoding can be performed by iteratively decoding the received codewords on a graph with cycles using BP. It was also shown in [18] that turbo decoding is an instance of Pearl's BP algorithm [15]. Because of the similarities between turbo equalization and turbo decoding, these works suggest that turbo equalization is in fact possible using a DBN or a graph.

1.3.1.2 Hopfield Neural Network Turbo Equalizer (HNN-TE)

- It was shown by this author in [19, 20] that the HNN can be used to equalize M-ary quadrature amplitude modulation (M-QAM) modulated information in single carrier systems transmitting information through highly dispersive multipath channels, with computational complexity that is approximately independent of the channel memory length and quadratic in the data block length.

- It has also been demonstrated that the HNN could be used to decode a special class of check codes, namely balanced codes [21, 22], where the HNN algorithm is applied to each codeword received. To perform turbo equalization the codeword sequence, and not each separate codeword, will have to be estimated.

Previous work therefore suggests that joint equalization using superstructures could be possible, and that a DBN and an HNN can be used as frameworks. The fact that a random interleaver, as is often used in wireless communication systems, might have an adverse effect on the computational complexity of the envisioned turbo equalizers, and may in fact inhibit the development thereof, remains an area of concern. Given the successful development of the non-iterative joint equalizer and decoder proposed in [14], the thesis shows that successful development and implementation can be achieved with some limited restrictions on the structure of the interleaver, which affects the computational complexity as well.

1.3.2 Approach

The approach that will be followed for the development of the two new turbo equalizers will be as follows:

1.3.2.1 DBN-TE

- The turbo equalization problem will be graphically modeled using a directed cyclic or acyclic graph, and system equations will be derived from the model.
- State transition probabilities will be determined for a given convolutional encoder, which will be used to describe the probabilities of transitioning from one state to the next in the DBN.
- The encoder outputs will be determined for each encoder state transition, which will be used to determine the cost of transitioning from on state to the next in the DBN.
- The equalizer and decoder cost functions of the respective MAP equalizer and decoder algorithms will be combined to enable the DBN to perform equalization while decoding.
- The effect of different interleavers - block interleavers and random interleavers - will be investigated and it will be attempted to adapt the model to allow for the use of any interleaver.

- Simulations will be performed to study the convergence properties as well as the performance of the proposed DBN-TE.

1.3.2.2 HNN-TE

- The HNN maximum likelihood sequence estimation (MLSE) equalizer in [19,20] and the HNN decoder in [21,22] will be implemented for binary phase shift keying (BPSK) modulation.
- The HNN decoder will be modified to decode sequences of codewords in a data block, since the HNN decoder is only able to decode single codewords.
- The HNN equalizer and the HNN decoder will be integrated so that joint equalization and decoding can be performed using one HNN rather than two separate HNNs.
- Like the DBN-TE, the HNN-TE will be adapted to be able to equalize and decoded interleaved information jointly.
- Simulations will be performed to study the convergence properties as well as the BER performance of the proposed HNN-TE.

1.4 RESEARCH GOALS

The research presented in this thesis aims to achieve the following goals:

- To derive and develop two novel joint equalizers/decoders, or turbo equalizers, each using one superstructure to perform joint equalization and decoding without exchanging information between separate algorithms.
- To ensure that the computational complexity characteristics of the proposed turbo equalizers are comparable to those of current MMSE based turbo equalizers, to enable turbo equalization in systems transmitting information through highly dispersive multipath channels.
- To enable the proposed turbo equalizer to be used in systems were the information is interleaved using a random interleaver, without limitations to the interleaver structure.
- To ensure that the performance of the proposed turbo equalizers is comparable to that of a CTE,

using an optimal MAP equalizer/decoder pair in static and mobile fading channels.

1.5 RESEARCH CONTRIBUTION

In this thesis two novel techniques for iterative joint equalization and decoding are developed using a DBN and an HNN as frameworks. These turbo equalizers are unique in the sense that they do not serve to replace the equalizer in a turbo equalizer setup, like other low complexity equalizers, but to replace both the equalizer and the decoder. They are able to process all available information as a whole, without having to pass information between separate algorithms.

The DBN-TE is developed by modeling the turbo equalizer on a DBN. The DBN-TE is modeled as a quasi-directed acyclic graph (DAG), assuming that a dominant connection exists between the observed variable and its corresponding hidden variable and weak connections between the observed variable and past and future hidden variables due to ISI. This allows for joint equalization and decoding to be performed in an iterative fashion. During the first iteration only the dominant connection is used to produce initial estimates regarding the transmitted coded information. During subsequent iterations the dominant connection as well as the weak connections to past and future hidden variables are used in conjunction with the coded symbol estimates from the previous iteration to remove the ISI introduced during multipath propagation. The performance of the DBN-TE is comparable to that of the CTE in fading channels, while being able to equalize and decode BPSK modulated information jointly in highly dispersive fading channels. However, the performance of the DBN-TE is worse than that of the CTE in some static channels where the leading channel impulse response (CIR) coefficient is not sufficiently dominant. The computational complexity of the DBN-TE is approximately quadratic at best, and cubic at worst in the coded data block length, exponential in the encoder constraint length and approximately independent of the channel memory length, and is much lower than that of the CTE for systems with long channel memory. Its computational complexity is however inferior compared to MMSE and DFE based turbo equalizers for large coded data block lengths, but comparable to MMSE based turbo equalizers for moderate data block lengths.

The HNN-TE is developed by combining the HNN MLSE equalizer developed by this author in [19, 20] and the HNN decoder in [21, 22]. The equalizer in [19, 20] has computational complexity that is quadratic in the data block length and approximately independent of the channel memory length. It is also able to near-optimally equalize uncoded M-QAM information in extremely dispersive multipath channels. Also, the decoder in [21, 22] is able to decode single balanced codes using the HNN, but in

order to merge the HNN MLSE decoder with the HNN MLSE equalizer, the decoder is modified to enable the decoding of sequences of balanced codes, resulting in an HNN MLSE decoder. The HNN MLSE equalizer and the HNN MLSE decoder are integrated to form the HNN-TE. The HNN-TE performance is comparable to that of the CTE and it is able to equalize and decode BPSK and 4-QAM modulated information jointly in extremely dispersive multipath channels. Like the DBN-TE, the performance of the HNN-TE is lacking in some static channels. The computational complexity of the HNN-TE is approximately independent of the channel memory length and almost quadratically related to the coded data block length, and it is comparable to that of MMSE based turbo equalizers for moderate code data block sizes.

The international journal publications that emanated from this research are as follows:

1. H.C. Myburgh, J.C. Olivier, A.J. Van Zyl, "Reduced Complexity Turbo Equalization Using a Dynamic Bayesian Network," *EURASIP Journal on Advances in Signal Processing*, 2012, 2012:136 (doi: 10.1186/1687-6180-2012-136).

2. H.C. Myburgh, J.C. Olivier, "A Low Complexity Hopfield Neural Network Turbo Equalizer," *EURASIP Journal on Advances in Signal Processing*, 2013, 2013:15 (doi: 10.1186/1687-6180-2013-15).

3. H.C. Myburgh, J.C. Olivier, "A Primer on Equalization, Decoding and Non-Iterative Joint Equalization and Decoding," *EURASIP Journal on Advances in Signal Processing*, 2013, 2013:79 (doi: 10.1186/1687-6180-2013-79).

1.6 OVERVIEW OF STUDY

This study commences in *Chapter 2* with a discussion of the MLSE and MAP algorithms generally used for equalization as well as decoding of convolutional codes. After discussing the MAP and MLSE algorithms in general, equalization and decoding using these algorithms are discussed. This is followed by a discussion on non-iterative joint equalization and decoding. Non-iterative joint equalization and decoding is first discussed for systems that do not employ an interleaver, after which it is discussed for depth-limited block interleavers. Complexity analyses of the non-iterative joint equalizers and decoders are also presented and simulation results are presented throughout this chapter.

Chapter 3 is dedicated to the discussion of turbo equalization. It starts by discussing the operation of the CTE and a computational complexity analysis. This is followed by a discussion on low complexity equalizers used as replacement in a turbo equalizer, after which computational complexity analyses are also performed and compared to that of the CTE. Simulation results are presented throughout this chapter.

In *Chapter 4* the DBN-TE is developed. DBNs are discussed in general, after which the forward-backward algorithm is discussed. The turbo equalization problem is presented as a graphical model, and the significant implications of interleaving are discussed. Next a transformation which mitigates the randomization effect of the interleaver is discussed, after which the DBN-TE algorithm is developed. Complexity reduction and optimization techniques are discussed next, followed by a complexity analysis, where the complexity of the DBN-TE is compared to that of the CTE as well as low complexity CTEs where the optimal MAP equalizer is replaced with suboptimal MMSE and DFE based equalizers. Following the complexity analysis and comparison, simulation results are presented, where the DBN-TE is simulated and compared to the CTE and other low complexity turbo equalizers (LCTE). This chapter is concluded with a discussion and some general remarks.

Chapter 5 presents the HNN-TE. In this chapter the HNN model is discussed first. The HNN MLSE equalizer is discussed next, after which the HNN decoder is discussed and expanded to enable the decoding of a sequence of balanced codes, resulting in an HNN MLSE decoder. The HNN MLSE equalizer and the HNN MLSE decoder are then merged to form the HNN-TE, and this is followed by a computational complexity analysis, where again the complexity of the HNN is compared to that of a CTE as well as other LCTEs. Simulation results are then presented, which show the performance of the HNN-TE compared to a CTE, after which the chapter is concluded with a discussion and some general remarks.

In *Chapter 6* a summary of the research and the resulting outcomes are presented, and suggestions are made for future research. Finally, *Appendix A* serves to explain the simulation environment used to generate the simulation results.

CHAPTER 2

EQUALIZATION, DECODING AND JOINT EQUALIZATION AND DECODING

Equalization and decoding are two essential aspects of any wireless communication system. The equalizer is tasked with reversing the effect of the communication channel on the transmitted information signal, while the decoder receives the equalized symbol sequence and attempts to correct errors that might have been caused during transmission.

The Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm, also known as the MAP algorithm, is useful when designing equalizer and decoder algorithms [2, 3]. The BCJR algorithm receives soft probabilistic information regarding the input and produces *a posteriori* probabilistic information regarding the output, and is aptly called a soft-input soft-output (SISO) algorithm. The availability of reliable soft information at the input allows for more accurate a posteriori estimates to be produced at the output, which improves overall system performance [5–7].

The equalizer can be designed by using the Viterbi algorithm (VA), also known as the MLSE algorithm, which makes use of the min-sum algorithm to find the most probable transmitted sequence [2, 4, 23]. The VA performs optimally but, it is only able to produce hard estimates at the output, and is therefore not an attractive choice when the equalizer is followed by a SISO decoder. The VA can be modified to produce suboptimal posterior probabilistic information at the output, resulting in the soft output Viterbi algorithm (SOVA) in [24], but because of its suboptimal nature the overall system performance will also be suboptimal. Using the MAP algorithm, the equalizer is able to produce optimal posterior probabilistic information regarding the transmitted information, which can be exploited by the SISO decoder.

While the VA and SOVA can also be used for decoding, the MAP algorithm is the algorithm of choice when the output of the decoder is fed back to be used by the equalizer, a technique known as turbo equalization [5, 6]. However, when the estimates of the uncoded transmitted symbols are taken directly from the output of the decoder, ie. when no turbo equalization is performed, the MSLE algorithm will suffice.

Joint equalization and decoding can be performed by MLSE or MAP algorithms and by employing a super-trellis, as shown in [14], given that the depth of the interleaver is limited by computational complexity limitations. The joint equalizer and decoder achieves optimal non-iterative equalization and decoding. Since the input of the joint equalizer and decoder is ISI-corrupted coded transmitted symbols, and the output is the equalized and decoded symbol estimates, no posterior probabilistic information is required at the output. Therefore the MLSE algorithm can also be used. The MAP algorithm will perform just as well, albeit with approximately twice the computational complexity. It has also been shown in [25] that joint decoding of turbo codes can be done on a super-trellis as well.

Fig. 2.1 shows a block diagram of the communication system considered in this chapter. The source information is encoded, after which an interleaver is used to separate adjacent coded bits temporally. The coded bits are mapped to modulation symbols chosen from a modulation alphabet $\mathscr{D}$, after which the symbols are used to modulate the carrier before transmission. The transmitted information passes through a multipath white Gaussian noise channel and is received by the receiver antenna. After reception the signal is demodulated and matched filtered in order to produce a received symbol sequence. The CIR is estimated using a number of known pilot symbols, and is provided as input to the equalizer together with the received symbol sequence. The equalizer reverses the effect of the multipath channel and produces optimal symbol estimates (MAP) or an optimal sequence estimate (MLSE) regarding the transmitted coded bits after interleaving. The output of the equalizer is deinterleaved and provided as input to the decoder, which produces optimal estimates regarding the uncoded transmitted information in the form of log-likelihood ratios (LLR), which are mapped back to bits to produce a final estimate of the source information. When joint equalization and decoding is performed, the equalizer, deinterleaver and decoder are replaced by one functional block, as indicated by the dashed line around these functional blocks. Equalization and decoding are performed simultaneously, producing LLR estimates of the source information at the output.

Since the subject of this thesis is turbo equalization, equalization, decoding and joint equalization

and decoding will be discussed in the context of the MAP algorithm, but for completeness the MLSE algorithm will also be discussed. The MLSE and MAP algorithms are discussed next, after which these algorithms will subsequently be related to equalization and decoding.

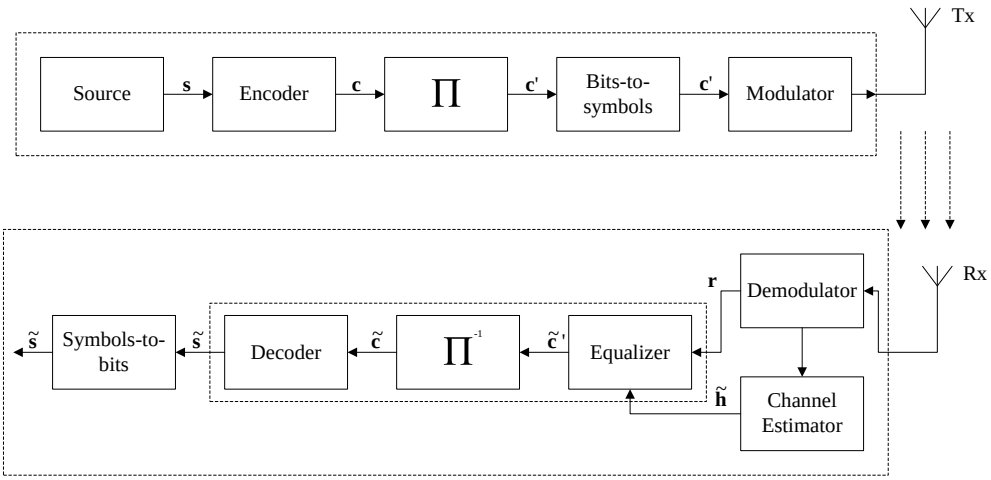


Figure 2.1: Wireless communication system block diagram.

2.1 THE MLSE AND MAP ALGORITHMS

The MLSE and MAP algorithms are used for equalization as well as for the decoding of convolutional codes [1–4,23]. The MLSE algorithm is able to optimally estimate the most probable sequence of transmitted symbols/codewords (depending on equalization/decoding), while the MAP algorithm exactly estimates the probability of each transmitted symbol/codeword. These algorithms are underpinned by Bayes' rule of conditional probability, which states that

$$\begin{aligned} P(d_t = d^{(m)}|r_t) &= \frac{P(d_t=d^{(m)})P(r_t|d_t=d^{(m)})}{P(r_t)} \\ &= \frac{P(d_t=d_t^{(m)})P(r_t|d_t=d^{(m)})}{\sum_{n=1}^{M} P(d_t=d^{(m)})P(r_t|d_t=d^{(m)})} \\ &= \beta P(r_t|d_t = d^{(m)}) \end{aligned} , \tag{2.1}$$

where $P(d_t = d^{(m)})$ is the prior probability of transmitting symbol/codeword $d^{(m)}$ at time instant t. For equalization, the symbols $d^{(m)}$ are chosen from a given modulation alphabet $\mathscr{D}$ of size M, where $m = 1,2,\ldots,M$, and for decoding, codewords symbols $d^{(m)}$ are chosen from a list of M possible codewords. Since it is assumed that the received symbol/codeword sequence is corrupted by white Gaussian noise, the probability of receiving r_t and time instant t having transmitted $d_t^{(m)}$, is expressed

as [1–4, 23]

$$P(r_t|d_t = d^{(m)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\Delta_t^{(m)}}{2\sigma^2}\right), \tag{2.2}$$

where σ is the noise standard deviation and $\Delta_t^{(m)}$ is a cost function for the purpose of minimizing the Euclidean distance between the received symbol/codeword and the symbol/codeword to be estimated, which will be discussed in *Section 2.2* and *Section 2.3* respectively for equalization and decoding. Since $P(d_t = d^{(m)})$ and $P(r_t)$ are independent of the choice of $d^{(m)}$, they can be absorbed into a normalization constant β, which, since $\sum_{m=1}^{M} P(d_t = d^{(m)}|r_t) = 1$, can be expressed as

$$\beta = \frac{1}{\sum_{m=1}^{M} P(d_t = d^{(m)}|r_t)}. \tag{2.3}$$

In order to apply Bayes' rule over a transmitted block of N symbols/codewords, the product rule can be used to express (2.1) for a sequence of length N such that

$$\begin{aligned} P(d_1, d_2, \ldots, d_N|r_1, r_2, \ldots, r_N) &= \beta P(r_1|d_1).P(r_2|d_2).\ldots.P(r_N|d_N) \\ P(\mathbf{d}|\mathbf{r}) &= \beta \prod_{t=1}^{N} P(r_t|d_t) \\ &= \beta \prod_{t=1}^{N} \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\Delta_t^{(m)}}{2\sigma^2}\right)\right) \\ &= \frac{\beta}{(\sqrt{2\pi}\sigma)^N} \exp\left(-\frac{\sum_{t=1}^{N} \Delta_t^{(m)}}{2\sigma^2}\right) \\ &= \frac{\beta}{(\sqrt{2\pi}\sigma)^N} \exp\left(-\frac{\Delta}{2\sigma^2}\right) \end{aligned}, \tag{2.4}$$

where any symbol/codeword d_t in the symbol/codeword sequence $\mathbf{d} = \{d_1, d_2, \ldots, d_N\}$ can be substituted for any $d^{(m)}$, and where $\mathbf{r} = \{r_1, r_2, \ldots, r_N\}$ is the received symbol/codeword sequence, and $\Delta = \sum_{t=1}^{N} \Delta_t^{(m)}$.

2.1.1 The Maximum Likelihood Sequence Estimation Algorithm

In 1972 Forney [1] showed that the VA [1, 4, 23], first developed by Viterbi in 1967 to decode convolutional error correction codes, 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 [1, 2].

The VA, or MLSE algorithm, attempts to minimize the cumulative cost function $\Delta = \sum_{t=1}^{N} \Delta_t^m$ in (2.4), which in turn maximizes $P(\mathbf{d}|\mathbf{r})$ in (2.4). After minimizing Δ in (2.4), there is no sequence of transmitted symbols/codewords that is more likely to have been transmitted. However, the probability

of each estimated symbol/codeword in the sequence will not necessarily be maximized among all possible transmitted symbols/codewords. As stated before, the MAP algorithm is used to calculate exact posterior probabilities on each symbol/codeword. The min-sum algorithm is used to find the MLSE solution and is discussed next.

2.1.1.1 The Min-Sum Algorithm

In order to perform optimal sequence estimation, the min-sum algorithm is used. Viterbi and Forney in [1, 23] showed that the min-sum algorithm can be used by constructing a trellis and tracing a path, corresponding to the most likely transmitted symbol/codeword sequence through the trellis. The min-sum algorithm allows for the elimination of more costly contending paths at each state on a trellis, thereby greatly reducing the computational complexity due to complete enumeration. For equalization, there will always be M^{L-1} possible paths at every stage in the trellis, where M is the modulation alphabet size and L is the channel memory length. Similarly, for decoding, there will always be 2^{K-1} remaining paths at every stage in the trellis (if convolutional encoding is performed in $GF(2)$), where K is the encoder constraint length which introduces memory to the transmitted codewords. M^{L-1} and 2^{K-1} also correspond to the number of respective states in the trellis for each time instant t, for the equalizer and the decoder.

Fig. 2.2 shows a trellis with $M^{L-1} = 2^{K-1} = 4$ states, corresponding to an equalizer used in a system where a BPSK modulated symbol sequence of length N is transmitted through a multipath channel with a CIR of length $L = 3$, or to a decoder used to decode a coded sequence of N codewords where a convolutional encoder with constraint length $K = 3$ is used. The trellis is initiated by assuming that it starts at state S_1 at time instant $t = 0$ and it is terminated at state S_1 at time instant $t = N$. For equalization, each state represents a unique combination of modulation symbols s^m, where $m = 1, 2, \ldots, M$, from a modulation alphabet $\mathscr{D}$ of size M, and for decoding each state represents a possible codeword at the output of the convolutional decoder. The edges, or transitions from state S_i, $i = 1, 2, 3, 4$ at time instant t to any other state S_j, $j = 1, 2, 3, 4$ at time instant $t + 1$ describes the likelihood of this occurrence, which will be a maximum if the cost function $\Delta_t^{(m)}$ in (2.2) is a minimum. Table 2.1 shows the values associated with states S_j, $j = 1, 2, 3, 4$ for equalization and decoding respectively.

For equalization, each state will have M incoming transitions (from the left), while there will be two incoming transitions for decoding.[1] But as stated above, there will only be $2^{L-1} = 2^{K-1} = 4$ remaining

[1] It is assumed that $GF(2)$ decoding is used as usual.

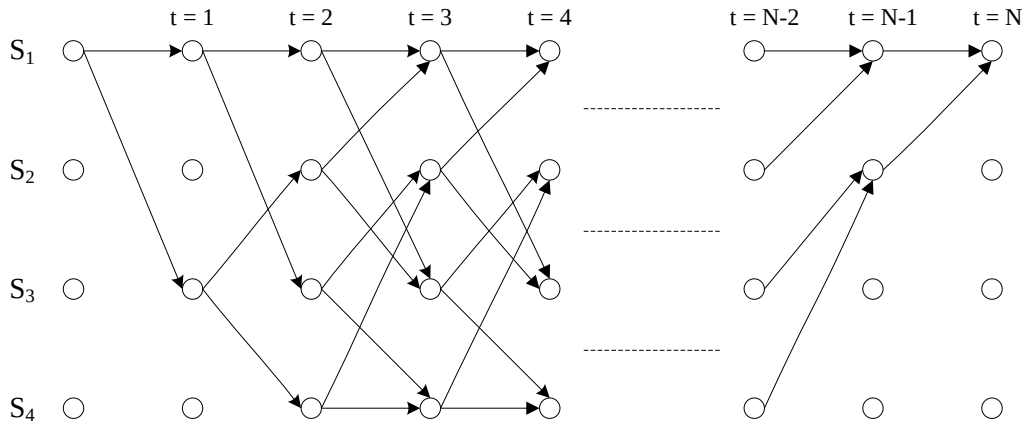


Figure 2.2: Trellis diagram.

Table 2.1: Equalizer/Decoder state values

	S_1	S_2	S_3	S_4
Equalizer	1,1	-1,1	1,-1	-1,-1
Decoder	-1,-1,	1,-1	-1,1	1,1

paths at each time instant in the trellis, which means that all but one path has to be eliminated at each state. To eliminate more costly paths at each state in the trellis at time t, the cumulative cost function $\Delta = \sum_{t=1}^{N} \Delta_t^{(m)}$ in (2.4) is calculated for all possible paths leading to the said state up to time t, where the cost of the contending paths are compared, and the path with the highest cost (corresponding to the lowest probability), is eliminated. Therefore, at each state all the previous $\Delta_{t,j\to i}$'s are accumulated, and where there are contending paths, the path with the largest accumulated cost is eliminated.

There will therefore be $2^2 = 4$ surviving paths in each stage on the trellis for stages beyond $t = 2$. When stages $t = N-2$ to $t = N$ are considered, the number of allowed transitions decrease, because of the known tail symbols at the end of the transmitted symbol sequence. For the last few states in the trellis the contending paths are also eliminated, until only one possible path remains at stage $t = N$. This path is then traced back to determine the most probable sequence of transmitted symbols/codewords.

The MLSE equalizer/decoder produces outputs from a set of M symbols for equalization or from a set of uncoded bits for decoding. 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 is discussed next.

2.1.2 The Maximum A Posteriori Probability Algorithm

Following the development of the Viterbi MLSE decoding algorithm [23], 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 (BP) [15]. The MAP algorithm is able 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 [1, 2, 4, 23].

The aim of the MAP algorithm is to maximize the posterior probability distribution for each transmitted symbol/codeword. While the MLSE algorithm assumes the prior probabilities $P(\mathbf{d})$ of the transmitted symbols to be equal, the MAP algorithm is able to exploit the prior probabilities $P(\mathbf{d})$, if necessary, in order to enhance the quality of posterior probabilistic information on each individual transmitted symbol. Like the MLSE algorithm, the MAP algorithm uses the model in (2.4) on a trellis, but unlike the MLSE algorithm, the MAP algorithm propagates the transition probabilities forward from past states to future states, as well as backwards from future states to past states, after which the marginalized probability for each estimated symbol/codeword d_t is produced, given past and future information. That is [3, 26]

$$P(d_t = d^{(m)}|\mathbf{r}) = \sum_{d_{t^\dagger}:t^\dagger \neq t}^{N} P(\mathbf{d}|\mathbf{r}) \tag{2.5}$$

where again $d^{(m)}$, $m = 1, 2, \ldots, M$, is the mth symbol chosen from a modulation alphabet $\mathscr{D}$ of size M for equalization, or $d^{(m)}$ is the mth codeword chosen from a list of M codewords produced by a convolutional encoder for decoding, $\mathbf{r}$ is the received sequence, and N is the number of symbols/codewords in the received sequence.

Referring to Fig. 2.2, the probability of a transition from state $S_{j,t-1}$ (at time $t-1$) to state $S_{i,t}$ (at time t), where $i, j = 1, 2, \ldots, M^{L-1}$ for equalization and $i, j = 1, 2, \ldots, 2^{K-1}$ for decoding, is given by

$$\omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) = \beta P(d_t = d^{(m)}) P(r_t | S_{j,t-1}, S_{i,t}), \tag{2.6}$$

where β is a normalization constant and $P(r_t|S_{j,t-1},S_{i,t})$ is given by

$$P(r_t|S_{j,t-1},S_{i,t}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-\Delta}{2\sigma^2}\right). \tag{2.7}$$

By considering the relevant transition value[2] d_ξ, $P(d_t)$ can be written as [5,6]

$$P(d_t = d^{(m)}) = \exp\left(\frac{1}{2} d_\xi L(\tilde{d}_t)\right), \tag{2.8}$$

where $L(.)$ denotes the LLR operation and $\tilde{d}_t$ is an estimate of d_t. Therefore, substituting (2.7) and (2.8) in (2.6), yields

$$\omega_{j\to i,t}(S_{j,t-1},S_{i,t}) = \frac{\delta}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-\Delta}{2\sigma^2} + \exp\left(\frac{1}{2} d_\xi L(\tilde{d}_t)\right)\right), \tag{2.9}$$

which completely describes the probability of a transition from $S_{j,t-1}$ (or d_j) at time $t-1$ to $S_{i,t}$ (or d_i) at time t, based on all available information. Thus, for a transition labeled $d_\xi = 1$ and $L(\tilde{d}_t)$ equal to any large positive value,[3] $P(d_t = d^{(m)})$ will be large, confirming the transition. Similarly, for a transition labeled $d_\xi = -1$ and $L(\tilde{d}_t)$ equal to any large negative value $P(d_t = d^{(m)})$ will be large, also confirming the transition. However, for a transition labeled $d_\xi = 1$ and $L(d_\xi)$ equal to any large negative number, or for a transition labeled $d_\xi = -1$ and $L(\tilde{d}_t)$ equal to any large positive number, $P(d_t = d^{(m)})$ will be small. Thus, if the prior information $L(\tilde{d}_t)$ is in agreement with the transition value d_ξ, the probability of that transition will increase, confirming that transition. Otherwise, if the prior information $L(\tilde{d}_k)$ contradicts the transition value d_ξ, the probability of that transition will be decreased.

Propagating the transition probabilities $\omega_{j\to i,t}(S_{j,t-1},S_{i,t})$ across the whole sequence from left to right and from right to left respectively and marginalizing according to (2.5), will produce the posterior probabilities of each estimated d_k. The sum-product algorithm achieves exactly this [26].

2.1.2.1 The Sum-product Algorithm

The sum-product algorithm, also known as the forward-backward algorithm, uses the trellis in Fig. 2.2 to perform marginalization. This algorithm follows three steps [26]:

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.

[2] $d_\xi \in$ {-1,1} for BPSK.

[3] The magnitude of $L(\tilde{d}_t)$ 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 t run from left to right (from 1 to N) on the trellis and compute for each state in the trellis

$$\alpha_{i,t} = \sum_{j \varepsilon Parent(i)} \omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) \alpha_{j,t-1}$$

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

$$\beta_{j,t} = \sum_{j \varepsilon Parent(i)} \omega_{j \to i,t}(S_{j,t}, S_{i,t+1}) \beta_{i,t+1}$$

where again j represents the parent states of the current state at stage i of the trellis and $\omega_t(S_{j,t}, S_{i,t+1})$ is the probability associated with the transition from $S_{j,t}$ to $S_{i,t+1}$. Note that $\beta_{i,t} = 1$. Finally, the exact marginalized symbol probability is determined by summing over all states at each time instant t corresponding to a transition of either $d_\xi = 1$ or $d_\xi = -1$ such that

$$P(d_t = 1|\mathbf{r}) = \sum_{j \varepsilon Parent(i), d_\xi = 1} \alpha_{j,t-1} \omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) \beta_{i,t} \tag{2.10}$$

$$P(d_t = -1|\mathbf{r}) = \sum_{j \varepsilon Parent(i), d_\xi = -1} \alpha_{j,t-1} \omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) \beta_{i,t}. \tag{2.11}$$

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

$$L(\tilde{s}_t) = \log \left(\frac{P(d_t = 1|\mathbf{r})}{P(d_t = -1|\mathbf{r})} \right), \tag{2.12}$$

where the sign of $L(\tilde{d}_t)$ indicates whether $\tilde{d}_t = -1$ or $\tilde{d}_t = 1$, and $|L(\tilde{s}_t)|$ is a measure of the confidence of that estimate.

2.2 EQUALIZATION

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 [2]. 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 receiver, to aid in the estimation of the transmitted information.

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 tT_s. As such, a tapped delay line is used to model the behavior of this channel, as shown in Fig. 2.3. Fig. 2.3 indicates that the tth transmitted symbol s_t is delayed by T_s seconds $L-1$ times,

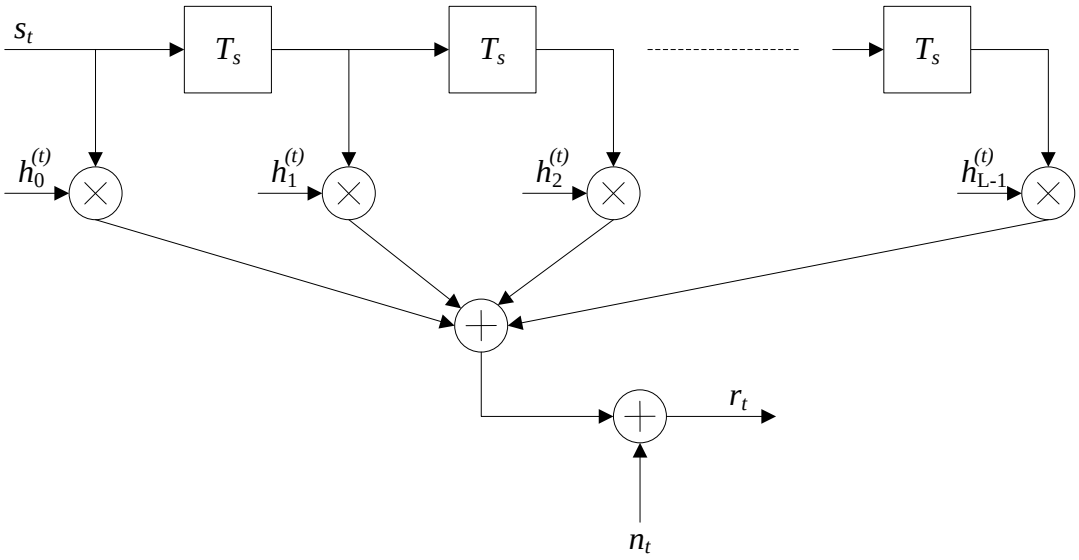


Figure 2.3: Tapped delay line for a multipath fading channel.

where L is the CIR length and T_s is the symbol period. Each delayed copy of s_t is multiplied by $h_l^{(t)}$, $l = 1,2,...,L-1$, corresponding to the lth delay branch at time t. Therefore, the tth received symbol can be described by [1, 2]

$$r_t = \sum_{l=0}^{L-1} h_l^{(t)} s_{t-l} + n_t, \tag{2.13}$$

where n_t is the tth noise sample from the distribution $\mathcal{N}(\mu = 0, \sigma^2 = 1)$. Each $h_l^{(t)}$ is a sample taken from one of L time-varying functions, where t corresponds with the tth transmitted symbol and $l = 1,2,...,L-1$ is the CIR tap number. Each CIR tap is modeled as an independent uncorrelated Rayleigh fading sequence, describing the fading behavior of that tap (see *Appendix A*).

If it is assumed that the CIR is time-invariant for the duration of a data block, (2.13) can be rewritten as

$$r_t = \sum_{l=0}^{L-1} h_l s_{t-l} + n_t, \tag{2.14}$$

where s_t denotes the tth complex symbol in the transmitted sequence of N symbols chosen from an alphabet $\mathscr{D}$ containing M complex symbols, r_t is the tth received symbol, n_t is the tth noise sample from the distribution $\mathscr{N}(\mu = 0, \sigma^2 = 1)$, and h_l is the lth coefficient of the estimated CIR. Equalization is performed under the assumption that each CIR coefficient h_l is time-invariant for the duration of a data block. The CIR $\mathbf{h} = \{h_0, h_1, \ldots, h_{L-1}\}$ therefore completely describes the multipath channel for a given received data block, assuming that the data block is sufficiently short so as to render the CIR time-invariant. The equalizer takes as input the received symbol sequence $\mathbf{r}$ as well as the CIR $\mathbf{h}$.

To estimate the transmitted sequence of length N optimally in a wireless communication system transmitting modulated symbols though a multipath channel, the cumulative cost function in (2.4)

$$\begin{aligned} \Delta &= \sum_{t=1}^{N} \Delta_t^{(m)} \\ &= \sum_{t=1}^{N} |r_t - \sum_{l=0}^{L-1} h_l s_{t-l}|^2 \end{aligned} \tag{2.15}$$

must be minimized [1, 2]. Here $\mathbf{s} = \{s_1, s_2, \cdots, s_N\}$ is the most likely transmitted sequence that will maximize (2.4). Although the minimization of (2.15) will maximize (2.4), the resulting sequence estimates will only be optimal in the sequence sense. Depending on the application and whether the equalizer is followed by a decoder, either optimal sequence estimation or exact posterior probabilistic symbol estimation can be performed using the MLSE or MAP algorithms discussed above.

2.3 DECODING

In a mobile communication system the transmitted signal is subjected 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 controlled redundancy to the source information. The decoder exploits the structure introduced to the encoded symbol sequence by the encoder, to reconstruct the source information from the estimated coded symbols, correcting errors while doing so.

ECC, also referred to as error control coding, plays an important role in digital communication systems. As the name suggests, ECC is used to allow for the correction of errors in the received symbol sequence. EEC is performed by adding controlled redundancy to the information being transmitted. Since the redundant information is mathematically related to the original information, errors can be corrected [2]. Although the redundancy adds an overhead to the transmitted data, the performance increase overshadows this drawback. During the last few decades, ECC has been the subject of much

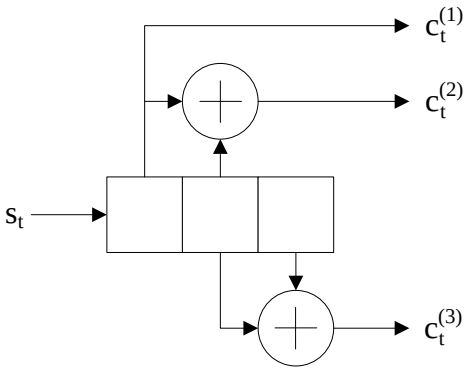


Figure 2.4: Rate $1/3$ convolutional encoder.

research and significant contributions and advancements have been made.

One of the coding schemes considered in this thesis is convolutional encoding, as it is most often used in turbo equalization. A convolutional code is generated by passing the information sequence to be transmitted through a linear finite state shift register, consisting of K stages.[4] The binary input is shifted into the shift register k bits at a time, producing n output bits. The code rate is therefore $R_c = k/n$. The encoder can be expressed in terms of n generator polynomials, describing the connections between the states of the encoder shift register. Fig. 2.4 shows the rate $R_c = k/n = 1/3$, constraint length $K = 3$, convolutional encoder considered in this thesis.

The generator polynomials that describe the connections between the elements in the shift register are given in sequence form as $g_1 = [100]$, $g_2 = [110]$ and $g_3 = [011]$, which can also be written in octal form as $G = [4,6,3]$ or in polynomial form as $G = [1, 1+X, X+X^2]$. For every input source bit s_t the encoder produces $k = 3$ output bits $c_t^{(1)}$, $c_t^{(2)}$ and $c_t^{(3)}$, where

$$\begin{aligned} c_t^{(1)} &= s_t \\ c_t^{(2)} &= s_t \oplus s_{(t-1)} \\ c_t^{(3)} &= s_{(t-1)} \oplus s_{(t-2)}, \end{aligned} \tag{2.16}$$

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 $K-1$ zero bits to the data block. This is done to enable decoding using an

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

MLSE or a MAP decoder [3, 4, 23].

Each convolution encoder has a corresponding state diagram, which is used to map state transitions to encoder outputs, which in turn are used to determine the most likely state transitions during decoding. The state diagram of the encoder in Fig. 2.4 is shown in Fig. 2.5. Each state contains two bits representing the two leftmost elements in the encoder shift register. As bits are shifted through the encoder $k = 1$ bit at a time, transitions occur, with each state transition producing $n = 3$ output bits. The dashed lines are associated with transitions resulting from a zero at the input of the encoder, and solid lines are association with a one at the input of the encoder.

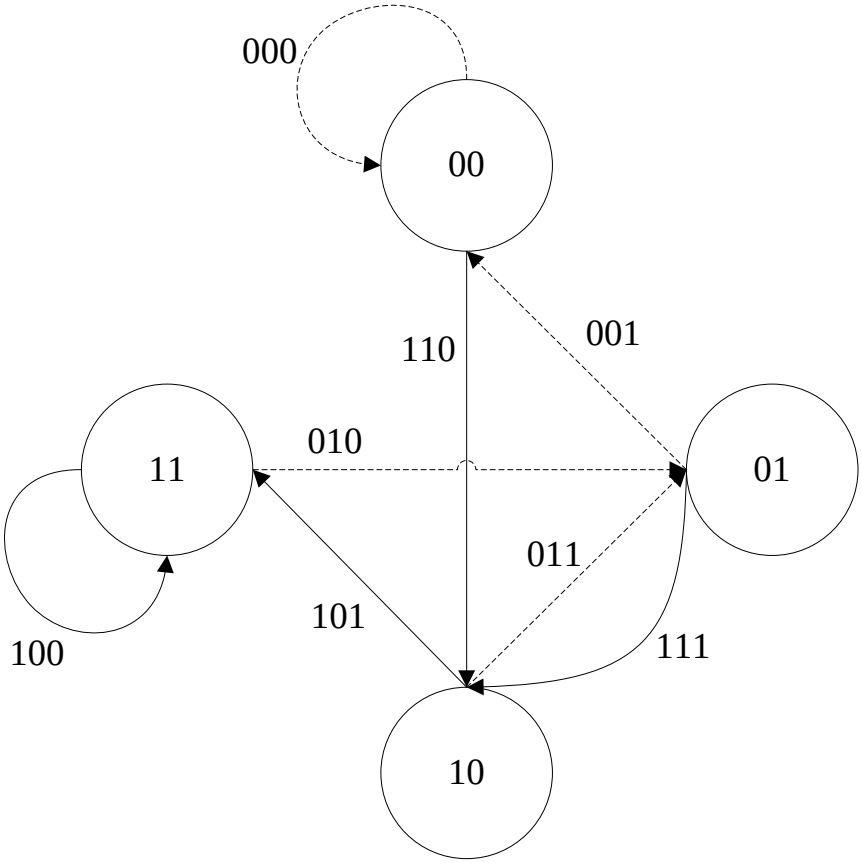


Figure 2.5: Rate $1/3$ convolutional encoder state diagram.

The decoding of convolutional codes is closely related to equalization discussed in *Section 2.2*, in that the min-sum (Viterbi MLSE) and sum-product (MAP/BCJR) algorithms can also be used for decoding. The MAP algorithm is an attractive choice for use in iterative equalization/decoding algorithms. It is attractive for two reasons: firstly because it includes prior probabilistic information in the estimation, and secondly, because it provides soft posterior estimates regarding individual coded

or uncoded symbols, which in turn can be used as prior information in subsequent iterations.

To decode a convolutional code using the MLSE or MAP algorithms discussed above, the cumulative cost function in (2.4)

$$\begin{aligned} \Delta &= \sum_{t=1}^{N} \Delta_t^{(m)} \\ &= \sum_{t=1}^{N} \sum_{n=1}^{k} |r_t^{(n)} - c_t^{(n)}|^2 \end{aligned} \tag{2.17}$$

is minimized [1, 23], where $\mathbf{r}_t = \{r_t^{(1)}, \ldots, r_t^{(n)}\}$ are the received coded symbols corresponding to codeword $\mathbf{c}_t = \{c_t^{(1)}, \ldots, c_t^{(k)}\}$ at time instant t, selected based on the encoder output generated by the relevant state transition.[5] Here k is the number of output bits generated by the encoder. The MLSE or MAP algorithms can be applied to find either the most probable transmitted sequence of codewords, or the most probable transmitted uncoded and/or coded symbols. The output of the latter can be used in a turbo equalizer as feedback to aid the equalizer in making more informed decisions.

2.4 NON-ITERATIVE JOINT EQUALIZATION AND DECODING (NI-JED)

In conventional single-carrier wireless communication systems, where the coded information is transmitted through a multipath channel, equalization and decoding, as explained in *Section 2.2* and *Section 2.3*, are performed separately. However, joint equalization and decoding can also be performed on a single trellis when it is assumed that that the coded symbols are not interleaved before transmission, or that certain assumptions are made regarding the structure of the interleavers used to interleave the coded symbols.

Interleavers allow for the temporal separation of adjacent coded symbols during transmission, so that when burst errors occur in response to a loss of signal power during fading, the resulting errors in an error burst will be distributed throughout the data block after deinterleaving. This leads to performance gains in the receiver, as the equalizer and the decoder are able to infer information and correct single errors more accurately, as opposed to correcting consecutive, or burst errors. Therefore interleavers are employed in a wireless communication system where the transmitted signal is subject to multipath and fading [2].

In [14] an optimal joint equalizer and decoder is presented, where the equalizer and the decoder

[5] During decoding, the output bits $\mathbf{c}_t = \{c_t^{(1)}, \ldots, c_t^{(n)}\}$ are made bipolar because the elements of $\mathbf{c}_t$ are compared to received *symbols*, and not bits.

are jointly modeled on one trellis, a super-trellis according to [14], and a block interleaver is used to interleave the coded symbols before transmission. The computational complexity of this joint equalizer and decoder, however, is not only exponentially related to the channel memory length and the encoder constraint length, but also to the interleaver depth. The interleaver depth determines the degree of separation between the coded symbols, which has a direct effect on system performance. In order to improve system performance, the interleaver depth has to be increased, which results in an exponential increase in computational complexity. Ideally a random interleaver should be used for maximum performance gains, but a random interleaver has a devastating effect on the causality relationship between transmitted and deinterleaved received symbols. This joint equalizer/decoder is therefore only feasible when limited depth block interleavers are used. Joint equalization and decoding using a super-trellis is discussed next, first for systems where no interleaving is performed, and then for systems where depth-limited interleavers are used.

2.4.1 No Interleaving

Noting the striking similarities between MAP equalization and MAP decoding in *Section 2.1*, a possible joint equalization and decoding algorithm can be envisioned. In order to equalize and decode a received symbol sequence jointly, the cumulative costs for equalization in (2.15) and decoding in (2.17) must be combined in order to equalize while decoding. Either the MLSE or MAP algorithms can be used for this purpose, since the super-trellis encapsulates all the available information and therefore no exchange of information between equalizer and decoder is necessary. No prior information is therefore required when calculating state transitions on a super-trellis.

Consider a system transmitting noninterleaved coded information through a multipath channel of length L. A rate $R_c = k/n$ constraint length K convolutional encoder is used to produce a sequence $\mathbf{c}$ of coded bits of length N_c from an uncoded bit sequence $\mathbf{s}$ of length N_u. The number of super-trellis states required to perform joint equalization and decoding is the product of the number of states of the individual trellises required to perform equalization and decoding separately. Therefore the number of super-trellis states is $M^{L-1}M^{K-1} = M^{(L-1)+(K-1)}$, where $M = 2$ due to BPSK modulation as well as $GF(2)$ encoding.

When the MAP equalizer was considered, the probability of a transition from uncoded *symbol* s_j transmitted and time $t-1$ to s_i transmitted at time t was calculated (see (2.6)), and when the MAP decoder was considered, the probability of a transition from *codeword* $\mathbf{c}_j$ transmitted at time $t-1$ to

$\mathbf{c}_i$ transmitted at time t was calculated (although d_j and d_i were used instead for both the equalization of symbols and decoding of codewords). When modeling the equalizer and decoder jointly, however, transitions between super states are calculated on a super-trellis, where each state represents the uncoded symbols that constitute a codeword, together with interfering symbols of previous codewords due to multipath.

Suppose a communication system encodes a source bit sequence $\mathbf{s}$ of length N_u with a rate $R_c = k/n = 1/3$, constraint length $K = 3$, convolutional encoder with arbitrary generator polynomials $g_1 = \{g_1^{(1)}, g_1^{(2)}, g_1^{(3)}\}$, $g_2 = \{g_2^{(1)}, g_2^{(2)}, g_2^{(3)}\}$ and $g_3 = \{g_3^{(1)}, g_3^{(2)}, g_3^{(3)}\}$, in order to produce a coded bit sequence $\mathbf{c}$ of length $N_c = N_u/R_c$, which is BPSK modulated and transmitted through a multipath channel with a CIR $\mathbf{h}$ of length L. The encoder produces the coded bit sequence

$$\mathbf{c} = \{c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, \ldots, c_{N_u-1}^{(1)}, c_{N_u-1}^{(2)}, c_{N_u-1}^{(3)}, c_{N_u}^{(1)}, c_{N_u}^{(2)}, c_{N_u}^{(3)}\}$$

of length N_c from the length N_c uncoded bit sequence

$$\mathbf{s} = \{s_1, s_2 \ldots, s_{N_u-1}, s_{N_u}\}.$$

2.4.1.1 Channel length $2 \leq L < 5$

When the coded sequence is transmitted through a channel with lengths $L = 2$, $L = 3$ and $L = 4$, the respective resulting received sequences, without noise, can be expressed in terms of the coded symbols

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)} h_0 + c_{t-1}^{(3)} h_1 \\ r_t^{(2)} &= c_t^{(2)} h_0 + c_t^{(1)} h_1 \\ r_t^{(3)} &= c_t^{(3)} h_0 + c_t^{(2)} h_1 \end{aligned} , \tag{2.18}$$

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)} h_0 + c_{t-1}^{(3)} h_1 + c_{t-1}^{(2)} h_2 \\ r_t^{(2)} &= c_t^{(2)} h_0 + c_t^{(1)} h_1 + c_{t-1}^{(3)} h_2 \\ r_t^{(3)} &= c_t^{(3)} h_0 + c_t^{(2)} h_1 + c_t^{(1)} h_3 \end{aligned} , \tag{2.19}$$

and

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)} h_0 + c_{t-1}^{(3)} h_1 + c_{t-1}^{(2)} h_2 + c_{t-1}^{(1)} h_3 \\ r_t^{(2)} &= c_t^{(2)} h_0 + c_t^{(1)} h_1 + c_{t-1}^{(3)} h_2 + c_{t-1}^{(2)} h_3 \\ r_t^{(3)} &= c_t^{(3)} h_0 + c_t^{(2)} h_1 + c_t^{(1)} h_3 + c_{t-1}^{(3)} h_3 \end{aligned} , \tag{2.20}$$

which can respectively be expressed in terms of the uncoded symbols

$$\begin{aligned} r_t^{(1)} &= (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_0 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_1 \\ r_t^{(2)} &= (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_0 + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_1 \\ r_t^{(3)} &= (g_3^{(1)}s_t \oplus g_3^{(2)}s_{t-1} \oplus g_3^{(3)}s_{t-2})h_0 + (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_1 \end{aligned}, \tag{2.21}$$

for $L = 2$,

$$\begin{aligned} r_t^{(1)} &= (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_0 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_1 \\ &\quad + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_2 \\ r_t^{(2)} &= (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_0 + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_1 \\ &\quad + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_2 \\ r_t^{(3)} &= (g_3^{(1)}s_t \oplus g_3^{(2)}s_{t-1} \oplus g_3^{(3)}s_{t-2})h_0 + (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_1 \\ &\quad + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_2 \end{aligned}, \tag{2.22}$$

for $L = 3$, and

$$\begin{aligned} r_t^{(1)} &= (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_0 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_1 \\ &\quad + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_2 + (g_1^{(1)}s_{t-1} \oplus g_1^{(2)}s_{t-2} \oplus g_1^{(3)}s_{t-3})h_3 \\ r_t^{(2)} &= (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_0 + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_1 \\ &\quad + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_2 + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_3 \\ r_t^{(3)} &= (g_3^{(1)}s_t \oplus g_3^{(2)}s_{t-1} \oplus g_3^{(3)}s_{t-2})h_0 + (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_1 \\ &\quad + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_2 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_3 \end{aligned}, \tag{2.23}$$

for $L = 4$. It is clear from (2.21) to (2.23) that each $\mathbf{r}_t = \{r_t^{(1)}, r_t^{(2)}, r_t^{(3)}\}$ contains uncoded symbols $\{s_t, s_{t-1}, s_{t-2}, s_{t-3}\}$. The transition probability between superstate $S_{j,t-1}$ and $S_{i,t}$ can therefore be calculated, where $S_{i,t}$ represents $\{s_t, s_{t-1}, s_{t-2}\}$ and $S_{j,t-1}$ represents $\{s_{t-1}, s_{t-2}, s_{t-3}\}$. The number of super-trellis states will therefore be $M = 2^3$, as three symbols need to be represented by each superstate. In order to express the number of states for $2 \leq L < 5$ this author defines

$$Q = 1, \quad if \quad 2 \leq L \leq 4. \tag{2.24}$$

The number of super-trellis states can therefore be expressed as $M = 2^{Q+(K-1)} = 2^{(1)+(2)} = 2^3$.

2.4.1.2 Channel length $5 \leq L < 8$

Transmitting the coded sequence

$$\mathbf{c} = \{c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, \ldots, c_{N_u-1}^{(1)}, c_{N_u-1}^{(2)}, c_{N_u-1}^{(3)}, c_{N_u}^{(1)}, c_{N_u}^{(2)}, c_{N_u}^{(3)}\}$$

through a channel length $L = 5$, $L = 6$ and $L = 7$ respectively, the received symbol sequences without noise can be expressed as

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)}h_0 + c_{t-1}^{(3)}h_1 + c_{t-1}^{(2)}h_2 + c_{t-1}^{(1)}h_3 + c_{t-2}^{(3)}h_4 \\ r_t^{(2)} &= c_t^{(2)}h_0 + c_t^{(1)}h_1 + c_{t-1}^{(3)}h_2 + c_{t-1}^{(2)}h_3 + c_{t-1}^{(1)}h_4 \\ r_t^{(3)} &= c_t^{(3)}h_0 + c_t^{(2)}h_1 + c_t^{(1)}h_3 + c_{t-1}^{(3)}h_3 + c_{t-1}^{(2)}h_4 \end{aligned} , \tag{2.25}$$

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)}h_0 + c_{t-1}^{(3)}h_1 + c_{t-1}^{(2)}h_2 + c_{t-1}^{(1)}h_3 + c_{t-2}^{(3)}h_4 + c_{t-2}^{(2)}h_5 \\ r_t^{(2)} &= c_t^{(2)}h_0 + c_t^{(1)}h_1 + c_{t-1}^{(3)}h_2 + c_{t-1}^{(2)}h_3 + c_{t-1}^{(1)}h_4 + c_{t-2}^{(3)}h_5 \\ r_t^{(3)} &= c_t^{(3)}h_0 + c_t^{(2)}h_1 + c_t^{(1)}h_3 + c_{t-1}^{(3)}h_3 + c_{t-1}^{(2)}h_4 + c_{t-1}^{(1)}h_5 \end{aligned} , \tag{2.26}$$

and

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)}h_0 + c_{t-1}^{(3)}h_1 + c_{t-1}^{(2)}h_2 + c_{t-1}^{(1)}h_3 + c_{t-2}^{(3)}h_4 + c_{t-2}^{(2)}h_5 + c_{t-2}^{(1)}h_6 \\ r_t^{(2)} &= c_t^{(2)}h_0 + c_t^{(1)}h_1 + c_{t-1}^{(3)}h_2 + c_{t-1}^{(2)}h_3 + c_{t-1}^{(1)}h_4 + c_{t-2}^{(3)}h_5 + c_{t-2}^{(2)}h_6 \\ r_t^{(3)} &= c_t^{(3)}h_0 + c_t^{(2)}h_1 + c_t^{(1)}h_3 + c_{t-1}^{(3)}h_3 + c_{t-1}^{(2)}h_4 + c_{t-1}^{(1)}h_5 + c_{t-2}^{(3)}h_6 \end{aligned} . \tag{2.27}$$

As before, expressing $\mathbf{r}_t = \{r_t^{(1)}, r_t^{(2)}, r_t^{(3)}\}$ above in terms of the uncoded symbols, one gets

$$\begin{aligned} r_t^{(1)} &= (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_0 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_1 \\ &\quad + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_2 + (g_1^{(1)}s_{t-1} \oplus g_1^{(2)}s_{t-2} \oplus g_1^{(3)}s_{t-3})h_3 \\ &\quad + (g_3^{(1)}s_{t-2} \oplus g_3^{(2)}s_{t-3} \oplus g_3^{(3)}s_{t-4})h_4 \end{aligned} , \tag{2.28}$$

$$\begin{aligned} r_t^{(2)} &= (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_0 + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_1 \\ &\quad + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_2 + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_3 \\ &\quad + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_4 \end{aligned} , \tag{2.29}$$

$$\begin{aligned} r_t^{(3)} &= (g_3^{(1)}s_t \oplus g_3^{(2)}s_{t-1} \oplus g_3^{(3)}s_{t-2})h_0 + (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_1 \\ &\quad + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_2 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_3 \\ &\quad + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_4 \end{aligned} , \tag{2.30}$$

for $L = 5$,

$$\begin{aligned} r_t^{(1)} &= (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_0 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_1 \\ &\quad + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_2 + (g_1^{(1)}s_{t-1} \oplus g_1^{(2)}s_{t-2} \oplus g_1^{(3)}s_{t-3})h_3 \\ &\quad + (g_3^{(1)}s_{t-2} \oplus g_3^{(2)}s_{t-3} \oplus g_3^{(3)}s_{t-4})h_4 + (g_2^{(1)}s_{t-2} \oplus g_2^{(2)}s_{t-3} \oplus g_2^{(3)}s_{t-4})h_5 \end{aligned} , \tag{2.31}$$

$$\begin{aligned} r_t^{(2)} &= (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_0 + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_1 \\ &\quad + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_2 + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_3 \\ &\quad + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_4 + (g_3^{(1)}s_{t-2} \oplus g_3^{(2)}s_{t-3} \oplus g_3^{(3)}s_{t-4})h_5 \end{aligned} , \tag{2.32}$$

$$\begin{aligned} r_t^{(3)} &= (g_3^{(1)}s_t \oplus g_3^{(2)}s_{t-1} \oplus g_3^{(3)}s_{t-2})h_0 + (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_1 \\ &\quad +(g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_2 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_3 \\ &\quad +(g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_4 + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_5 \end{aligned}, \tag{2.33}$$

for $L = 6$,

$$\begin{aligned} r_t^{(1)} &= (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_0 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_1 \\ &\quad +(g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_2 + (g_1^{(1)}s_{t-1} \oplus g_1^{(2)}s_{t-2} \oplus g_1^{(3)}s_{t-3})h_3 \\ &\quad +(g_3^{(1)}s_{t-2} \oplus g_3^{(2)}s_{t-3} \oplus g_3^{(3)}s_{t-4})h_4 + (g_2^{(1)}s_{t-2} \oplus g_2^{(2)}s_{t-3} \oplus g_2^{(3)}s_{t-4})h_5 \\ &\quad +(g_1^{(1)}s_{t-2} \oplus g_1^{(2)}s_{t-3} \oplus g_1^{(3)}s_{t-4})h_6 \end{aligned}, \tag{2.34}$$

$$\begin{aligned} r_t^{(2)} &= (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_0 + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_1 \\ &\quad +(g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_2 + (g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_3 \\ &\quad +(g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_4 + (g_3^{(1)}s_{t-2} \oplus g_3^{(2)}s_{t-3} \oplus g_3^{(3)}s_{t-4})h_5 \\ &\quad +(g_2^{(1)}s_{t-2} \oplus g_2^{(2)}s_{t-3} \oplus g_2^{(3)}s_{t-4})h_6 \end{aligned}, \tag{2.35}$$

$$\begin{aligned} r_t^{(3)} &= (g_3^{(1)}s_t \oplus g_3^{(2)}s_{t-1} \oplus g_3^{(3)}s_{t-2})h_0 + (g_2^{(1)}s_t \oplus g_2^{(2)}s_{t-1} \oplus g_2^{(3)}s_{t-2})h_1 \\ &\quad +(g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_2 + (g_3^{(1)}s_{t-1} \oplus g_3^{(2)}s_{t-2} \oplus g_3^{(3)}s_{t-3})h_3 \\ &\quad +(g_2^{(1)}s_{t-1} \oplus g_2^{(2)}s_{t-2} \oplus g_2^{(3)}s_{t-3})h_4 + (g_1^{(1)}s_t \oplus g_1^{(2)}s_{t-1} \oplus g_1^{(3)}s_{t-2})h_5 \\ &\quad +(g_3^{(1)}s_{t-2} \oplus g_3^{(2)}s_{t-3} \oplus g_3^{(3)}s_{t-4})h_6 \end{aligned}, \tag{2.36}$$

for $L = 7$. It is clear from (2.30) to (2.36) that for $5 \leq L < 8$ $\mathbf{r}_t = \{r_t^{(1)}, r_t^{(2)}, r_t^{(3)}\}$ contains $\{s_t, s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4}\}$, necessitating super-trellis state $S_{j,t-1}$ and $S_{i,t}$ to represent respectively $\{s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4}\}$ and $\{s_t, s_{t-1}, s_{t-2}, s_{t-3}\}$ in order to calculate the transition probability. With the new information, this author redefines Q in (2.24) as

$$Q = \begin{cases} 1 & if \quad 2 \leq L < 5 \\ 2 & if \quad 5 \leq L < 8 \end{cases}.$$

The number of super-trellis states required for $5 \leq L < 8$ is therefore $M = 2^{Q+(K-1)} = 2^{(2)+(2)} = 2^4$.

2.4.1.3 Channel length L

Given any channel memory length L, encoder constraint length K, code rate $R_c = k/n = 1/3$, the number of bits needed to be represented by a superstate can be determined by $Q + (K-1)$, where

$$Q = \begin{cases} 1 & if \quad n-1 \leq L < 2(n-1) \\ 2 & if \quad 2(n-1) \leq L < 3(n-1) \\ 3 & if \quad 3(n-1) \leq L < 4(n-1) \\ \vdots & \vdots \quad \vdots \\ q-1 & if \quad (q-1)(n-1) \leq L < (q-2)(n-1) \\ q & if \quad q(n-1) \leq L < (q-1)(n-1) \end{cases},$$

which in turn can be used to determine the number of super-trellis states by $M = 2^{Q+(K+1)}$. Keeping track of the ISI undergone by the coded symbols due to multipath, and replacing those symbols with their corresponding uncoded symbols, a model for the received symbols can be derived.

The probability of a transition from superstate $S_{j,t-1}$ to superstate $S_{i,t}$ is given by

$$\omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | S_{j,t-1}, S_{i,t}) P(s_t), \tag{2.37}$$

which can be rewritten in terms of the uncoded symbols represented by each state

$$\omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | s_t, s_{t-1}, \ldots, s_{t-\log_2(M)}) P(s_t), \tag{2.38}$$

where

$$P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | s_t, s_{t-1}, \ldots, s_{t-\log_2(M)}) = \frac{1}{\sqrt{2\pi}\sigma} \left(\frac{-\sum_{i=1}^{k} \Delta_{i,t}}{2\sigma^2} \right). \tag{2.39}$$

$\Delta_{i,t}$ can be determined by minimizing the receiver equations such that

$$\begin{aligned} \Delta_{1,t} &= \left| r_t^{(1)} - \textstyle\sum_{l=0}^{L-1} h_l (g_u^{(1)} s_{t-m} \oplus g_u^{(2)} s_{(t-m)-1} \oplus g_u^{(3)} s_{(t-m)-2}) \right|^2 \\ \Delta_{2,t} &= \left| r_t^{(2)} - \textstyle\sum_{l=0}^{L-1} h_l (g_v^{(1)} s_{t-n} \oplus g_v^{(2)} s_{(t-n)-1} \oplus g_v^{(3)} s_{(t-n)-2}) \right|^2 \\ \Delta_{3,t} &= \left| r_t^{(3)} - \textstyle\sum_{l=0}^{L-1} h_l (g_w^{(1)} s_{t-o} \oplus g_w^{(2)} s_{(t-o)-1} \oplus g_w^{(3)} s_{(t-o)-2}) \right|^2 \end{aligned}, \tag{2.40}$$

where

$$u = \begin{cases} 1 & if \quad l = 0, n, 2n, \ldots \\ 3 & if \quad l = 1, n+1, 2n+1, \ldots \\ 2 & if \quad l = 2, n+2, 3n+1, \ldots \end{cases},$$

and $v = ((u+1) \mod 3)+1$ and $w = ((u+2) \mod 3)+1$. Also m, n and o starts at 0 and increases by 1 each time uncoded symbols from the previous time instant are used to create the coded symbol interfering with the current received symbol such that

$$m = \begin{cases} 0 & if \quad l = 0 \\ 1 & if \quad l = 1,2,3 \\ 2 & if \quad l = 4,5,6 \\ 3 & if \quad l = 7,8,9 \end{cases},$$

$$n = \begin{cases} 0 & if \quad l = 0,1 \\ 1 & if \quad l = 2,3,4 \\ 2 & if \quad l = 5,6,7 \\ 3 & if \quad l = 8,9 \end{cases},$$

$$o = \begin{cases} 0 & if \quad l = 0,1,2 \\ 1 & if \quad l = 3,4,5 \\ 2 & if \quad l = 6,7,8 \\ 3 & if \quad l = 9 \end{cases}.$$

Finally, since there are always only two equiprobable codewords that can follow each codeword, $P(s_t) = 0.5$. Therefore,

$$\omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) = \frac{1}{2\sqrt{2\pi}\sigma} \left(\frac{-\sum_{i=1}^{k} \Delta_{i,t}}{2\sigma^2} \right), \tag{2.41}$$

which completely describes the transition from state $S_{j,t-1}$ to $S_{i,t}$.

The model derived for the NI-JED without interleaving can be used to jointly equalize and decode BPSK modulated coded information transmitted through a multipath channel with CIR length L, where the convolutional encoder has constraint length K and the code rate is $R_c = k/n = 1/3$. This model can be extended for higher order modulation alphabets, larger encoder constraint lengths and different code rates.[6]

2.4.1.4 Computational Complexity

The computational complexity is determined by counting the number of computations performed for each received data block, and expressed in terms of the uncoded block length N_u, the CIR length L and

[6] Joint equalization and decoding using a higher order modulation alphabet will require convolutional encoding to be performed in GF(M), where M is the modulation alphabet size.

the encoder constraint length K and the modulation alphabet size M. The computational complexity of the NI-JED without interleaving, for the MLSE and MAP algorithms, is determined by

$$CC_{MLSE} = O(8KLN_u M^{Q+(K-1)}) \tag{2.42}$$

and

$$CC_{MAP} = O(2N_u M^{Q+(K-1)}(8KL+6)). \tag{2.43}$$

Fig. 2.6 shows the normalized computational complexity[7] of this joint equalizer and decoder, em-

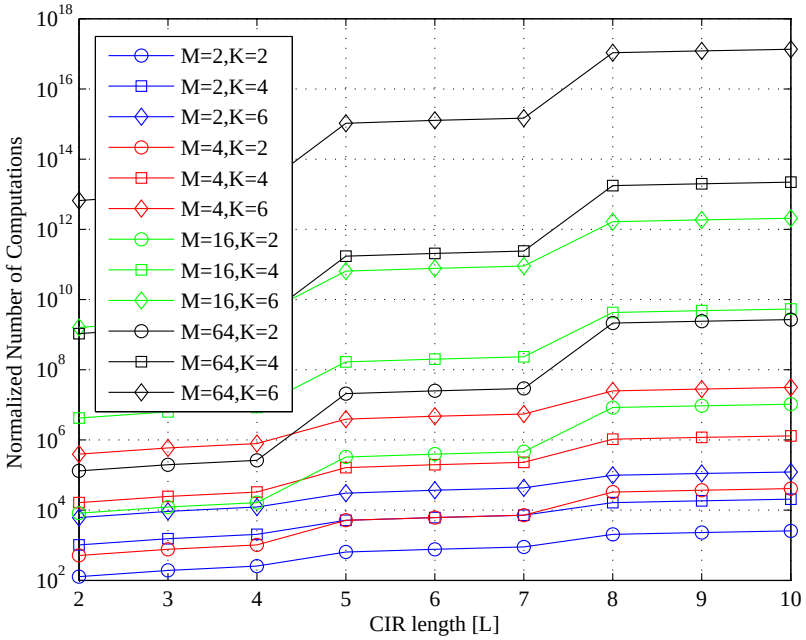


Figure 2.6: Non-iterative MLSE joint equalizer/decoder normalized computational complexity (without interleaver).

ploying the MLSE algorithm, for a CIR length of $L = 2$ to $L = 10$, constraint lengths $K = 2$, $K = 4$ and $K = 16$, and modulation alphabet sizes of $M = 2$, $M = 4$, $M = 16$ and $M = 64$. Fig. 2.7 shows the complexity of employing the MAP algorithm for the same parameters, where the respective complexity curves are only slightly higher than their corresponding MLSE complexity curves in Fig. 2.6. It is clear that the computational complexity grows exponentially with an increase in channel memory, encoder constraint length and modulation alphabet size, and it should be clear that computational complexity for long channel memories and large encoder constraint lengths is too high for feasible implementation.

[7] The computational complexity is normalized by the number of coded transmitted symbols N_c.

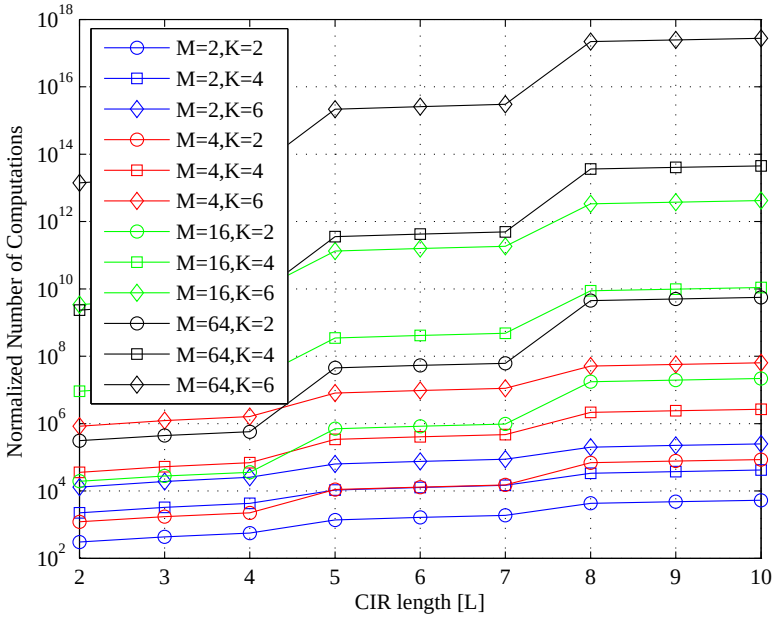


Figure 2.7: Non-iterative MAP joint equalizer/decoder normalized computational complexity (without interleaver).

2.4.2 Block Interleaving

In [14] it was demonstrated that equalization and decoding can be performed jointly by transforming the convolution decoder trellis into a super-trellis, and then using this super-trellis to perform equalization while decoding. This is a novel idea in the sense that all the available information is processed as a whole, and there is no exchange of information between independent subunits. This approach achieves optimal performance, because all calculations on the super-trellis are performed with complete information. The only limitation is that the interleaver has to be a block interleaver and that this interleaver has a certain depth limit due to the exponential relationship between the interleaver depth and the computational complexity.

Contrary to this approach, turbo equalization makes use of two independent subunits, namely a MAP equalizer and a MAP decoder, with each unit being supplied with information regarding the decisions made by the other unit [5–7]. This results in assumptions and estimations being made by the respective subunits based on incomplete information, which in turn results in suboptimal calculations during subsequent iterations, ultimately leading to suboptimal performance. Only by iteratively exchanging extrinsic information between the equalizer and decoder, where this information is used as prior in-

formation on the calculations in the next iteration, can the performance be increased. However, the turbo equalizer is not limited by the structure of the interleaver, since interleaving and deinterleaving are performed during each turbo iteration.

It was shown in [14] that the NI-JED performs optimally and therefore outperforms the turbo equalizer, but its computational complexity grows unimaginably as the interleaver depth and the CIR length increase. Despite the excellent performance of the NI-JED, it is not a viable solution for practical systems. Although the computational complexity of a CTE is not as high, it is still exponentially related to the encoder constraint length, as well as the channel memory length, but it is not influenced by the structure of the interleaver.

In this section this author attempts to derive a general model for the NI-JED proposed in [14], which can be presented as a function of encoder constraint length, interleaver depth and channel memory length. The authors of [14] were very vague as to how the algorithm functions. After explaining the effect of the interleaver on the coded information bits, and the subsequent effect after passing the information through a multipath channel, they simply show that the received symbols contain the uncoded information bits, and state that the MAP algorithm is "invoked." Nothing is said about the fact that the received symbols have to be deinterleaved, nor is it explained how the probabilities between state transitions are determined. The author of this thesis therefore presents the NI-JED for various interleaver depths and then derives a general model.

Suppose a communication system encodes a source bit sequence $\mathbf{s}$ of length N_u with a rate $R_c = k/n = 1/3$, constraint length $K = 3$, convolutional encoder with generator polynomials $g_1 = 4$ (100_2), $g_2 = 6$ (110_2) and $g_3 = 3$ (011_2) in order to produce a coded bit sequence $\mathbf{c}$ of length $N_c = N_u/R_c$, which is interleaved using a block interleaver, BPSK modulated and transmitted through a multipath channel with a CIR $\mathbf{h} = \{h_0, h_1\}$ of length $L = 2$. The encoder produces coded bits $\mathbf{c_t} = \{c_t^{(1)} c_t^{(2)} c_t^{(3)}\}$ from the uncoded bit s_t.

2.4.2.1 Interleaver Depth: $D = n = 3$

When using a $n \times N_u$ interleaver, as shown in Fig. 2.8, the coded sequence

$$\mathbf{c} = \{c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, \ldots\ldots, c_{N_u-1}^{(1)}, c_{N_u-1}^{(2)}, c_{N_u-1}^{(3)}, c_{N_u}^{(1)}, c_{N_u}^{(2)}, c_{N_u}^{(3)}\},$$

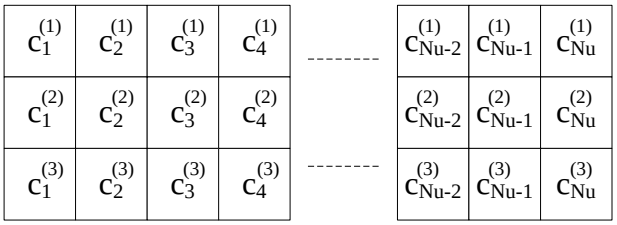


Figure 2.8: $n \times N_u$ interleaver.

is transformed to

$$\mathbf{c} = \{c_1^{(1)}, c_2^{(1)}, \ldots, c_{t-1}^{(1)}, c_N^{(1)} c_1^{(2)}, c_2^{(2)}, \ldots, c_{N_u-1}^{(2)}, c_{N_u}^{(2)}, c_1^{(3)}, c_2^{(3)}, \ldots, c_{N_u-1}^{(3)}, c_{N_u}^{(3)}\}$$

before transmission, grouping all the first, second and third encoder output bits together after interleaving. During transmission the interleaved coded symbols travel through a multipath channel $\mathbf{h} = \{h_0, h_1\}$ of length $L = 2$ and are received, as shown in the graphical model in Fig. 2.9.[8]

The noiseless received symbols can therefore be expressed in terms of the interleaved coded symbols

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)} h_0 + c_{t-1}^{(1)} h_1 \\ r_t^{(2)} &= c_t^{(2)} h_0 + c_{t-1}^{(2)} h_1 \\ r_t^{(3)} &= c_t^{(3)} h_0 + c_{t-1}^{(3)} h_1 \end{aligned} , \tag{2.44}$$

for $t = 1, 2, \ldots, N_u$, which in turn can be expressed in terms of the uncoded bits

$$\begin{aligned} r_t^{(1)} &= s_t h_0 + s_{t-1} h_1 \\ r_t^{(2)} &= (s_t \oplus s_{t-1}) h_0 + (s_{t-1} \oplus s_{t-2}) h_1 \\ r_t^{(3)} &= (s_{t-1} \oplus s_{t-2}) h_0 + (s_{t-2} \oplus s_{t-3}) h_1 \end{aligned} . \tag{2.45}$$

From (2.45) it is clear that $\{s_t, s_{t-1}, s_{t-2}, s_{t-3}\}$ is contained in $\mathbf{r} = \{r_t^{(1)}, r_t^{(2)}, r_t^{(3)}\}$. Therefore, the received symbol sequence $\mathbf{r}$ must be deinterleaved before joint equalization and decoding can commence.[9] Deinterleaving $\mathbf{r}$ will result in the sequence

$$\mathbf{r} = \{r_1^{(1)}, r_2^{(1)}, \ldots, r_{N_u-1}^{(1)}, r_{N_u}^{(1)}, r_1^{(2)}, r_2^{(2)}, \ldots, r_{N_u-1}^{(2)}, r_{N_u}^{(2)}, r_1^{(3)}, r_2^{(3)}, \ldots, r_{N_u-1}^{(3)}, r_{N_u}^{(3)}, \}$$

being transformed to

$$\mathbf{r} = \{r_1^{(1)}, r_1^{(2)}, r_1^{(3)}, r_2^{(1)}, r_2^{(2)}, r_2^{(3)}, \ldots, r_{N_u-1}^{(1)}, r_{N_u-1}^{(2)}, r_{N_u-1}^{(3)}, r_{N_u}^{(1)}, r_{N_u}^{(2)}, r_{N_u}^{(3)}\},$$

[8] Note that the channel coefficients are not shown in Fig. 2.9.

[9] This step in not mentioned, nor implied in [14], but has been inferred by the author of this thesis.

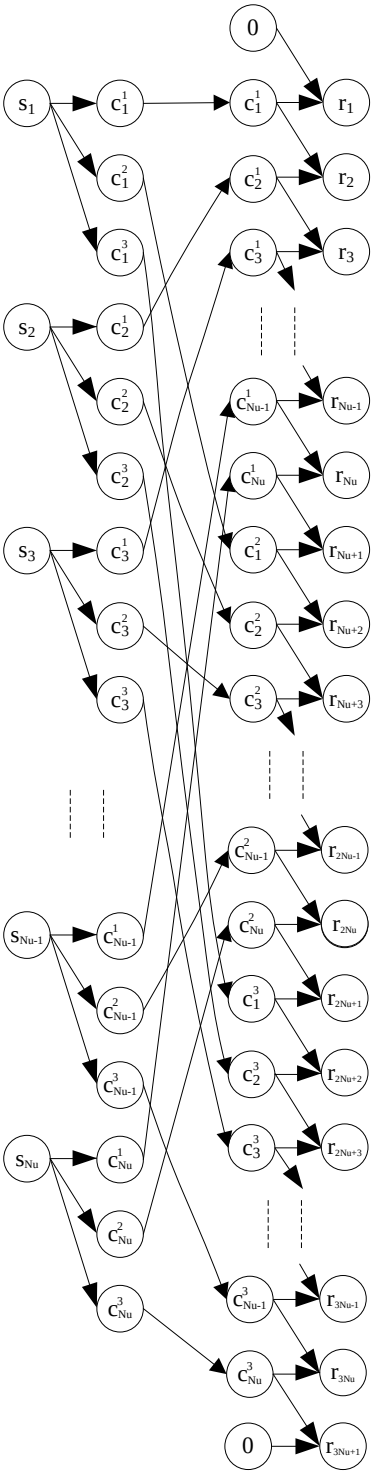


Figure 2.9: Graphical model for a system with a rate $R_c = 1/3$ convolutional encoder, a $3 \times N_u$ block interleaver and $L = 2$.

from which it is clear that all $\mathbf{r} = \{r_t^{(1)}, r_t^{(2)}, r_t^{(3)}\}$ are adjacent and can be used to estimate s_t using a trellis-based algorithm, assuming $\{s_{t-1}, s_{t-2}, s_{t-3}\}$ is known from previous states.

The NI-JED uses the sum-product algorithm,[10] as explained in *Section 2.1.2.1*, and the number of super-trellis states are determined by the interleaver depth (D), the channel memory length ($L-1$) as well as the number of encoder memory elements ($K-1$). Since BPSK modulation is used, the number of states in the super-trellis is $M = 2^{D(L-1)+(K-1)}$ [14]. Therefore $M = 8$. The number of states can also be determined by the range of source bits contained in (2.45) minus one, since the trellis has to model state transitions $\{s_{t-1}, s_{t-2}, s_{t-3}\}$ to $\{s_t, s_{t-1}, s_{t-2}\}$, which when combined yields $\{s_t, s_{t-1}, s_{t-2}, s_{t-3}\}$. Therefore the number of states can also be determined by $M = 2^{4-1} = 8$.

The task of the NI-JED is to produced the maximum a posteriori probability of each uncoded source symbol at time t from the deinterleaved received symbol sequence, or

$$P(s_t|\mathbf{r}) = \sum_{s_{t^\dagger}:t^\dagger \neq t}^{N_c} P(\mathbf{s}|\mathbf{r}). \tag{2.46}$$

The probability of a transition from state $S_{j,t-1}$ to state $S_{i,t}$, where $i, j = 1, 2, \ldots, M$ is given by

$$\omega_{j\to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)}|S_{j,t-1}, S_{i,t})P(s_t), \tag{2.47}$$

where the symbols associated with states $S_{j,t-1}$ ($\{s_{t-1}, s_{t-2}, s_{t-3}\}$) and $S_{i,t}$ ($\{s_t, s_{t-1}, s_{t-2}\}$) can be combined so that

$$\omega_{j\to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)}|s_t, s_{t-1}, s_{t-2}, s_{t-3})P(s_t), \tag{2.48}$$

where

$$P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)}|s_t, s_{t-1}, s_{t-2}, s_{t-3}) = \frac{1}{\sqrt{2\pi}\sigma}\left(\frac{-\sum_{i=1}^{n}\Delta_{i,t}}{2\sigma^2}\right). \tag{2.49}$$

$\Delta_{i,t}$ can be determined by minimizing the receiver equations in (2.45) such that

$$\begin{aligned}
\Delta_{1,t} &= \left|r_t^{(1)} - \textstyle\sum_{l=0}^{L-1} h_l(s_{t-l})\right|^2 \\
\Delta_{2,t} &= \left|r_t^{(2)} - \textstyle\sum_{l=0}^{L-1} h_l(s_{t-l} \oplus s_{(t-1)-l})\right|^2 \\
\Delta_{3,t} &= \left|r_t^{(3)} - \textstyle\sum_{l=0}^{L-1} h_l(s_{(t-1)-l} \oplus s_{(t-2)-l})\right|^2
\end{aligned}, \tag{2.50}$$

where the calculation of each $\Delta_{i,t}$, $i = 1, 2, \ldots, n$ is determined by the structure of the encoder. Finally, since there are only two equiprobable bits that can be transmitted, $P(s_t) = 0.5$. Therefore,

$$\omega_{j\to i,t}(S_{j,t-1}, S_{i,t}) = \frac{1}{2\sqrt{2\pi}\sigma}\left(\frac{-\sum_{i=1}^{n}\Delta_{i,t}}{2\sigma^2}\right), \tag{2.51}$$

which completely describes the transition from state $S_{j,t-1}$ to $S_{i,t}$.

[10] The min-sum algorithm can also be used.

2.4.2.2 Interleaver Depth: $D = 2n = 6$

In the above explanation the depth of the block interleaver was $D = n$. It is subsequently assumed that an interleaver with depth $D = 2n$ is used. Applying the $2n \times N_u/2$ interleaver in Fig. 2.10 to the coded symbols

$$\mathbf{c} = \{c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, \ldots\ldots, c_{N_u-1}^{(1)}, c_{N_u-1}^{(2)}, c_{N_u-1}^{(3)}, c_{N_u}^{(1)}, c_{N_u}^{(2)}, c_{N_u}^{(3)}\},$$

transforms it to

$$\begin{aligned}\mathbf{c} \;=\; \{&c_1^{(1)}, c_3^{(1)}, \ldots, c_{N_u-1}^{(1)}, c_1^{(2)}, c_3^{(2)}, \ldots, c_{N_c-1}^{(2)}, c_1^{(3)}, c_3^{(3)}, \ldots, c_{N_c-1}^{(3)}, \ldots \\ &c_2^{(1)}, c_4^{(1)}, \ldots, c_{N_c}^{(1)}, c_2^{(2)}, c_4^{(2)}, \ldots, c_{N_c}^{(2)}, c_2^{(3)}, c_4^{(3)}, \ldots, c_{N_c}^{(3)}\}.\end{aligned} \tag{2.52}$$

The received symbols can be expressed in terms of the coded symbols

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)} h_0 + c_{t-2}^{(1)} h_1 \\ r_t^{(2)} &= c_t^{(2)} h_0 + c_{t-2}^{(2)} h_1 \\ r_t^{(3)} &= c_t^{(3)} h_0 + c_{t-2}^{(3)} h_1 \end{aligned}, \tag{2.53}$$

where $t = 1, 2, \ldots, N_u$, which in turn can be expressed in terms of the uncoded bits

$$\begin{aligned} r_t^{(1)} &= s_t h_0 + s_{t-2} h_1 \\ r_t^{(2)} &= (s_t \oplus s_{t-1}) h_0 + (s_{t-2} \oplus s_{t-3}) h_1 \\ r_t^{(3)} &= (s_{t-1} \oplus s_{t-2}) h_0 + (s_{t-3} \oplus s_{t-4}) h_1 \end{aligned}. \tag{2.54}$$

Comparing (2.53) and (2.54) with (2.44) and (2.45) respectively reveals that the system memory has doubled with a twofold increase in interleaver depth (from $D = n$ to $D = 2n$).

$c_1^{(1)}$	$c_3^{(1)}$	$c_5^{(1)}$	$c_{Nu-3}^{(1)}$	$c_{Nu-1}^{(1)}$
$c_1^{(2)}$	$c_3^{(2)}$	$c_5^{(2)}$	$c_{Nu-3}^{(2)}$	$c_{Nu-1}^{(2)}$
$c_1^{(3)}$	$c_3^{(3)}$	$c_5^{(3)}$	$c_{Nu-3}^{(3)}$	$c_{Nu-1}^{(3)}$
$c_2^{(1)}$	$c_4^{(1)}$	$c_6^{(1)}$	$c_{Nu-2}^{(1)}$	$c_{Nu}^{(1)}$
$c_2^{(2)}$	$c_4^{(2)}$	$c_6^{(2)}$	$c_{Nu-2}^{(2)}$	$c_{Nu}^{(2)}$
$c_2^{(3)}$	$c_4^{(3)}$	$c_6^{(3)}$	$c_{Nu-2}^{(3)}$	$c_{Nu}^{(3)}$

Figure 2.10: $2n \times N_u/2$ interleaver.

As before, the received symbol sequence $\mathbf{r}$ must be deinterleaved. Deinterleaving $\mathbf{r}$ will result in the sequence

$$\begin{aligned}\mathbf{r} = \{&r_1^{(1)}, r_3^{(1)}, \ldots, r_{N_u-1}^{(1)}, r_1^{(2)}, r_3^{(2)}, \ldots, r_{N_r-1}^{(2)}, r_1^{(3)}, r_3^{(3)}, \ldots, r_{N_r-1}^{(3)}, \ldots \\ &r_2^{(1)}, r_4^{(1)}, \ldots, r_{N_r}^{(1)}, r_2^{(2)}, r_4^{(2)}, \ldots, r_{N_r}^{(2)}, r_2^{(3)}, r_4^{(3)}, \ldots, r_{N_r}^{(3)}\}.\end{aligned}$$

being transformed to

$$\mathbf{r} = \{r_1^{(1)}, r_1^{(2)}, r_1^{(3)}, r_2^{(1)}, r_2^{(2)}, r_2^{(3)}, \ldots\ldots, r_{N_u-1}^{(1)}, r_{N_u-1}^{(2)}, r_{N_u-1}^{(3)}, r_{N_u}^{(1)}, r_{N_u}^{(2)}, r_{N_u}^{(3)}\},$$

which ensures that all $\mathbf{r} = \{r_t^{(1)}, r_t^{(2)}, r_t^{(3)}\}$ are adjacent and can be used to estimate s_t, assuming that $\{s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4}\}$ is known from previous states.

As explained earlier, the number of super-trellis states can be determined by the range of source bits in (2.54) minus one. Counting the range of source bits, minus one, gives the number of states $M = 2^{5-1} = 16$. Alternatively the number of trellis states can be determined by $M = 2^{D(L-1)+(K-1)}$ which gives $M = 2^{2(1)+2} = 16$. Therefore, for an interleaver with depth $D = 2n$, the calculation of state transition probabilities will require four bits per state.

As before, the probability of a transition from state $S_{j,t-1}$ to state $S_{i,t}$, where $i, j = 1, 2, \ldots, M$ is given by

$$\omega_{j\to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | S_{j,t-1}, S_{i,t}) P(s_t), \tag{2.55}$$

which can be written as

$$\omega_{j\to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | s_t, s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4}) P(s_t), \tag{2.56}$$

where

$$P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | s_t, s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4}) = \frac{1}{\sqrt{2\pi}\sigma} \left(\frac{-\sum_{i=1}^{n} \Delta_{i,t}}{2\sigma^2} \right). \tag{2.57}$$

$\Delta_{i,t}$ can be determined by minimizing the receiver equations in (2.54) such that

$$\begin{aligned}\Delta_{1,t} &= \left| r_t^{(1)} - \textstyle\sum_{l=0}^{L-1} h_l(s_{t-2l}) \right|^2 \\ \Delta_{2,t} &= \left| r_t^{(1)} - \textstyle\sum_{l=0}^{L-1} h_l(s_{t-2l} \oplus s_{(t-1)-2l}) \right|^2 \\ \Delta_{3,t} &= \left| r_t^{(1)} - \textstyle\sum_{l=0}^{L-1} h_l(s_{(t-1)-2l} \oplus s_{(t-2)-2l}) \right|^2\end{aligned}. \tag{2.58}$$

By comparing (2.58) and (2.50), it is already clear that the interleaver depth has an effect on the computational complexity of the NI-JED. Apart from the increase in the number of super-trellis states as a function of interleaver depth, this is the only part of the algorithm that is different from when the interleaver depth is $D = n$. Therefore, the transition probabilities are calculated as before.

2.4.2.3 Interleaver Depth: $D = 3n = 9$

In order to derive a general model for the NI-JED, the system parameters with different interleaver depths are analyzed. Here it is assumed that the $3n \times N_u/3$ interleaver in Fig. 2.11 is used. Interleaving the coded symbols

$$\mathbf{c} = \{c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, \ldots\ldots, c_{N_u-1}^{(1)}, c_{N_u-1}^{(2)}, c_{N_u-1}^{(3)}, c_{N_u}^{(1)}, c_{N_u}^{(2)}, c_{N_u}^{(3)}\},$$

with a $3n \times N_u/3$ interleaver, will result in

$$\begin{aligned}\mathbf{c} = \{ \quad & c_1^{(1)}, c_4^{(1)}, c_7^{(1)}, \ldots, c_{N_u-2}^{(1)}, c_1^{(2)}, c_4^{(2)}, c_7^{(2)}, \ldots, c_{N_u-2}^{(2)}, c_1^{(3)}, c_4^{(3)}, c_7^{(3)}, \ldots, c_{N_u-2}^{(3)}, \ldots \\ & c_2^{(1)}, c_5^{(1)}, c_6^{(1)}, \ldots, c_{N_u-1}^{(1)}, c_2^{(2)}, c_5^{(2)}, c_6^{(2)}, \ldots, c_{N_u-1}^{(2)}, c_2^{(3)}, c_5^{(3)}, c_6^{(3)}, \ldots, c_{N_u-1}^{(3)}, \ldots \\ & c_3^{(1)}, c_6^{(1)}, c_9^{(1)}, \ldots, c_{N_u}^{(1)}, c_2^{(2)}, c_5^{(2)}, c_9^{(2)}, \ldots, c_{N_u}^{(2)}, c_2^{(3)}, c_5^{(3)}, c_9^{(3)}, \ldots, c_{N_u}^{(3)} \quad \}\end{aligned}.$$

Again the received symbols can be expressed in terms of the coded symbols

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)} h_0 + c_{t-3}^{(1)} h_1 \\ r_t^{(2)} &= c_t^{(2)} h_0 + c_{t-3}^{(2)} h_1 \\ r_t^{(3)} &= c_t^{(3)} h_0 + c_{t-3}^{(3)} h_1 \end{aligned}, \tag{2.59}$$

where $t = 1, 2, \ldots, N_u$, which can be expressed in terms of the uncoded bits

$$\begin{aligned} r_t^{(1)} &= s_t h_0 + s_{t-3} h_1 \\ r_t^{(2)} &= (s_t \oplus s_{t-1}) h_0 + (s_{t-3} \oplus s_{t-4}) h_1 \\ r_t^{(3)} &= (s_{t-1} \oplus s_{t-2}) h_0 + (s_{t-4} \oplus s_{t-5}) h_1 \end{aligned}, \tag{2.60}$$

from which it is clear that the system memory is once again affected by the increase in interleaver depth.

$c_1^{(1)}$	$c_4^{(1)}$	$c_7^{(1)}$		$c_{Nu-5}^{(1)}$ $c_{Nu-2}^{(1)}$
$c_1^{(2)}$	$c_4^{(2)}$	$c_7^{(2)}$		$c_{Nu-5}^{(2)}$ $c_{Nu-2}^{(2)}$
$c_1^{(3)}$	$c_4^{(3)}$	$c_7^{(3)}$		$c_{Nu-5}^{(3)}$ $c_{Nu-2}^{(3)}$
$c_2^{(1)}$	$c_5^{(1)}$	$c_8^{(1)}$		$c_{Nu-4}^{(1)}$ $c_{Nu-1}^{(1)}$
$c_2^{(2)}$	$c_5^{(2)}$	$c_8^{(2)}$		$c_{Nu-4}^{(2)}$ $c_{Nu-1}^{(2)}$
$c_2^{(3)}$	$c_5^{(3)}$	$c_8^{(3)}$		$c_{Nu-4}^{(3)}$ $c_{Nu-1}^{(3)}$
$c_3^{(1)}$	$c_6^{(1)}$	$c_9^{(1)}$		$c_{Nu-3}^{(1)}$ $c_{Nu}^{(1)}$
$c_3^{(2)}$	$c_6^{(2)}$	$c_9^{(2)}$		$c_{Nu-3}^{(2)}$ $c_{Nu}^{(2)}$
$c_3^{(3)}$	$c_6^{(3)}$	$c_9^{(3)}$		$c_{Nu-3}^{(3)}$ $c_{Nu}^{(3)}$

Figure 2.11: $3n \times N_u/3$ interleaver.

Deinterleaving $\mathbf{r}$

$$\mathbf{r} = \{ \begin{array}{l} r_1^{(1)}, r_4^{(1)}, r_7^{(1)}, \ldots, r_{N_u-2}^{(1)}, r_1^{(2)}, r_4^{(2)}, r_7^{(2)}, \ldots, r_{N_u-2}^{(2)}, r_1^{(3)}, r_4^{(3)}, r_7^{(3)}, \ldots, r_{N_u-2}^{(3)}, \ldots \\ r_2^{(1)}, r_5^{(1)}, r_6^{(1)}, \ldots, r_{N_u-1}^{(1)}, r_2^{(2)}, r_5^{(2)}, r_6^{(2)}, \ldots, r_{N_u-1}^{(2)}, r_2^{(3)}, r_5^{(3)}, r_6^{(3)}, \ldots, r_{N_u-1}^{(3)}, \ldots \\ r_3^{(1)}, r_6^{(1)}, r_9^{(1)}, \ldots, r_{N_u}^{(1)}, r_3^{(2)}, r_6^{(2)}, r_9^{(2)}, \ldots, r_{N_u}^{(2)}, r_3^{(3)}, r_6^{(3)}, r_9^{(3)}, \ldots, r_{N_u}^{(3)} \end{array} \}$$

results in

$$\mathbf{r} = \{r_1^{(1)}, r_1^{(2)}, r_1^{(3)}, r_2^{(1)}, r_2^{(2)}, r_2^{(3)}, \ldots\ldots, r_{N_u-1}^{(1)}, r_{N_u-1}^{(2)}, r_{N_u-1}^{(3)}, r_{N_u}^{(1)}, r_{N_u}^{(2)}, r_{N_u}^{(3)}\},$$

which ensures that all $\mathbf{r} = \{r_t^{(1)}, r_t^{(2)}, r_t^{(3)}\}$ are adjacent and can be used to estimate s_t, assuming that $\{s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4}, s_{t-5}\}$ is known from previous states. Therefore the number of super-trellis states for an interleaver depth of $D = 3n$ and channel memory length $L-1$ is $M = 2^{D(L-1)+(K-1)}$ which gives $M = 2^{3(1)+2} = 32$.

The probability of a transition from state $S_{j,t-1}$ to state $S_{i,t}$, where $i, j = 1, 2, \ldots, M$ is given by

$$\omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | S_{j,t-1}, S_{i,t}) P(s_t), \tag{2.61}$$

which can be written as

$$\omega_{j \to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | s_t, s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4}, s_{t-5}) P(s_t). \tag{2.62}$$

where

$$P(r_t^{(1)}, r_t^{(2)}, r_t^{(3)} | s_t, s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4}, s_{t-5}) = \frac{1}{\sqrt{2\pi}\sigma} \left(\frac{-\sum_{i=1}^{n} \Delta_{i,t}}{2\sigma^2} \right). \tag{2.63}$$

$\Delta_{i,t}$ can be determined by minimizing the receiver equations in (2.54) such that

$$\begin{aligned} \Delta_{1,t} &= \left| r_t^{(1)} - \sum_{l=0}^{L-1} h_l(s_{t-3l}) \right|^2 \\ \Delta_{2,t} &= \left| r_t^{(1)} - \sum_{l=0}^{L-1} h_l(s_{t-3l} \oplus s_{(t-1)-3l}) \right|^2 \\ \Delta_{3,t} &= \left| r_t^{(1)} - \sum_{l=0}^{L-1} h_l(s_{(t-1)-3l} \oplus s_{(t-2)-3l}) \right|^2 \end{aligned}. \tag{2.64}$$

2.4.2.4 Interleaver Depth: $D = dn$

After deriving the NI-JED for various interleaver depths it is now possible to present a general model for this algorithm for any $nk \times N_u/d$ interleaver. Even though the channel memory length $L-1$ remained fixed at $L-1=1$, and the number of encoder output bits remained fixed at $n=3$ throughout the analysis, it is easy to generalize the model for any L and n, as will be shown.

When an $nk \times N_u/d$ interleaver is used to interleave the coded symbol sequence

$$\mathbf{c} = \{c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, \ldots\ldots, c_{N_u-1}^{(1)}, c_{N_u-1}^{(2)}, c_{N_u-1}^{(3)}, c_{N_u}^{(1)}, c_{N_u}^{(2)}, c_{N_u}^{(3)}\},$$

encoded with an encoder that produces n output bits, the result will be

$$\begin{aligned} \mathbf{c} = \{ \quad & c_1^{(1)}, c_{1+d}^{(1)}, c_{1+2d}^{(1)}, \ldots, c_{N_u-(d-1)}^{(1)}, c_1^{(2)}, c_{1+d}^{(2)}, c_{1+2d}^{(2)}, \ldots, c_{N_u-(d-1)}^{(2)}, \ldots, c_1^{(n)}, c_{1+d}^{(n)}, c_{1+2d}^{(n)}, \ldots, c_{N_u-(d-1)}^{(n)}, \cdots \\ & c_2^{(1)}, c_{2+d}^{(1)}, c_{1+2d}^{(1)}, \ldots, c_{N_u-(d-2)}^{(1)}, c_2^{(2)}, c_{2+d}^{(2)}, c_{2+2d}^{(2)}, \ldots, c_{N_u-(d-2)}^{(2)}, \ldots, c_2^{(n)}, c_{2+d}^{(n)}, c_{2+2d}^{(n)}, \ldots, c_{N_u-(d-2)}^{(n)}, \cdots \\ & \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ & c_d^{(1)}, c_{d+d}^{(1)}, c_{d+2d}^{(1)}, \ldots, c_{N_u}^{(1)}, c_d^{(2)}, c_{d+d}^{(2)}, c_{d+2d}^{(2)}, \ldots, c_{N_u}^{(2)}, \ldots, c_d^{(n)}, c_{d+d}^{(n)}, c_{d+3d}^{(n)}, \ldots, c_{N_u}^{(n)}\}. \end{aligned}$$

Transmitting the interleaved code symbol sequence through a multipath channel with a CIR length of L, where the encoder produces n output bits at a rate of $R_c = 1/n$, the received symbols can be expressed in terms of the coded symbols

$$\begin{aligned} r_t^{(1)} &= c_t^{(1)} h_0 + c_{t-d}^{(1)} h_1 + c_{t-2d}^{(1)} h_2 + \ldots + c_{t-(L-1)d}^{(1)} h_{L-1} \\ r_t^{(2)} &= c_t^{(2)} h_0 + c_{t-d}^{(2)} h_1 + c_{t-2d}^{(2)} h_2 + \ldots + c_{t-(L-1)d}^{(2)} h_{L-1} \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ r_t^{(n)} &= c_t^{(n)} h_0 + c_{t-d}^{(n)} h_1 + c_{t-2d}^{(n)} h_2 + \ldots + c_{t-(L-1)d}^{(n)} h_{L-1} \end{aligned}, \tag{2.65}$$

which can be rewritten as

$$\begin{aligned} r_t^{(1)} &= \sum_{l=0}^{L-1)} c_{t-ld}^{(1)} h_l \\ r_t^{(2)} &= \sum_{l=0}^{L-1)} c_{t-ld}^{(2)} h_l \\ &\vdots \qquad\qquad \vdots \\ r_t^{(n)} &= \sum_{l=0}^{L-1)} c_{t-ld}^{(3)} h_l \end{aligned}, \tag{2.66}$$

which can be expressed in terms of the uncoded bits

$$\begin{array}{rcl} r_t^{(1)} & = & \sum_{l=0}^{L-1}\left((g_1^{(1)}s_{t-ld})\oplus(g_1^{(2)}s_{(t-1)-ld})\oplus(g_1^{(3)}s_{(t-2)-ld})\oplus\ldots\oplus(g_1^{(K)}s_{(t-(K-1))-ld})\right)h_l \\ r_t^{(2)} & = & \sum_{l=0}^{L-1}\left((g_2^{(1)}s_{t-ld})\oplus(g_2^{(2)}s_{(t-1)-ld})\oplus(g_2^{(3)}s_{(t-2)-ld})\oplus\ldots\oplus(g_2^{(K)}s_{(t-(K-1))-ld})\right)h_l \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ r_t^{(n)} & = & \sum_{l=0}^{L-1}\left((g_n^{(1)}s_{t-ld})\oplus(g_n^{(2)}s_{(t-1)-ld})\oplus(g_n^{(3)}s_{(t-2)-ld})\oplus\ldots\oplus(g_n^{(K)}s_{(t-(K-1))-ld})\right)h_l \end{array}, \tag{2.67}$$

where K is the encoder constraint length and generator polynomials $g_1 = \{g_1^{(1)}, g_1^{(2)}, \ldots, g_1^{(K)}\}$, $g_2 = \{g_2^{(1)}, g_2^{(2)}, \ldots, g_2^{(K)}\}$ and $g_n = \{g_n^{(1)}, g_n^{(2)}, \ldots, g_n^{(K)}\}$. Deinterleaving $\mathbf{r}$ will result in the sequence

$$\begin{array}{rl} \mathbf{r} = \{ & r_1^{(1)}, r_{1+d}^{(1)}, r_{1+2d}^{(1)}, \ldots, r_{N_u-(d-1)}^{(1)}, r_1^{(2)}, r_{1+d}^{(2)}, r_{1+2d}^{(2)}, \ldots, r_{N_u-(d-1)}^{(2)}, \ldots, r_1^{(n)}, r_{1+d}^{(n)}, r_{1+2d}^{(n)}, \ldots, r_{N_u-(d-1)}^{(n)}, \ldots \\ & r_2^{(1)}, r_{2+d}^{(1)}, r_{1+2d}^{(1)}, \ldots, r_{N_u-(d-2)}^{(1)}, r_2^{(2)}, r_{2+d}^{(2)}, r_{2+2d}^{(2)}, \ldots, r_{N_u-(d-2)}^{(2)}, \ldots, r_2^{(n)}, r_{2+d}^{(n)}, r_{2+2d}^{(n)}, \ldots, r_{N_u-(d-2)}^{(n)}, \ldots \\ & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ & r_d^{(1)}, r_{d+d}^{(1)}, r_{d+2d}^{(1)}, \ldots, r_{N_u}^{(1)}, r_d^{(2)}, r_{d+d}^{(2)}, r_{d+2d}^{(2)}, \ldots, r_{N_u}^{(2)}, \ldots, r_d^{(n)}, r_{d+d}^{(n)}, r_{d+3d}^{(n)}, \ldots, r_{N_u}^{(n)}\}. \end{array}$$

being transformed to

$$\mathbf{r} = \{r_1^{(1)}, \ldots, r_1^{(n)}, r_2^{(1)}, \ldots, r_2^{(n)}, \ldots\ldots, r_{N_u-1}^{(1)}, \ldots, r_{N_u-1}^{(n)}, r_{N_u}^{(1)}, r_{N_u}^{(2)}, r_{N_u}^{(3)}\},$$

from which it is clear that all $\mathbf{r} = \{r_t^{(1)}, \ldots, r_t^{(n)}\}$ are adjacent and can be used to estimate s_t using a trellis-based algorithm, assuming $\{s_{t-1}, s_{t-2}, s_{t-3}, \ldots, s_{t-(D(L-1)+(K-1))}\}$ is known from previous states. The number of super-trellis states for an interleaver depth of $D = dn$, channel memory length $L-1$, and encoder constraint length K, is $M = 2^{D(L-1)+(K-1)}$.

The probability of a transition from state $S_{j,t-1}$ to state $S_{i,t}$, where $i, j = 1, 2, \ldots, M$ is given by

$$\omega_{j\to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, \ldots, r_t^{(n)} | S_{j,t-1}, S_{i,t}) P(s_t), \tag{2.68}$$

which can be written as

$$\omega_{j\to i,t}(S_{j,t-1}, S_{i,t}) = P(r_t^{(1)}, \ldots, r_t^{(n)} | s_t, s_{t-1}, s_{t-2}, \ldots, s_{t-(d(L-1)+(K-1))}) P(s_t), \tag{2.69}$$

where

$$P(r_t^{(1)}, \ldots, r_t^{(n)} | s_t, s_{t-1}, s_{t-2}, \ldots, s_{t-(d(L-1)+(K-1))}) = \frac{1}{\sqrt{2\pi}\sigma}\left(\frac{-\sum_{i=1}^{n}\Delta_{i,t}}{2\sigma^2}\right) \tag{2.70}$$

and $P(s_t) = 0.5$ as before.

$\Delta_{i,t}$ can be determined by minimizing the receiver equations in (2.67) such that

$$\begin{array}{rcl} \Delta_{1,t} = & \left| r_t^{(1)} - \sum_{l=0}^{L-1}\left((g_1^{(1)}s_{t-ld})\oplus(g_1^{(2)}s_{(t-1)-ld})\oplus\ldots\oplus(g_1^{(K)}s_{(t-(K-1))-ld})\right)h_l \right|^2 \\ \Delta_{2,t} = & \left| r_t^{(2)} - \sum_{l=0}^{L-1}\left((g_2^{(1)}s_{t-ld})\oplus(g_2^{(2)}s_{(t-1)-ld})\oplus\ldots\oplus(g_2^{(K)}s_{(t-(K-1))-ld})\right)h_l \right|^2 \\ \vdots & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \Delta_{n,t} = & \left| r_t^{(n)} - \sum_{l=0}^{L-1}\left((g_n^{(1)}s_{t-ld})\oplus(g_n^{(2)}s_{(t-1)-ld})\oplus\ldots\oplus(g_n^{(K)}s_{(t-(K-1))-ld})\right)h_l \right|^2 \end{array}. \tag{2.71}$$

This algorithm can therefore be used to jointly, non-iteratively equalize and decode BPSK modulated information, encoded with a rate $R_c = 1/n$, constraint length K convolutional encoder, transmitted through a multipath channel with a CIR length of L, using an interleaver with depth $D = dn$. This concludes the derivation of a general model for the NI-JED.

2.4.2.5 Computational Complexity

The computational complexity of the NI-JED, with block interleaving, is determined by counting the number of computations performed for each data block received, and expressed in terms of the uncoded block length N_u, the CIR length L and the encoder constraint length K, the modulation alphabet size M and the interleaver depth D. The computational complexity of the NI-JED without interleaving, using the MAP algorithm, is determined by

$$CC_{MAP} = O(2N_u M^{D(L-1)+(K-1)}(8KL+6)). \tag{2.72}$$

Fig. 2.12 shows the normalized computational complexity for a CIR length of $L = 2$ to $L = 10$, encoder constraint lengths $K = 2$, $K = 4$, $K = 6$, a modulation alphabet size of $M = 2$ and interleaver depths of $D = 3$, $D = 6$, $D = 9$ and $D = 12$. It is clear that the computational complexity grows exponentially with an increase in channel memory, encoder constraint length and interleaver depth. Fig. 2.13 shows the normalized computational complexity for the same parameters but with a fixed interleaver depth of $D = 6$ and varying modulation alphabet sizes of $M = 2$, $M = 4$, $M = 16$ and $M = 64$, resulting in an increase in complexity with an increase in modulation alphabet size.

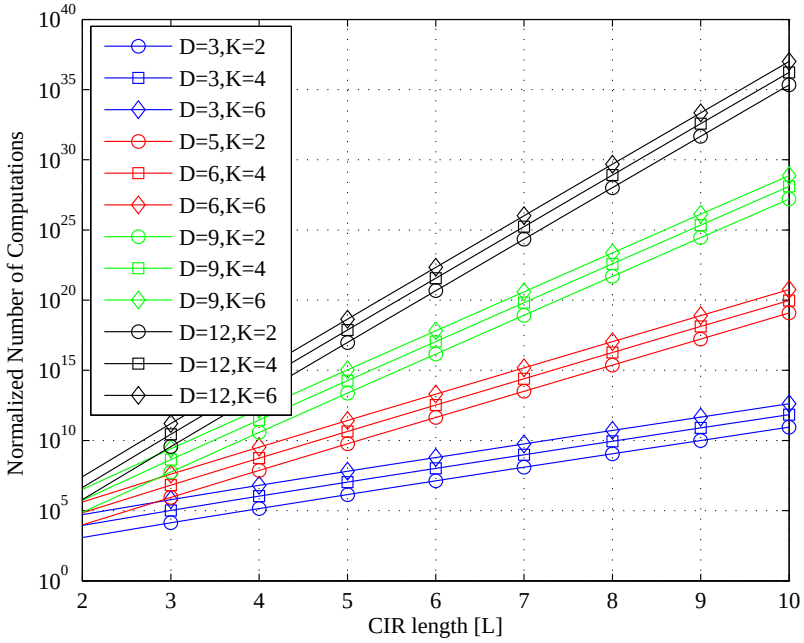


Figure 2.12: NI-JED normalized computational complexity (with a block interleaver) for M=2.

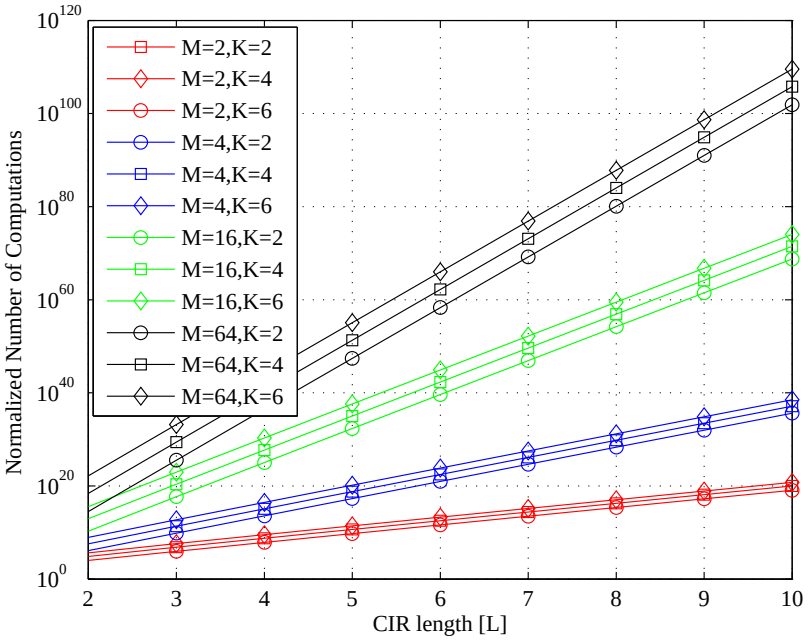


Figure 2.13: NI-JED normalized computational complexity (with a block interleaver) for D=6.

2.5 SIMULATION RESULTS

Parameter	**Setting**
Modulation	BPSK
Interleaver	None & Block
Interleaver depth (D)	1, k, 2k, 3k
Uncoded block length (N_u)	600
Coded block length (N_c)	1800
Channel	Static
CIR length (L)	2, 3, 4
Channel Estimation	No
Channel State Information	Perfect CSI
PDP	Uniform

The performance of the NI-JED is evaluated with and without interleaving, and it is compared to that of a CTE, which will be discussed at length in the next chapter, where the number of CTE iterations is $Z = 10$. The CIR $\mathbf{h} = \{h_0, h_1, \ldots, h_{L-1}\}$ of length L is normalized such that $\mathbf{h}'\mathbf{h} = 1$, the uncoded and coded block lengths are $N_u = 600$ and $N_u = 1800$ respectively, the various channel lengths are $L = 2$, $L = 3$ and $L = 4$, and interleaver depths of $D = 1$, $D = k$, $D = 2k$ and $D = 3k$ were used, where $k = 3$ is the number of encoder output bits. For details regarding the simulation setup used throughout this thesis please refer to *Appendix A*.

Fig. 2.14 shows the BER performance of the NI-JED compared to that of the CTE, for various interleaver depths and a channel length of $L = 2$. From Fig. 2.14 it can be seen that the performance of both algorithms improves with an increase in the interleaver depth, except for an increase from $D = 1$ to $D = 3$. It is clear that the NI-JED outperforms the CTE, even thought the CTE uses multiple iterations, while the NI-JED performs close to the coded additive white Gaussian noise (AWGN) bound when the interleaver depth is $D = 9$. In Fig. 2.15 and Fig. 2.16 the performance of the NI-JED is again compared to that of the CTE for various interleaver depths, for $L = 3$ and $L = 4$ respectively. As before, the NI-JED outperforms the CTE. From Fig. 2.14 through to Fig. 2.16 it is clear that the NI-JED is superior in terms of performance, and it is in fact optimal. Even though the NI-JED outperforms the CTE, its vast computational complexity inhibits it from finding practical application. The CTE is therefore used as an alternative solution in practical systems.

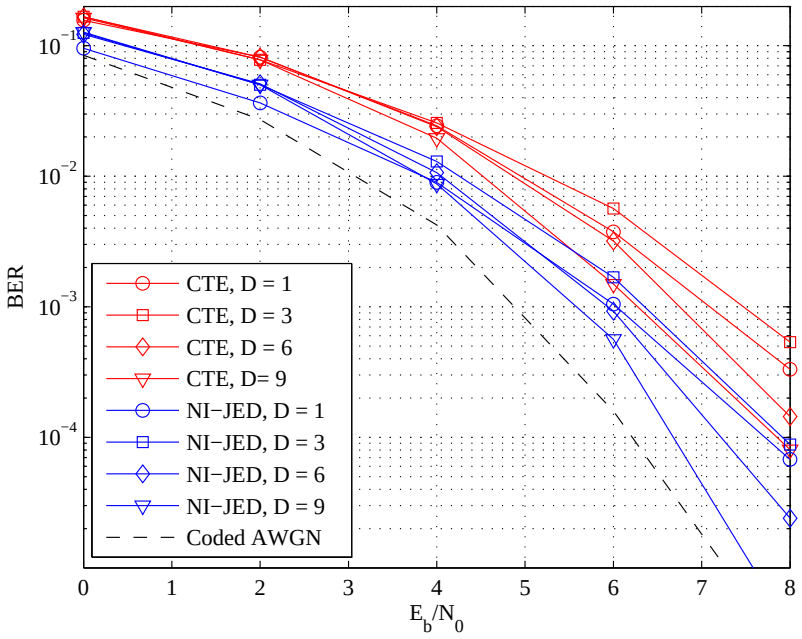


Figure 2.14: NI-JED and CTE performance for interleaver depths $D = 1$, $D = k$, $D = 2k$ and $D = 3k$, and CIR length $L = 2$.

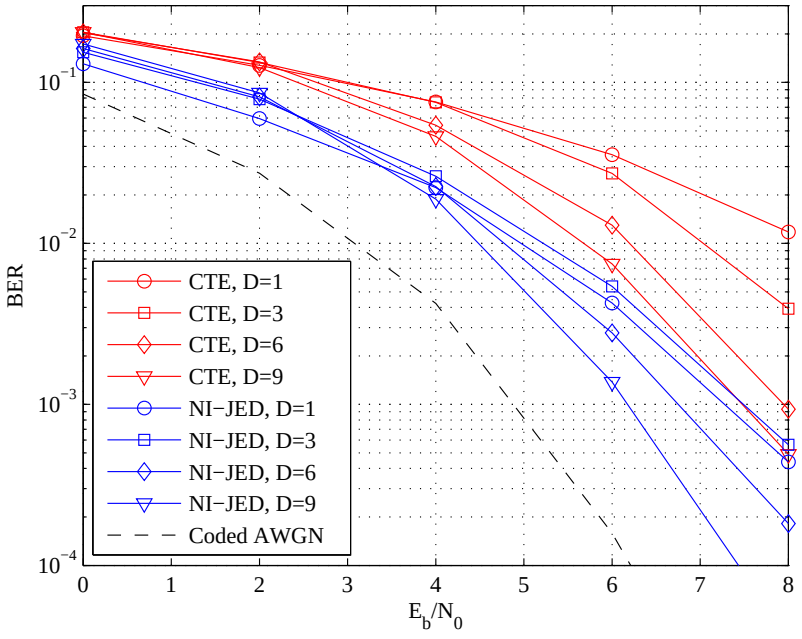


Figure 2.15: NI-JED and CTE performance for interleaver depths $D = 1$, $D = k$, $D = 2k$ and $D = 3k$, and CIR length $L = 3$.

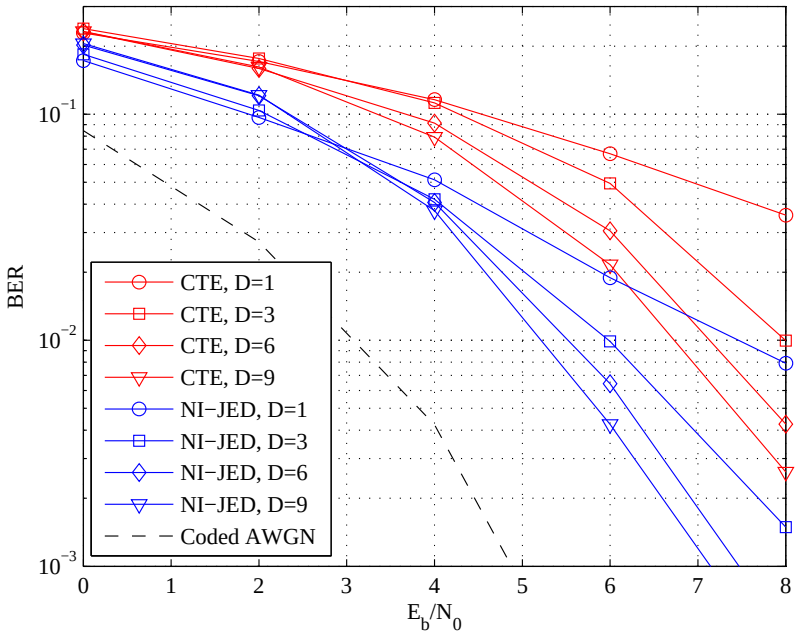


Figure 2.16: NI-JED and CTE performance for interleaver depths $D = 1$, $D = k$, $D = 2k$ and $D = 3k$, and CIR length $L = 4$.

2.6 CONCLUDING REMARKS

In this chapter optimal equalization and decoding using the MLSE (min-sum) and MAP (sum-product) algorithms were discussed. It was shown how the MLSE algorithm can be used to determine the most likely sequence of estimates, while the MAP algorithm can be used to determine optimal posterior probabilities regarding the transmitted symbols or codewords. Non-iterative joint equalization and decoding was also discussed, first assuming no interleaving and then assuming that special block interleavers were used for interleaving. As a result a general model was derived for systems transmitting convolutionally encoded BPSK modulated information through a multipath channel of length L, where the information is interleaved with an interleaver of depth $D = dk$, and where the uncoded information is encoded with a rate $R_c = 1/k$ encoder with constraint length K. The computational complexity was analyzed by counting the number of computations performed, given certain system parameters. The complexity of the NI-JED without interleaving grows exponentially with an increase in either channel memory length or encoder constraint length, while the complexity of the NI-JED with block interleaving is exponentially related to the channel memory length, encoder constraint length and the interleaver depth.

From these analyses it is clear that non-iterative joint equalization and decoding is extremely expensive in terms of the number of computations required, even for moderate channel memory lengths, encoder constraint lengths and interleaver depths. In order to achieve acceptable performance in a multipath fading environment, block interleavers with depths of multiple orders of the encoder output length are required to separate adjacent coded symbols sufficiently. Under these conditions the NI-JED becomes infeasible. Ideally a random interleaver will allow for maximum performance gains, but the NI-JED cannot be applied when interleaving is performed with a random interleaver. In *Chapter 4* and *Chapter 5* this author develops two novel turbo equalizers using superstructures, where the computational complexity is not influenced by the interleaver structure.

In the next chapter turbo equalization is discussed, where the MAP equalizer and MAP decoder iteratively exchange information, while allowing for any interleaver to be used without additional complexity. Turbo equalization is used as an alternative to optimal non-iterative joint equalization and decoding, which is not feasible because of computational complexity constraints discussed before, to the extent that approximate inference via the iterative exchange of information is the last resort. Turbo equalization is not optimal, as demonstrated in this chapter, but it is the best alternative amongst all iterative joint equalization and decoding solutions, as its constituent parts - the MAP equalizer and the MAP decoder - produce optimal posterior estimates about the respective coded and uncoded transmitted symbols.

CHAPTER 3

TURBO EQUALIZATION

In the previous chapter optimal equalization and decoding using the MLSE and MAP algorithms were discussed, after which non-iterative joint equalization and decoding with and without depth-limited block interleaving using the NI-JED was discussed. It was shown that, although the NI-JED performs optimally, the computational complexity is exponentially related to the channel memory length, encoder constraint length and interleaver depth, making it infeasible for use in practical systems.

In this chapter conventional turbo equalization is discussed. Instead of performing optimal non-iterative joint equalization and decoding on a super-trellis, a turbo equalizer uses an optimal MAP equalizer and MAP decoder pair to exchange information iteratively in order to improve BER performance. When the channel memory is long, the computational complexity of the CTE becomes prohibitive owing to its exponential relationship to the CIR length [5, 6]. Low complexity SISO equalizers are therefore often used as a replacement for the optimal MAP equalizer in a CTE, yielding LCTEs [8–11]. A number of these reduced complexity SISO equalizers are also discussed. Where applicable, permissions was obtained to use figures from the literature.

It is important to note that these LCTEs belong to a different paradigm than the DBN-TE and the HNN-TE developed in this thesis. The low complexity SISO equalizers discussed in this chapter are used to replace the optimal MAP equalizer in the equalizer/decoder pair in a CTE, whereas the DBN-TE and the HNN-TE replace both the equalizer and the decoder. The DBN-TE and the HNN-TE use superstructures to model the turbo equalization problem, and do not require the exchange of extrinsic information between subunits, since all the available information is processed in its entirety.

3.1 TURBO EQUALIZER

Turbo equalization has its origin in the *turbo principle*, first proposed in [27], where it was applied to the iterative decoding of concatenated convolutional codes. A Turbo Decoder uses two MAP decoders to decode convolutional coded concatenated codes iteratively. 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 iteration [2], achieving near Shannon limit performance in AWGN channels [27].]

Since the communication channel can be viewed as a non-binary convolutional encoder, the channel can be viewed as an inner-code while a convolutional encoder is used as an outer-code in much the same way as in serially concatenated Turbo Codes [5, 6], so that the Turbo Principle can be applied to channel equalization. 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) [5, 6]. 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 unequalized 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 improved significantly, but at the cost of additional computational complexity. [5–7, 12].

Turbo equalization becomes exceedingly complex in terms of the number of computations, because of the high computational complexity of the MAP equalizer and MAP decoder most often used in a turbo equalizer. The complexity of the MAP equalizer is linear in the data block length N, but grows exponentially with an increase in channel memory. Its complexity is therefore $O(NM^{L-1})$, where L is the CIR length, and M is the modulation alphabet size. Similarly, the complexity of the MAP decoder is linear in the data block length and exponential in the encoder constraint length K [2].

An interleaver is used as standard in a turbo equalizer, not only to mitigate the effect of burst errors by randomizing the occurrence of bit errors in a transmitted data block, but also to aid in the dispersion of the positive feedback effect, which is due to the fact that the MAP algorithm used for equalization and decoding produces outputs that are locally highly correlated [7].

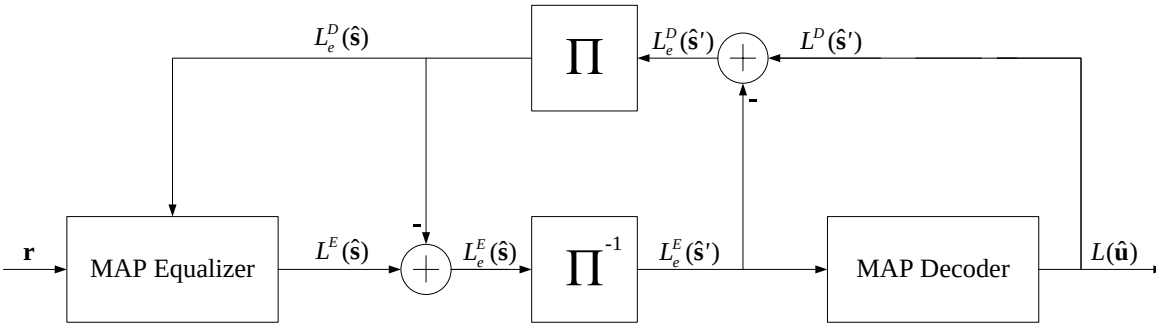


Figure 3.1: Turbo equalizer.

Fig. 3.1 shows the structure of the turbo equalizer. The MAP equalizer takes as input the ISI-corrupted received symbols $\mathbf{r}$ and the extrinsic information $L_e^D(\hat{\mathbf{s}})$ and produces a sequence of posterior transmitted symbol LLR estimates $L^E(\hat{\mathbf{s}})$ (note that $L_e^D(\hat{\mathbf{s}})$ is zero during the first iteration). The LLR estimates are produced by marginalizing over each bit in a codeword, for each codeword in the sequence. The LLR estimate $i = 1,2,...,k$ (k is the number of encoder output bits) at time instant t is calculated as

$$L^E(\hat{s_t}^{(i)}) = \log\left(\frac{P(s_t{}^{(i)} = 1)}{P(s_t{}^{(i)} = 0)}\right) \tag{3.1}$$

where

$$P(s_t^{(i)} = 1) = \sum_{j=1}^{M} P(s^{(i)} = 1 | S_{j,t}, r_t), \tag{3.2}$$

$$P(s_t^{(i)} = 0) = \sum_{j=1}^{M} P(s^{(i)} = 0 | S_{j,t}, r_t), \tag{3.3}$$

where $S_{j,t}$ denotes the jth state in the trellis at time instant t, r_t is the received codeword at time instant t, and M is the number of states in the trellis. After all $L^E(\hat{s_t}^{(i)})$ are calculated, $L^E(\hat{\mathbf{s}})$ is constructed as follows:

$$L^E(\hat{\mathbf{s}}) = \{L^E(\hat{s_1}^{(1)}), \ldots, L^E(\hat{s_1}^{(i)}), L^E(\hat{s_2}^{(1)}), \ldots, L^E(\hat{s_2}^{(i)}), \ldots, L^E(\hat{s_N}^{(1)}), \ldots, L^E(\hat{s_N}^{(i)})\}, \tag{3.4}$$

where N is the number of codewords in the sequence. Extrinsic information $L_e^E(\hat{\mathbf{s}})$ is determined by

$$L_e^E(\hat{\mathbf{s}}) = L^E(\hat{\mathbf{s}}) - L_e^D(\hat{\mathbf{s}}), \tag{3.5}$$

which is deinterleaved to produce $L_e^E(\hat{\mathbf{s}}')$, which is used as input to the MAP decoder to produce a sequence of posterior coded symbol LLR estimates $L^D(\hat{\mathbf{s}}')$. $L^D(\hat{\mathbf{s}}')$ is used together with $L_e^E(\hat{\mathbf{s}}')$ to determine the extrinsic information

$$L_e^D(\hat{\mathbf{s}}') = L^D(\hat{\mathbf{s}}') - L_e^E(\hat{\mathbf{s}}'), \tag{3.6}$$

$L_e^D(\hat{\mathbf{s}}')$ is interleaved to produce $L_e^D(\hat{\mathbf{s}})$. $L_e^D(\hat{\mathbf{s}})$ is used together with the received symbols $\mathbf{r}$ in the MAP equalizer, with $L_e^D(\hat{\mathbf{s}})$ serving to provide prior information on the received symbols. 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 [5]. After termination, the output $L(\hat{\mathbf{u}})$ of the decoder gives an estimate of the source symbols.

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 a near-optimal state in the solution space. If information is exchanged directly between the equalizer and the decoder by ignoring 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 [5–7].

3.2 REDUCED COMPLEXITY SISO EQUALIZERS

Because of the high computational complexity of the MAP equalizer, especially with long channel memory, various suboptimal low complexity alternatives have been considered as a replacement for the optimal MAP equalizer in the turbo equalizer structure [8]. These equalizers typically have linear to quadratic complexity with respect to the channel memory length, but still achieve acceptable performance. These equalizers are also modified to incorporate prior information so as to improve equalizer performance with each turbo iteration.

3.2.1 MMSE-based SISO Equalizer

The MMSE equalizer is a well-known low complexity alternative to reduce the effect of ISI on the transmitted information. The MMSE equalizer determines a set of linear filter coefficients which are then used to filter the ISI-corrupted received symbols in order to produce estimates of the transmitted

symbol sequence. This equalizer therefore belongs to the class of linear equalizers. MMSE based equalizers have per symbol complexity that is quadratic in the filter length (N) as well as in the CIR length (L) and is expressed as $O(N^2+L^2)$ [8]. Below follows a discussion of the MMSE linear equalizer (MMSE-LE) and the MMSE decision feedback equalizer (MMSE-DFE).

3.2.1.1 MMSE-LE

Assuming a system that transmits BPSK symbols through a finite impulse response (FIR) channel $\mathbf{h} = \{h_{-M_1}, h_{-M_1+1}, \ldots, h_{M_2}\}$ of length $M = M_1 + M_2 + 1$, the received symbol at time t is

$$r_t = \sum_{l=-M_1}^{M_2} h_l s_{t-l} + n_t, \tag{3.7}$$

where n_t is the tth AWGN sample which is assumed to be independent and identically distributed (i.i.d). The MMSE-LE equalizer aims to calculated a linear filter of length $N = N_1 + N_2 + 1$, $\mathbf{w}_t = \{w_{-N_1,t}, w_{-N_1+1,t}, \ldots, w_{N_2,t}\}$, in order to minimize the mean squared error (MSE) [8] $E\{|s_t - \tilde{s}_t|^2\}$ such that the estimated symbol $\tilde{s}_t$ is given by

$$\tilde{s}_t = \mathbf{w}_t^H(\mathbf{r}_t - \mathbf{p}_t^r) + p_t^s, \tag{3.8}$$

where $\mathbf{r}_t = \{r_{t+N_1}, r_{t+N_1-1}, \ldots, r_{t-N_2}\}^T$. According to [8,9], $E\{|s_t - \tilde{s}_t|^2\}$ is minimized when

$$\begin{aligned} \mathbf{w}_t &= Cov(\mathbf{r}_t, \mathbf{r}_t)^{-1} Cov(\mathbf{r}_t, s_t) \\ p_t^s &= E\{s_t\} - \mathbf{w}_t^H E\{\mathbf{r}_t\} \end{aligned}. \tag{3.9}$$

Substituting (3.9) in (3.8) gives

$$\begin{aligned} \tilde{s}_t &= Cov(\mathbf{r}_t, \mathbf{r}_t)^{-1} Cov(\mathbf{r}_t, s_t)\mathbf{r}_t + E\{s_t\} - Cov(\mathbf{r}_t, \mathbf{r}_t)^{-1} Cov(\mathbf{r}_t, s_t) E\{\mathbf{r}_t\} \\ &= E\{s_t\} + Cov(\mathbf{r}_t, \mathbf{r}_t)^{-1} Cov(\mathbf{r}_t, s_t)(\mathbf{r}_t - E\{\mathbf{r}_t\}). \end{aligned} \tag{3.10}$$

The structure of this MMSE-LE is shown in Fig. 3.2.

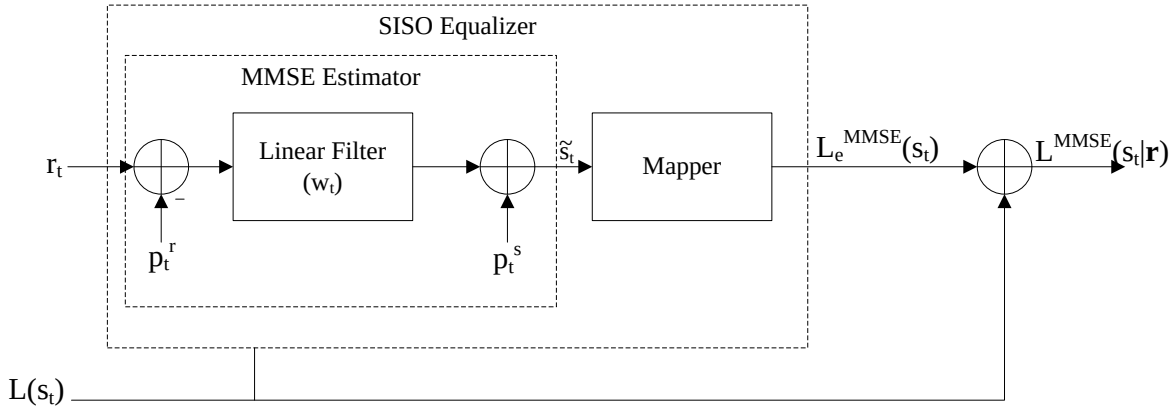


Figure 3.2: SISO MMSE-LE [8] (Figure 3).

3.2.1.1.1 Without prior information: Since no prior information is available, the transmitted information symbols are equiprobable, therefore $E\{s_t\} = 0$ and hence $E\{\mathbf{r}_t\} = 0$. Also $E\{\mathbf{r}_t\} = \mathbf{H}E\{\mathbf{s}_t\}$, where

$$\mathbf{H} = \begin{bmatrix} h_{-M_1} & h_{-M1+1} & \dots & h_{M2} & 0 & \dots & \dots & 0 \\ 0 & h_{-M_1} & h_{-M1+1} & \dots & h_{M2} & 0 & \dots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \dots & \vdots \\ \vdots & \vdots & \vdots & h_{-M_1} & h_{-M_1+1} & \dots & h_{M2} & 0 \\ 0 & \dots & \dots & 0 & h_{-M_1} & h_{-M_1+1} & \dots & h_{M2} \end{bmatrix} \tag{3.11}$$

is an $N \times (N+M-1)$ convolution matrix. $\tilde{s}_t$ is determined by

$$\tilde{s}_t = Cov(\mathbf{r}_t, \mathbf{r}_t)^{-1} Cov(\mathbf{r}_t, s_t)\mathbf{r}_t. \tag{3.12}$$

Therefore, without the availability of prior information, that is when $L(s_t) = 0$, the linear filter parameters in (3.8) are determined by

$$\begin{aligned} \mathbf{w}_t &= Cov(\mathbf{r}_t, \mathbf{r}_t)^{-1} Cov(\mathbf{r}_t, s_t) \\ p_t^s &= 0 \end{aligned}. \tag{3.13}$$

Considering the i.i.d. properties of the additive noise samples n_t, $E\{\mathbf{n}_t\} = \mathbf{0}$ and $Cov(\mathbf{n}_t, \mathbf{n}_t) = \sigma^2\mathbf{I}$, and due to uniform prior information, $E\{\mathbf{s}\} = 0$,

$$\begin{aligned} Cov(\mathbf{r}_t, \mathbf{r}_t) &= E\{\mathbf{r}_t, \mathbf{r}_t^H\} - E\{\mathbf{r}_t\}E\{\mathbf{r}_t^H\} \\ &= E\{(\mathbf{H}\mathbf{s}_t + \mathbf{n}_t)(\mathbf{H}\mathbf{s}_t + \mathbf{n}_t)^H\} - \mathbf{H}E\{\mathbf{s}_t\}\mathbf{H}^H E\{\mathbf{s}_t^H\} \\ &= E\{(\mathbf{H}\mathbf{s}_t)(\mathbf{H}\mathbf{s}_t)^H + (\mathbf{H}_t\mathbf{s}_t)\mathbf{n}_t^H + (\mathbf{H}_t\mathbf{s}_t)^H\mathbf{n}_t + \mathbf{n}_t\mathbf{n}_t^H\} - 0 \\ &= E\{(\mathbf{H}\mathbf{s}_t)(\mathbf{H}\mathbf{s}_t)^H\} + E\{(\mathbf{H}_t\mathbf{s}_t)\mathbf{n}_t^H\} + E\{(\mathbf{H}_t\mathbf{s}_t)^H\mathbf{n}_t\} + E\{\mathbf{n}_t, \mathbf{n}_t^H\} \\ &= \mathbf{H}Cov(\mathbf{s}_t, \mathbf{s}_t)\mathbf{H}^H + 0 + 0 + (Cov(\mathbf{n}_t, \mathbf{n}_t) + E\{\mathbf{n}_t\}E\{\mathbf{n}_t^H\}) \\ &= \mathbf{H}Cov(\mathbf{s}_t, \mathbf{s}_t)\mathbf{H}^H + 0 + 0 + (Cov(\mathbf{n}_t, \mathbf{n}_t) + 0) \\ &= \mathbf{H}Cov(\mathbf{s}_t, \mathbf{s}_t)\mathbf{H}^H + \sigma^2\mathbf{I}, \end{aligned} \tag{3.14}$$

and

$$Cov(\mathbf{r}_t, s_t) = \mathbf{H}Cov(\mathbf{s}_t, s_t). \tag{3.15}$$

Therefore $\mathbf{w}_t$ can be rewritten as [8]

$$\mathbf{w}_t = \left(\sigma^2\mathbf{I} + \mathbf{H}Cov(\mathbf{s}_t, \mathbf{s}_t)\mathbf{H}^H\right)^{-1}\mathbf{H}Cov(\mathbf{s}_t, s_t), \tag{3.16}$$

By further noting that $Cov(s_t, s_t') = \delta(t - t')$ due to the fact that s_t and s_t' are independent and that all transmitted symbols are equally likely,

$$\mathbf{w}_t = (\sigma^2\mathbf{I} + \mathbf{H}\mathbf{H}^H)^{-1}\mathbf{H}\mathbf{u}, \tag{3.17}$$

where $\mathbf{u}$ is the length $N + M - 1$ vector

$$\mathbf{u} = \{\mathbf{0}_{N_1+M_1} \quad 1 \quad \mathbf{0}_{N_2+M_2}\}^T.$$

3.2.1.1.2 With prior information: When prior information is available the statistics of s_t are time variant, which will affect the parameters $\mathbf{w}_t$, $E\{\mathbf{r}_t\}$ and $E\{s_t\}$ with respect to t. The MMSE estimate for the transmitted symbol at time t is obtained by (3.10), which is repeated here for convenience:

$$\tilde{s_t} = E\{s_t\} + Cov(\mathbf{r}_t, \mathbf{r}_t)^{-1}Cov(\mathbf{r}_t, s_t)(\mathbf{r}_t - E\{\mathbf{r}_t\}). \tag{3.18}$$

The time-variant statistics can be calculated as follows:

$$\begin{aligned} E\{s_t\} &= P(s_t = 1).1 + P(s_t = -1).(-1) \\ &= \frac{\exp(L(s_t))}{1+\exp(L(s_t))} + \frac{(-1)}{1+\exp(L(s_t))} \\ &= \tanh\left(\tfrac{1}{2}L(s_t)\right), \end{aligned}$$

and

$$Cov(s_t, s_{t'}) = \begin{cases} 1 - \tanh^2\left(\frac{1}{2}L(s_t)\right) & \text{if } t = t' \\ 0 & \text{otherwise,} \end{cases}$$

such that

$$Cov(\mathbf{s}_t, \mathbf{s}_t) = \begin{bmatrix} 1 - E\{s_{t+M_1+N_1}\}^2 & 0 & \dots & 0 \\ 0 & 1 - E\{s_{t+M_1+N_1-1}\}^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 - E\{s_{t-M_2-N_2}\}^2 \end{bmatrix}. \tag{3.19}$$

The covariance matrix associated with $Cov(\mathbf{s}_t, \mathbf{s}_t)$ is defined as

$$\mathbf{D}_t = \begin{bmatrix} 1 - \tanh^2\left(\frac{1}{2}L(s_{t+M_1+N_1})\right) & 0 & \dots & 0 \\ 0 & 1 - \tanh^2\left(\frac{1}{2}L(s_{t+M_1+N_1-1})\right) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 - \tanh^2\left(\frac{1}{2}L(s_{t-M_2-N_2})\right) \end{bmatrix} \tag{3.20}$$

Given the variables $E\{\mathbf{r}_t\} = \mathbf{H}E\{\mathbf{s}_t\}$, $\mathbf{D}_t = Cov(\mathbf{s}_t, \mathbf{s}_t)$ and $Cov(\mathbf{s}_t, \mathbf{s}_t) = \left(\sigma^2\mathbf{I} + \mathbf{H}Cov(\mathbf{s}_t, \mathbf{s}_t)\mathbf{H}^H\right)$, and by noting that, since $\tilde{\mathbf{s}}_\mathbf{t}$ should not depend on $L(s_t)$, $L(s_t)$ is set to 0 while computing $\tilde{s}_t$. The estimate for s_t is therefore given by [8]

$$\begin{aligned} \tilde{s}_t &= \mathbf{w}_t^H\left(\mathbf{r}_t - E\{\mathbf{r}_t\} + E\{s_t\}\mathbf{H}\mathbf{u}\right) \\ &= \mathbf{w}_t^H\left(\mathbf{r}_t - \mathbf{H}E\{\mathbf{s}_t\} + E\{s_t\}\mathbf{H}\mathbf{u}\right) \\ &= \mathbf{w}_t^H\left(\mathbf{r}_t - \mathbf{H}\mathbf{p}_t^s + p_t^s\mathbf{H}\mathbf{u}\right) \\ &= \mathbf{w}_t^H\left(\mathbf{r}_t - \mathbf{H}\mathbf{p}_t^s + \tanh\left(\tfrac{1}{2}L(s_t)\right)\mathbf{H}\mathbf{u}\right) \end{aligned}, \tag{3.21}$$

where $\mathbf{p}_t^s = \left\{\tanh\left(\frac{1}{2}L(s_{t+M_1+N_1})\right), \tanh\left(\frac{1}{2}L(s_{t+M_1+N_1-1})\right), \dots, \tanh\left(\frac{1}{2}L(s_{t-M_2-N_2})\right)\right\}^T$, and where $\mathbf{w}_t$ is determined by

$$\begin{aligned} \mathbf{w}_t &= \left(\sigma^2\mathbf{I} + \mathbf{H}\mathbf{D}_t\mathbf{H}^H + (E\{s_t\})^2(\mathbf{H}\mathbf{u})(\mathbf{H}\mathbf{u})^H\right)^{-1}\mathbf{H}\mathbf{u} \\ &= \left(\sigma^2\mathbf{I} + \mathbf{H}\mathbf{D}_t\mathbf{H}^H + \tanh^2\left(\tfrac{1}{2}L(s_t)\right)(\mathbf{H}\mathbf{u})(\mathbf{H}\mathbf{u})^H\right)^{-1}\mathbf{H}\mathbf{u} \end{aligned}. \tag{3.22}$$

For and MMSE equalizer incorporating priors, the above calculation has to be performed for every symbol s_t to be estimated, which leads to high computational complexity. Computational complexity reduction is achieved by assuming that $\mathbf{w}_t$ is time-invariant for the duration of the data block being processed, but leads to performance degradation [8]. Disregarding this complexity reduction, it will be seen in *Section 3.5* that the MMSE-LE based turbo equalizer achieves full convergence and performs optimally when the data block length is extremely long and the turbo equalizer is iterated a large number of times.

3.2.1.2 MMSE-DFE

An MMSE-DFE uses the MMSE structure in Fig. 3.3 with the addition of a feedback loop, feeding back $\tilde{s}_t$ to be used for interference cancellation in the estimation of $\tilde{s}_{t+1}$. The feedback filter $\mathbf{w}_t^b = \{w_{1,t}^b, w_{2,t}^b, \ldots, w_{N_b,t}^b\}$, where $N_b \leq N_2 + M_2$, which implies that ISI caused by $N_2 + M_2$ previous symbols can be removed. Hard decisions are made on $\tilde{s}_t$ in the feedback loop such that $\tilde{s}_t^h = sign(\tilde{s}_t^h)$.

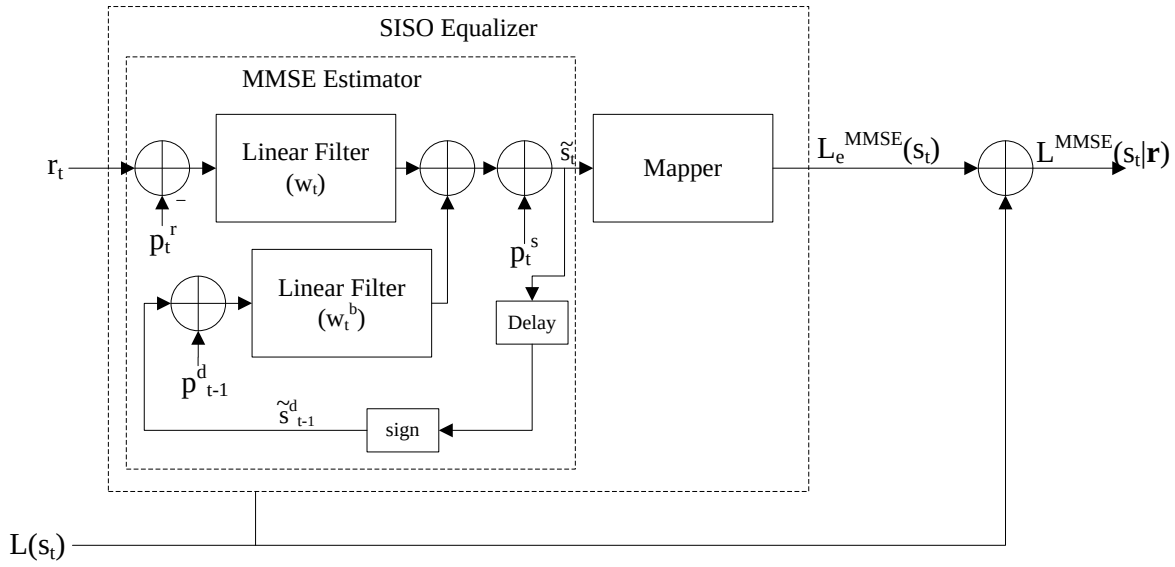


Figure 3.3: SISO MMSE-DFE Equalizer [8] (Figure 4).

The estimated output $\tilde{s}_t$ can be expressed as [8]

$$\begin{aligned} \mathbf{r}_t &= \mathbf{H}\mathbf{s}_t + \mathbf{n}_t, \\ \tilde{s}_t &= \mathbf{w}_t^H (\mathbf{r}_t - \mathbf{p}_t^r) + (\mathbf{w}_t^b)^H (\tilde{\mathbf{s}}_t^h - \mathbf{p}_t^h) + p_t^x, \end{aligned} \tag{3.23}$$

where $\mathbf{p}_t^r$ is subtracted from $\mathbf{r}_t$ before filtering, and $\tilde{s}_t^h = \{s_{t-1}^h, s_{t-2}^h, \ldots, s_{t-N_f}^h\}$ and $\mathbf{m}_t^h = \{m_{t-1}^h, m_{t-2}^h, \ldots, m_{t-N_b}^h\}$, where $m_t^h = E\{s_t^h\}$. It is clear from (3.23) that $\tilde{s}_t$ is a function of $\mathbf{r}_t$ as well as N_f previous estimates $\tilde{s}_t^h$. $\mathbf{w}_t$ and $\mathbf{w}_t^b$ can be expressed as [8]

$$\begin{aligned} \mathbf{w}_t &= \left(\sigma^2\mathbf{I} + \mathbf{H}Cov(\mathbf{s}_t, \mathbf{s}_t)\mathbf{H}^H\right)^{-1} \mathbf{H}Cov(\mathbf{s}_t, s_t) + \left(\mathbf{H}Cov(\mathbf{s}, \tilde{\mathbf{s}}_\mathbf{t}^\mathbf{h})\right)^{-1} Cov(\tilde{\mathbf{s}}_\mathbf{t}^\mathbf{h}, \mathbf{s}) \\ \mathbf{w}_t^b &= \left(Cov(\tilde{\mathbf{s}}_\mathbf{t}^\mathbf{h}, \mathbf{s})\mathbf{H}^H\right)^{-1} \mathbf{H}Cov(\mathbf{s}_t, \mathbf{s}_t) + Cov(\tilde{\mathbf{s}}_\mathbf{t}^\mathbf{h}, \tilde{s}_t^h)^{-1} Cov(\tilde{\mathbf{s}}_\mathbf{t}^\mathbf{h}, s_t) \end{aligned} . \tag{3.24}$$

Since it is assumed that $\tilde{s}_t^h$ is a perfect estimate of s_t, the statistics of $\tilde{s}_t^h$ are identical to that of s_t. Therefore $E\{\tilde{s}_t^h\} = E\{s_t\}$ and $Cov(\tilde{s}_t^h, \tilde{s}_t^h) = Cov(s_t, s_t)$. Therefore the filter parameters $\mathbf{w}_t$ and $\mathbf{w}_t^b$ can

be expressed as [8]

$$\begin{aligned} \mathbf{w}_t &= \left(\sigma^2\mathbf{I}+\mathbf{H}\left(Cov(\mathbf{s}_t,\mathbf{s}_t)-Cov(\mathbf{s}_t,\tilde{\mathbf{s}}_t^h)Cov(\tilde{\mathbf{s}}_t,\tilde{\mathbf{s}}_t)^{-1}Cov(\tilde{\mathbf{s}}_t,\mathbf{s}_t)\right)\mathbf{H}^H\right)^{-1}\mathbf{H}Cov(\mathbf{s}_t,s_t) \\ \mathbf{w}_t^b &= -Cov(\tilde{\mathbf{s}}_t,\tilde{\mathbf{s}}_t)^{-1}Cov(\tilde{\mathbf{s}}_t,\mathbf{s}_t)\mathbf{H}^H\mathbf{w}_t \end{aligned}. \quad (3.25)$$

Therefore $\tilde{s}_t$ is given by

$$\begin{aligned} \tilde{s}_t &= \mathbf{w}_t^H(\mathbf{r}_t-\mathbf{p}_t^r)-\mathbf{w}_t^H\mathbf{H}(Cov(\tilde{\mathbf{s}}_t,\tilde{\mathbf{s}}_t^h)^{-1}Cov(\tilde{\mathbf{s}}_t,\mathbf{s}_t))^H(\tilde{\mathbf{s}}_t^h-\mathbf{p}_t^h)+p_t^s \\ &= \mathbf{w}_t^H(\mathbf{r}_t-\mathbf{H}\mathbf{p}_t^s)+p_t^s \end{aligned}, \quad (3.26)$$

where $\mathbf{p}_t^s=\{p_{t+M_1+N_1}^s,\ldots,p_t^s,\tilde{s}_{t-1}^h,\ldots,\tilde{s}_{t-N_f}^h,p_{t-N_f-1}^s,\ldots,p_{t-M_2-N2}^s\}^T$

3.2.1.2.1 No prior information: Assuming that transmitted symbols s_t are equiprobable, yields

$$\begin{aligned} \tilde{s}_t &= \mathbf{w}^H(\mathbf{r}_t-\mathbf{H}\mathbf{p}_t^s), \\ \mathbf{w}_t &= (\sigma^2\mathbf{I}+\mathbf{H}\mathbf{H}^H)^{-1}\mathbf{H}\mathbf{u} \end{aligned}, \quad (3.27)$$

where $\mathbf{p}_t^s=\{\mathbf{0}_{1\times(M_1+N_1+1)},\tilde{s}_{t-1}^h,\ldots,\tilde{s}_{t-N_f}^h,p_{t-N_f-1}^s,\mathbf{0}_{1\times(M_2+N_2-N_f)}\}^T$.

3.2.1.2.2 With prior information: Assuming general priors $L(s_t)$

$$\begin{aligned} \tilde{s}_t &= \mathbf{w}^H(\mathbf{r}_t-\mathbf{H}\mathbf{p}_t^s+p_t^s\mathbf{H}\mathbf{u}), \\ \mathbf{w}_t &= (\sigma^2\mathbf{I}+\mathbf{H}\mathbf{D}_t\mathbf{H}^H)^{-1}\mathbf{H}\mathbf{u} \end{aligned}, \quad (3.28)$$

where again $\mathbf{p}_t^s=\{\mathbf{0}_{1\times(M_1+N_1+1)},\tilde{s}_{t-1}^h,\ldots,\tilde{s}_{t-N_f}^h,p_{t-N_f-1}^s,\mathbf{0}_{1\times(M_2+N_2-N_f)}\}^T$. As will be seen in *Section 3.5*, the MMSE-DFE based turbo equalizer does not achieve full convergence and performs suboptimally, even with a large number of turbo iterations.

3.2.1.3 Mapping

After calculating the estimate $\tilde{s}_t$, mapping needs to be performed in order to produce extrinsic information $L_e^{MMSE}(s_t)$ from $\tilde{s}_t$. The SISO equalizer output, for both the MMSE-LE and the MMSE-DFE is given by [8]

$$L_e^{MMSE}{}_{s_t}=\frac{2\tilde{s}_t\mu_t^{(+1)}}{\sigma_n^2}=\frac{2\tilde{s}_t}{1-(\mathbf{H}\mathbf{u})^H\mathbf{w}_n}, \quad (3.29)$$

where

$$\begin{aligned} \mu_t^{(+1)} &= \mathbf{w}^H(\mathbf{H}E\{\mathbf{s}_t|s_t=1\}-\mathbf{H}E\{\mathbf{s}_t|s_t=-1\})=\mathbf{w}_t^H\mathbf{s}, \\ \mu_t^{(-1)} &= -\mathbf{w}_t^H(\mathbf{H}\mathbf{u}), \\ \sigma_t^2 &= \mathbf{w}_t^H\left(\sigma^2\mathbf{I}+\mathbf{H}\mathbf{D}_t\mathbf{H}^H-\mathbf{H}\mathbf{u}(\mathbf{H}\mathbf{u})^H\right)\mathbf{w}_t=\mathbf{w}_t^H(\mathbf{H}\mathbf{u})-|\mu_t^{(+1)}|^2. \end{aligned}$$

3.2.2 Decision Feedback Equalizer

MMSE based equalizers have been shown to be able to exploit prior information on the transmitted symbols, therefore making them an attractive choice for use as a replacement for the MAP equalizer in a turbo equalizer setup. Although the computational complexity of the MMSE equalizers are quadratic in the number of filter coefficients (a function of channel memory), whereas the complexity of the MAP equalizer grows exponentially with an increase in channel memory, the complexity of MMSE equalizers making use of prior information becomes prohibitive when the channel memory is long, since the coefficients of the MMSE equalizer have to be determined anew for each estimated symbol. The soft feedback equalizers proposed in [10] and [11] allow for a reduction in complexity while being able to incorporate prior information by combining it with the equalizer outputs to determine more reliable estimates on the residual ISI. The per symbol complexity of these decision feedback equalizers is only linearly related to the CIR length, as well as the filter length [10, 11].

Suppose a wireless communication system, after interleaving, transmits BPSK symbols $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}$ through a frequency-selective fading channel such that the received symbol at time t is

$$r_t = \sum_{l=0}^{L-1} h_l s_{t-l} + n_t, \tag{3.30}$$

where the CIR is $\mathbf{h} = \{h_0, h_1, \ldots, h_{L-1}\}$ and n_t is an AWGN sample from the distribution $\mathcal{N}(\mu = 0, \sigma^2)$. It is assumed that prior information on the transmitted symbols is available at the receiver in the form of LLRs

$$\lambda_t^p = \log\left(\frac{P(s_t = 1)}{P(s_t = -1)}\right) \tag{3.31}$$

which, together with the received symbols $\mathbf{r} = \{r_1, r_2, \ldots, r_{N+(L-1)}\}$ are used to determine the LLR, given the a posterior probabilities, such that

$$L_t = \log\left(\frac{P(s_t = 1|\mathbf{r})}{P(s_t = -1|\mathbf{r})}\right). \tag{3.32}$$

3.2.2.1 Soft Feedback Equalizer

Fig. 3.4 shows the structure of the soft feedback equalizer proposed in [10], where the received symbols are filtered with a linear filter $\mathbf{f}$ of which the output contains residual ISI due to imperfect ISI removal. The output of the equalizer, together with the prior information, is then combined in order to mitigate the residual ISI by subtracting the result of the feedback loop from the output of $\mathbf{f}$.

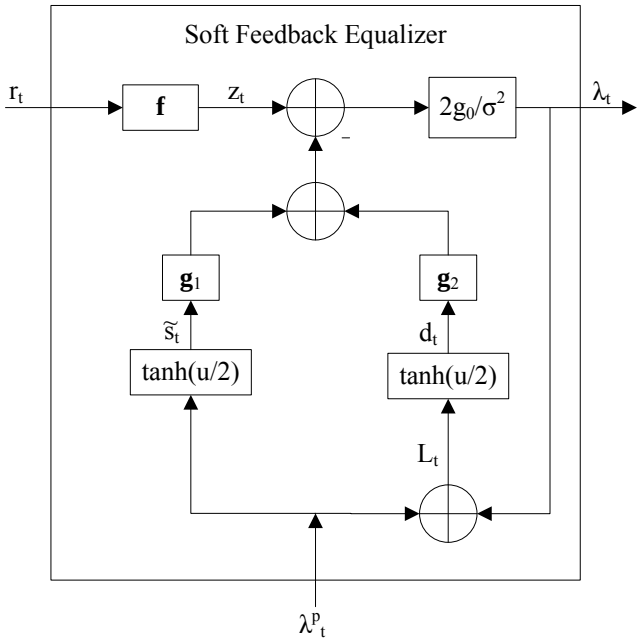


Figure 3.4: Soft feedback equalizer [10] (Figure 1).

More specifically, in the right branch of the feedback loop the output of the equalizer is additively combined with the prior information, after which the result is passed through a soft decision function ($\tanh(u/2)$) of which the result is linearly filtered by a filter $\mathbf{g}_2 = \{g_1, g_2, \ldots, g_{M_2+(L-1)}\}$ of length $M_2 + (L-1)$, whose output is an estimate of the causal part of the residual ISI. In the left feedback branch the prior information is passed through a soft decision function and filtered by a linear filter $\mathbf{g}_1 = \{g_{-M_1}, g_{-M_1+1}, \ldots, g_{-1}\}$ of length M_1, whose output is an estimate of the anticausal part of the residual ISI at the output of $\mathbf{f}$. The results of $\mathbf{g}_1$ and $\mathbf{g}_2$ are combined to cancel residual ISI in the output of $\mathbf{f} = \{f_{-M_1}, f_{-M_1+1}, \ldots, f_{M_2-1}, f_{M_2}\}$. This combination of equalizer output with prior information ensures better performance than conventional decision feedback equalizers, where only the equalizer output, but not the prior information, is used to cancel ISI.

3.2.2.1.1 Extrinsic Information: In order to compute the extrinsic information, the output of $\mathbf{f}$ after ISI cancellation is given by [10]

$$z_t = g_0 s_t + v_k, \tag{3.33}$$

where $g_0 = \sum_i h_i f_{-i}$ and $v_t = z_t - g_0 s_t$, which is the channel noise and residual ISI that has to be removed from z_t . Assuming that the elements of the sequence v_t are i.i.d. Gaussian random variables

with variance σ_v^2, the LLR of s_t is

$$\lambda_t = \frac{2g_0 z_t}{\sigma_v^2}, \tag{3.34}$$

which is the extrinsic LLR, which can be passed directly to the decoder in a turbo equalizer setup.

3.2.2.1.2 Filter coefficients: The respective filters' coefficients can be calculated by minimizing the MSE $E\{|z_t - s_t|^2\}$. If z_t is rewritten such that

$$z_t = \mathbf{f}^T \mathbf{r}_t - \mathbf{g}_1^T \tilde{\mathbf{s}}_t - \mathbf{g}_2^T \mathbf{d}_t \tag{3.35}$$

while assuming that the statistics of $s_t, \tilde{s}_t, d_t$ are identical, such that $E\{\tilde{s}_t s_{t'}\} = E\{\tilde{s}_t d_{t'}\} = E\{s_t d_{t'}\} = 0$ for $t \neq t'$ and $E\{\tilde{s}_t s_t\} = E\{|\tilde{s}_t|^2\}$ and $E\{d_t s_t\} = E\{|d_t|^2\}$, the filter coefficients can be derived by following the derivation for the MMSE-DFE as before, which gives

$$\mathbf{f} = \left(\mathbf{H}\mathbf{H}^T - \alpha_1 \mathbf{H_1}\mathbf{H_1}^T - \alpha_2 \mathbf{H_2}\mathbf{H_2}^T + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{h}_0 \tag{3.36}$$

$$\mathbf{g}_1 = \mathbf{H}_1^T \mathbf{f} \tag{3.37}$$

$$\mathbf{g}_2 = \mathbf{H}_2^T \mathbf{f} \tag{3.38}$$

where $\mathbf{H}$ is and $M \times (M + L - 1)$ channel convolution matrix

$$\mathbf{H} = \begin{bmatrix} h_0 & h_1 & \dots & h_{L-1} & 0 & 0 & \dots & 0 \\ 0 & h_0 & h_1 & \dots & h_{L-1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & h_0 & h_1 & \dots & h_{L-1} & 0 \\ 0 & \dots & 0 & 0 & h_0 & h_1 & \dots & h_{L-1} \end{bmatrix} \tag{3.39}$$

where $M = M_1 + M_2 + 1$, $\alpha_1 = E\{\tilde{s}_t s_t\}$, $\alpha_2 = E\{d_t s_t\}$, $\mathbf{H}_1 = \{\mathbf{h}_{M_1}, \dots, \mathbf{h}_{-1}\}$, $\mathbf{H}_2 = \{\mathbf{h}_1, \dots, \mathbf{h}_{M_2+(L-1)}\}$.

To complete the derivation of the soft feedback equalizer, expressions need to be determined for $\alpha_1 = E\{\tilde{s}_t s_t\}$ and $\alpha_2 = E\{d_t s_t\}$. Following [10], α_1 and α_2 can be determined by conditioning on $s_t = 1$, that is $E\{\tilde{s}_t | s_t = 1\}$ and $E\{d_t | s_t = -1\}$. Assuming that λ_k^p can be modeled as an information symbol transmitted through an AWGN channel such that $\lambda_k^p = \gamma_p s_t + w_t$, where w_t is a sample from the distribution $\mathcal{N}(\mu = 0, \sigma^2 = 2\gamma_p)$ z. Then, conditioning on $s_t = 1$, λ_k^p can be modeled as a sample from the distribution $\mathcal{N}(\mu = \gamma_p, \sigma^2 = 2\gamma_p)$. Therefore

$$\alpha_1 = \phi_1(\gamma_p), \tag{3.40}$$

where

$$\phi_1(\gamma) = E\{\tanh(u/2)\}, \tag{3.41}$$

where u is a sample take from a Gaussian distribution with γ mean and 2γ variance. γ_p can be estimated from the prior information such that

$$\tilde{\gamma_p} = \sqrt{1 + \frac{1}{N}\sum_{t=0}^{N-1} |\lambda_t^p|^2} - 1. \tag{3.42}$$

To determine α_2, one can use a Gaussian approximation for λ_t, in $L_t = \lambda_t^p + \lambda_t$, by letting $\gamma_e = \frac{2g_0^2}{\sigma^2}$ be a parameter that is twice the low signal-to-noise ratio (SNR) of the AWGN channel that produces γ_t. Using the Gaussian approximation to λ_t^p,

$$L_t = (\gamma_p + \gamma_e)s_t + \gamma_p w_t + \gamma_e v_t. \tag{3.43}$$

Conditioning on $s_t = 1$ produces L_t that is a sample from a distribution $\mathcal{N}(\mu = \gamma_p + \gamma_e, \sigma^2 = 2(\gamma_p + \gamma_e))$ and $2(\gamma_p + \gamma_e)$, and therefore

$$\alpha_2 = \phi_1(\gamma_p + \gamma_e). \tag{3.44}$$

Noting that, since $g_0 = \mathbf{f}^T\mathbf{h}_0$ and $\sigma_v^2 = g_0(1 - g_0)$,

$$\gamma_e = \frac{2\mathbf{f}^T\mathbf{h}_0}{1 - \mathbf{f}^T\mathbf{h}_0}, \tag{3.45}$$

which requires $\mathbf{f}$ to determine γ_e, in order to determine α_2, which is used to determine $\mathbf{f}$. Therefore, given an initial estimate of γ_e

$$\mathbf{f} = \left(\mathbf{H}\mathbf{H}^T - \alpha_1\mathbf{H_1}\mathbf{H_1}^T - \phi_1(\gamma_p + \gamma_e)\mathbf{H_2}\mathbf{H_2}^T + \sigma^2\mathbf{I}\right)^{-1}\mathbf{h}_0 \tag{3.46}$$

can be determined in an iterative fashion until a certain stop-criterion is met, which quickly converges to a fixed point [10].

3.2.2.2 Soft-decision Feedback Equalizer [11]

The soft-decision feedback equalizer (SDFE) is shown in Fig. 3.5. The received sequence $\mathbf{r}_t = \{r_{t-N_2}, r_{t-N_2+1}, \ldots, r_{t+N_1}\}^T$ of length $N_1 + N_2 + 1$ is filtered by a linear filter $\mathbf{f}_t = \{f_{N_2,t}, f_{N_2-1,t}, \ldots, f_{-N_1,t}\}^T$, after which residual ISI is canceled by feedback based on previous decisions. The result is transformed to LLRs (λ_t) on which soft decisions are made to produce $\mathbf{s}_t^d$, before filtering the result with a linear filter $\mathbf{b}_t = \{b_{N_3,t}, b_{N_3-1,t}, \ldots, b_{1,t}\}^T$, where $N_3 = N_2 + M - 1$, which is used to cancel the residual ISI.

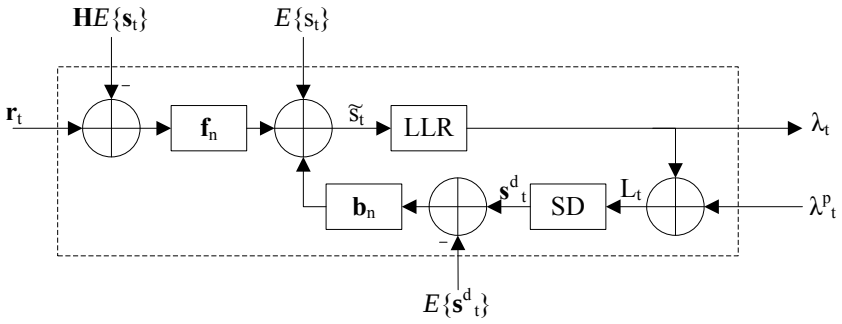


Figure 3.5: Soft-decision Feedback Equalizer [11] (Figure 2).

For the first iteration $\mathbf{H}E\{\mathbf{s}_t\} = \mathbf{0}$, $E\{\mathbf{s}_t\} = 0$ and $E\{\mathbf{s}_t^d\} = \mathbf{0}$ and the SDFE reduces to a conventional DFE [2]. During subsequent iterations $\mathbf{H}E\{\mathbf{s}_t\}$ becomes more reliable and is used to cancel residual ISI from $\mathbf{r}_t$.

The MMSE estimate of the transmitted symbol s_t is given by

$$\tilde{s}_n = \mathbf{f}_t\mathbf{r}_t + \mathbf{b}_t\mathbf{s}_t^d + d_t, \tag{3.47}$$

where $\mathbf{s}_n^d = \{s_{t-N_3}^d, s_{t-N_3+1}^d, \ldots, s_{t-1}^d\}$. It is assumed that the statistics of s_t^d and s_t are identical, therefore $Cov(s_t, s_{t'}) = 0$, $Cov(s_t, s_{t'}^d) = 0$ and $Cov(s_t^d, s_{t'}^d) = 0$ for all $t \neq t'$.

To determine the filter coefficients $\mathbf{f}_t$, $\mathbf{b}_t$ and d_t, the partial derivatives of $E\{|s_t - \tilde{s}_t|^2\}$ are taken with respect to $\mathbf{f}_t$, $\mathbf{b}_t$ and d_t [11]:

$$\frac{\partial E\{|s_t - \tilde{s}_t|^2\}}{\partial d_n} = 2E\{s_t - \mathbf{f}_t\mathbf{r}_t - \mathbf{b}_t\mathbf{s}_t^d - d_t\} \tag{3.48}$$

$$\frac{\partial E\{|s_t - \tilde{s}_t|^2\}}{\partial \mathbf{f}_n^H} = 2E\{(s_t - \mathbf{f}_t\mathbf{r}_t - \mathbf{b}_t\mathbf{s}_t^d - d_t)\mathbf{r}_t^H\} \tag{3.49}$$

$$\frac{\partial E\{|s_t - \tilde{s}_t|^2\}}{\partial \mathbf{b}_n^H} b = 2E\{(s_t - \mathbf{f}_t\mathbf{r}_t - \mathbf{b}_t\mathbf{s}_t^d - d_t)\mathbf{s}_t^{dH}\} \tag{3.50}$$

where

$$\mathbf{H} = \begin{bmatrix} h_{L-1} & h_{L-2} & \ldots & h_0 & 0 & 0 & \ldots & 0 \\ 0 & h_{L-1} & h_{L-2} & \ldots & h_0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & 0 & h_{L-1} & h_{L-2} & \ldots & h_0 & 0 \\ 0 & \ldots & 0 & 0 & h_{L-1} & h_{L-2} & \ldots & h_0 \end{bmatrix} \tag{3.51}$$

Setting (3.48) to zero

$$\begin{aligned} d_t &= 2E\{s_t - \mathbf{f}_t\mathbf{r}_t - \mathbf{b}_t\mathbf{s}_t^d\} \\ &= E\{s_t\} - E\{\mathbf{f}_t\mathbf{r}_t\} - E\{\mathbf{b}_t\mathbf{s}_t^d\} \\ &= E\{s_t\} - \mathbf{f}_t E\{\mathbf{r}_t\} - \mathbf{b}_t E\{\mathbf{s}_t^d\} \\ &= E\{s_t\} - \mathbf{f}_t\mathbf{H}E\{\mathbf{s}_t\} - \mathbf{b}_t E\{\mathbf{s}_t^d\} \end{aligned}, \tag{3.52}$$

Substituting d_t in (3.49) and (3.50) and setting both to zero gives

$$\mathbf{f}_t\mathbf{H}Cov(\mathbf{s}_t,\mathbf{s}_t^d) + \mathbf{b}_t Cov(\mathbf{s}_t^d,\mathbf{s}_t^d) = 0 \tag{3.53}$$

and

$$(\sigma^2\mathbf{I} + \mathbf{H}Cov(\mathbf{s}_t,\mathbf{s}_t)\mathbf{H}^H)\mathbf{f}_t^H = \mathbf{g}_t. \tag{3.54}$$

Therefore, solving (3.53) and (3.54) for $\mathbf{f}_t$ and $\mathbf{b}_t$, the filter coefficients are given by

$$\begin{aligned} \mathbf{f}_t^H &= \left(\sigma^2\mathbf{I} + \mathbf{H}(Cov(\mathbf{s}_t,\mathbf{s}_t) - Cov(\mathbf{s}_t,\mathbf{s}_t^d)Cov(\mathbf{s}_t^d,\mathbf{s}_t^d)^{-1}Cov(\mathbf{s}_t,\mathbf{s}_t^d)\mathbf{H}^H)\right)^{-1}\mathbf{g}_t \\ \mathbf{b}_t^H &= -Cov(\mathbf{s}_t^d,\mathbf{s}_t^d)^{-1}\left(\mathbf{H}Cov(\mathbf{s}_t,\mathbf{s}_t^d)\right)^H\mathbf{f}_t^H \\ d_t &= E\{s_t\} - \mathbf{f}_t\mathbf{H}E\{\mathbf{s}_t\} - \mathbf{b}_t E\{\mathbf{s}_t^d\} \\ \tilde{s}_t &= \mathbf{f}_t(\mathbf{r}_t - \mathbf{H}E\{\mathbf{s}_t\}) + \mathbf{b}_t(\mathbf{s}_t^d - E\{\mathbf{s}_n^d\}) + E\{s_t\} \end{aligned}, \tag{3.55}$$

where $\mathbf{g}_t = \mathbf{H}(E\{\mathbf{s}_t,\mathbf{s}_t^*\} - E\{\mathbf{s}_t\}E\{s_t^*\})$.

3.2.2.2.1 Expected value computation: A number of expected values have to be determined in order to calculate $Cov(\mathbf{s}_t,\mathbf{s}_t)$, $Cov(\mathbf{s}_t,\mathbf{s}_t^d)$ and $Cov(\mathbf{s}_t^d,\mathbf{s}_t^d)$. Following a similar approach as for the SFE in *Section 3.2.2.1*, the required expected values are determined as follows [11]:

$$E\{s_t^d\} = E\{s_t^d|s_t=1\}P(s_t=1) + E\{s_t^d|s_t=-1\}P(s_t=-1), \tag{3.56}$$

$$\begin{aligned} E\{s_t s_t^{d*}\} &= E\{s_t^d|s_t=1\}P(s_t=1) - E\{s_t^d|s_t=-1\}P(s_t=-1) \\ &= E\{s_t^d|s_t=-1\} \end{aligned}, \tag{3.57}$$

and

$$\begin{aligned} E\{s_t^d s_t^{d*}\} &= E\{(s_t^d)^2|s_t=1\}P(s_t=1) - E\{(s_t^d)^2|s_t=-1\}P(s_t=-1) \\ &= E\{(s_t^d)^2|s_t=1\} \end{aligned}, \tag{3.58}$$

where

$$s_t^d = \tanh\left(\frac{L_t}{2}\right), \tag{3.59}$$

and $L_t = \lambda_t + \lambda_t^p$ as before. $E\{s_t^d\}$, $E\{s_t s_t^{d*}\}$ and $E\{s_t^d s_t^{d*}\}$ are time-varying and needs to be computed according to the prior probability for each s_t. The same iterative method proposed in [10], as discussed in *Section 3.2.2.1*, is used to determine $E\{s_t s_t^{d*}\}$ and $E\{s_t^d s_t^{d*}\}$.

Since the filter coefficients and covariance matrices have to be determined for each $\tilde{s}_t$ to be estimated, the computational complexity is excessive. The complexity can be greatly reduced if it is assumed that these variables are time-invariant and that $\mathbf{f}$, $\mathbf{b}$, $Cov(\mathbf{s}_t, \mathbf{s}_t^d)$ and $Cov(\mathbf{s}_t^d, \mathbf{s}_t^d)$ are constant for each received data block.

3.3 COMPUTATIONAL COMPLEXITY ANALYSIS

The computational complexity of the CTE as well as the various other turbo equalizers, employing the low complexity equalizers discussed in this chapter, are determined for a given received data block, and normalized with respect to coded data block length so as to present the computational complexity per coded bit. The computational complexity is determined for each algorithm by counting the number of computations performed during the iterative equalization and decoding of one data block. It is assumed that a rate $R_c = 1/2$, constraint length $K = 5$ convolutional encoder is used to encode the information, and that $Z = 10$ iterations are used in each case. L is the CIR length and N_u and N_c are the respective uncoded and coded data block lengths.

For the MMSE based equalizers, as well as the DFE equalizers, the maximum and minimum computational complexities are given. The maximum complexities were determined by counting the number of computations performed during equalization, and approximating each maximum complexity by eliminating insignificant terms. The minimum complexities were obtained from the relevant cited literature, and were determined by simplifying the respective matrix and vector operations so as to minimize the number of computations required.

The computational complexity of the MAP equalizer is

$$CC_{MAP} = 4N_c 2^{L-1} L. \tag{3.60}$$

The minimum and maximum complexities of the MMSE-LE and MMSE-DFE equalizers are given by

$$\begin{aligned} \mathrm{CC}_{MMSE-LE}^{max} &= N_c(3N(4+2N+N^2)+4NQ(2+Q)) \\ &\approx N_c(3N^3+4NQ^2) \end{aligned}, \tag{3.61}$$

$$CC_{MMSE-LE}^{min} \approx N_c(N^2+Q^2), \tag{3.62}$$

and

$$\begin{aligned} \mathrm{CC}^{max}_{MMSE-DFE} &= N_c(5N+2N^2+N^3+6NQ+2NQ^2) \\ &\approx N_c(3N^3+2NQ^2) \end{aligned} . \tag{3.63}$$

$$CC^{min}_{MMSE-DFE} \approx N_c(N^2+Q^2), \tag{3.64}$$

where $Q = N+L-1$ and $N = N_1+N_2+1$, where $N_1 = L$ and $N_2 = L/2$. For the SFE the minimum and maximum computational complexities are given by

$$\begin{aligned} \mathrm{CC}^{max}_{SFE} &= 7M+2(K_1+K_2)+2M(K_1+K_2)+\ldots \\ &\quad \ldots+(2\mathrm{M}^2(K+K_1+K_2)+M(6+4M+M^2)) \\ &\approx 2M^2+R(2M^2(K+K1+K2)+M^3) \end{aligned} . \tag{3.65}$$

$$CC^{min}_{SFE} \approx N_c(N+M), \tag{3.66}$$

where $M = M_1+M_2+1$, $K_1 = M_1$ and $K_2 = M_2+L-1$, where M_1 and M_2 are the respective lengths of the causal and anticausal parts of the linear filter $\mathbf{f}$, and R is the number of iterations required to deterimine $\mathbf{f}$, chosen as $R = 5$ for this analysis. Finally the minimum and maximum computational complexity of the SDFE are given by

$$\begin{aligned} \mathrm{CC}^{max}_{SDFE} &= 4N+4N^2+5N^3+2N_3+R(4N+14N^3+4N^2+2N_3) \\ &\approx 5N^3+2N3+R(14N^3) \end{aligned} , \tag{3.67}$$

$$CC^{min}_{SDFE} \approx N_c(N+M). \tag{3.68}$$

where $N = N_1+N_2+1$, $N_3 = N_2+L-1$.

Fig. 3.6 shows the maximum and minimum computational complexities of the various low complexity equalizers together with that of the MAP equalizer, from which it can be seen that the low complexity equalizers, without complexity reduction, are in fact more computationally complex that the MAP equalizer for low to moderate CIR lengths. However, after optimizing matrix and vector calculations the respective complexities of the MMSE and SFE/SDFE equalizers are more favorable and only increases exponentially or linearly with an increase in CIR length.

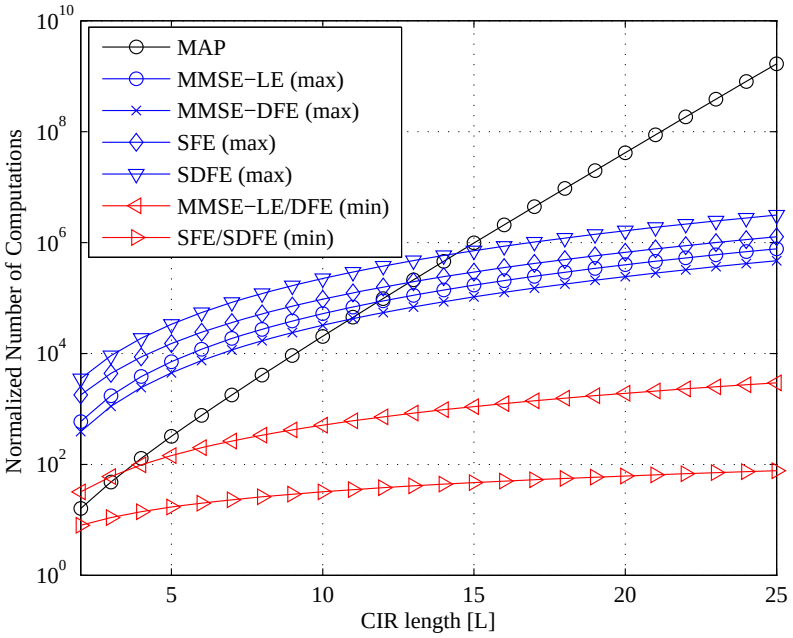


Figure 3.6: Maximum and minimum computational complexities of various low complexity equalizers compared to that of the MAP equalizer.

3.4 EXIT CHARTS

The extrinsic information transfer (EXIT) chart is a tool developed to analyze the effectiveness of information transfer between the equalizer and the decoder in a turbo equalizer. It was first proposed in [28] for parallel concatenated turbo codes, but can also be used for turbo equalization. It is useful in analyzing the convergence properties and the performance of turbo equalizers and aids in designing good codes for use in turbo equalizers.

In order to show the information transfer between the input and the output of the equalizer, an EXIT chart graphically depicts the mutual information between the extrinsic information at the input of the equalizer and the transmitted symbols **s**,

$$I(L_{in}^E, s) = \sum_{s\varepsilon\{-1,1\}} \int p(L_{in}^E|s) \log_2 \frac{p(L_{in}^E|s)}{p(L_{in}^E|s=1) + p(L_{in}^E|s=-1)} \mathrm{d}L_{in}^E, \tag{3.69}$$

and the mutual information between the extrinsic information at the output of the equalizer and the transmitted symbols **s**,

$$I(L_{out}^E, s) = \sum_{s\varepsilon\{-1,1\}} \int p(L_{out}^E|s) \log_2 \frac{p(L_{out}^E|s)}{p(L_{out}^E|s=1) + p(L_{out}^E|s=-1)} \mathrm{d}L_{out}^E, \tag{3.70}$$

where $p(L_{in}^E|s)$ and $p(L_{out}^E|s)$ are the respective probability density functions of L_{in}^E and L_{out}^E given that s was transmitted, which are given by

$$p(L_{in}^E|s=\pm 1) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(L_{in}^E \pm 1)^2}{2\sigma^2}\right) \tag{3.71}$$

and

$$p(L_{out}^E|s=\pm 1) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(L_{out}^E \pm 1)^2}{2\sigma^2}\right). \tag{3.72}$$

According to [2, 29] (3.69) and (3.70) can be simplified to

$$\begin{aligned} I(L_{in}^E, s) &= 1 - E\{\log_2\left(1+\exp(-L_{in}^E)\right)\} \\ &\approx 1 - \tfrac{1}{N}\textstyle\sum_{n=1}^{N}\log_2\left(1+\exp\left(-sL_{in}^E\right)\right) \end{aligned} \tag{3.73}$$

and

$$\begin{aligned} I(L_{out}^E, s) &= 1 - E\{\log_2\left(1+\exp(-L_{out}^E)\right)\} \\ &\approx 1 - \tfrac{1}{N}\textstyle\sum_{n=1}^{N}\log_2\left(1+\exp(-sL_{out}^E)\right) \end{aligned} \tag{3.74}$$

respectively, where N is the total number of output symbols generated. The same principles can be applied to determine the mutual information between the extrinsic information at the input/output of the decoder and the decoded transmitted symbols, $I(L_{in}^D, s)$ and $I(L_{out}^D, s)$. To determine the mutual information, the expected values in (3.69) and (3.70) must be determined, calculating the approximations in (3.73) and (3.74) by means of Monte Carlo simulation over a large number of samples (typically $N > 10^6$) for a given E_b/N_0, and plotting the results.

Fig. 3.7 [28] shows the EXIT chart for a MAP decoder at various E_b/N_0, where the encoder is a rate $1/2$ recursive systematic convolutional code with octal generator polynomials $(G_r, G) = (23, 37)$ and constraint length $K = 5$, of which the output is punctured to obtain a rate of $2/3$. The prior information is available. From Fig. 3.7 the evolution of the mutual information $I(L_{in}^E, s)$ and $I(L_{out}^E, s)$ is clear for a given E_b/N_0.

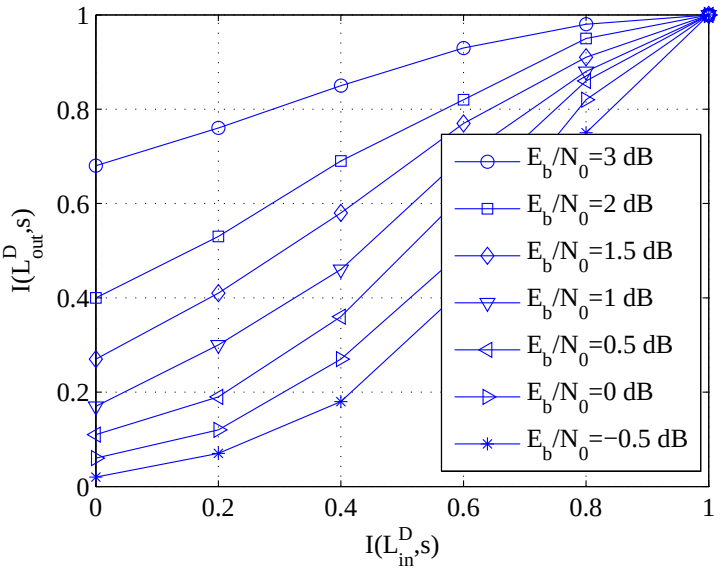


Figure 3.7: EXIT chart for a rate 2/3 recursive convolutional code [28].

3.5 SIMULATION RESULTS

The various SISO equalizers discussed in this chapter are used as a replacement for the optimal MAP equalizer in a turbo equalizer, and the resulting turbo equalizer is simulated for a number of scenarios. The corresponding EXIT charts are shown to evaluate the effectiveness of each low complexity SISO equalizer. In each case the performance of the reduced complexity turbo equalizer is compared to that of a CTE.

3.5.1 MMSE-LE and MMSE-DFE

In [12] the MMSE-LE and MMSE-DFE based turbo equalizers are simulated through a channel $\mathbf{h} = \{0.227, 0.46, 0.688, 0.46, 0.227\}$ of length $L = 5$, using a rate $1/2$ recursive systematic convolutional encoder with octal generator polynomial $G = \{7, 5\}$, where the first generator is the feedback part. $N_u = 2^{15}$ uncoded bits are encoded to $N_c = 2^{16}$ coded bits and interleaved using an S-random interleaver with $S = 0.5\sqrt{0.5N_c}$. Fig. 3.8 shows the performance of the MMSE-LE and MMSE-DFE based turbo equalizers compared to that of the CTE employing a MAP equalizer. It is clear that, although the convergence of the MMSE-LE-TE is worse than that of the CTE, the MMSE-LE-TE still achieves coded AWGN performance after 14 iterations, as if the CIR were $\mathbf{h} = 1$. The MMSE-DFE however does not perform well, even after a great number of iterations. Fig. 3.9 shows the corres-

ponding exit chart for $E_b/N_0 = 4$ dB, from which it is clear that the mutual information at the input and the output of the MMSE-LE equalizer converges to 1, while the MMSE-DFE fails to transfer much information.

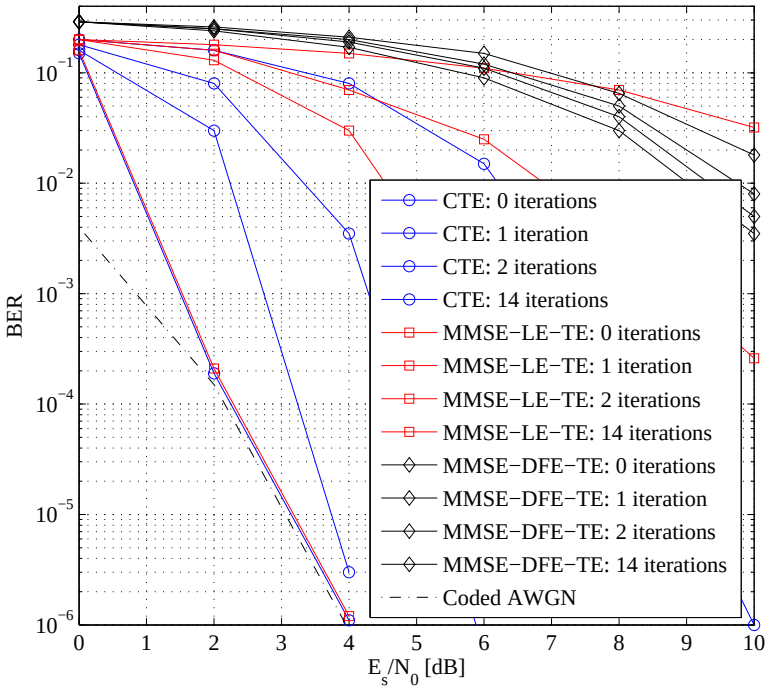


Figure 3.8: MMSE-LE and MMSE-DFE turbo equalizer BER performance [12].

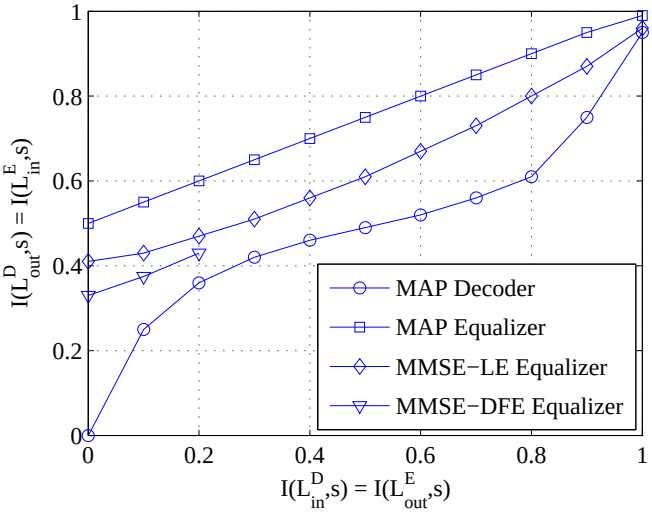


Figure 3.9: MMSE-LE and MMSE-DFE turbo equalizer EXIT chart for $E_b/N_0 = 4$ dB [12].

3.5.2 SFE

The SFE based turbo equalizer (SFE-TE) in [10] is simulated for a number of dispersive channels. A rate $1/2$ recursive convolutional encoder with generator $(1+D^2)/(1+D+D^2)$ was used before interleaving. The first channel is of length $L=5$ and has an CIR $\mathbf{h}=\{0.227, 0.46, 0.688, 0.46, 0.227\}$ for which $M_1=9$ and $M_2=5$ were used. $N_u=2^{15}$ bits were encoded before interleaving, using a random interleaver, and transmission. The SFE-TE was iterated 14 times and compared to a CTE using a MAP equalizer. Fig. 3.10 shows the performance of the SFE-TE compared to that of the CTE, from which it can be seen that both the CTE and SFE-TE achieve matched filter performance, although the SFE-TE does so only for $E_b/N_0 \leq 5$ dB. The corresponding EXIT chart is shown in Fig. 3.11.

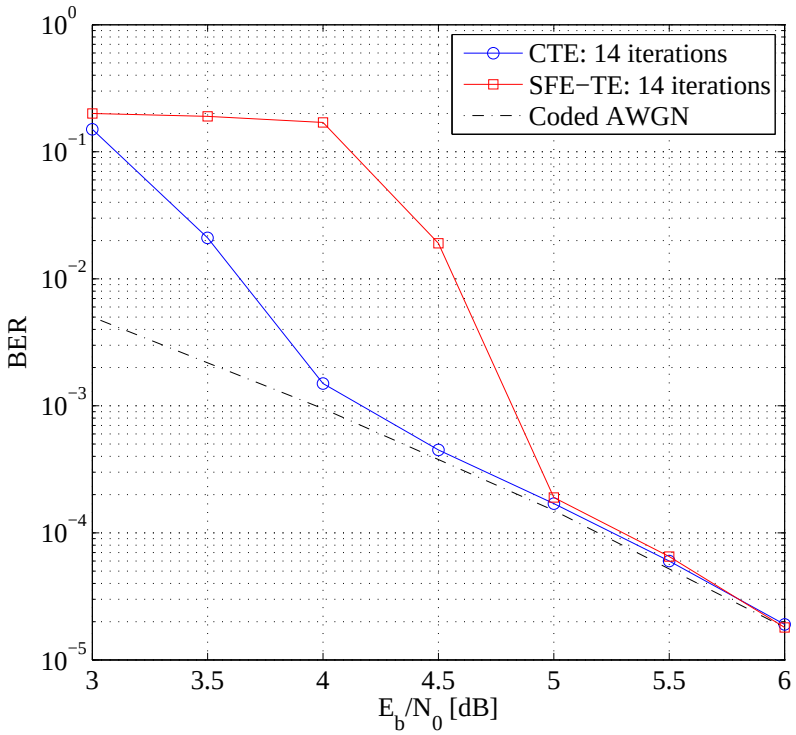


Figure 3.10: SFE-TE BER performance for $\mathbf{h}=\{0.227, 0.46, 0.688, 0.46, 0.227\}$ [10].

The second channel has a CIR of $\mathbf{h}=\{0.23, 0.42, 0.52, 0.52, 0.42, 0.23\}$ of length $L=5$, which causes maximum degradation for the ML sequence detector, through which $N_u=2^{11}$ encoded bits are transmitted after interleaving. $M_1=15$ and $M_2=10$ was used. The performance of the SFE-TE and the CTE is shown in Fig. 3.12 for various numbers of iterations. Here it is clear that the CTE greatly

outperforms the SFE-TE for this channel. The high performance gap between the CTE and SFE-TE is due to the fact that energy is almost uniformly spread throughout the channel, ie. there is no dominant CIR coefficient in **h**.

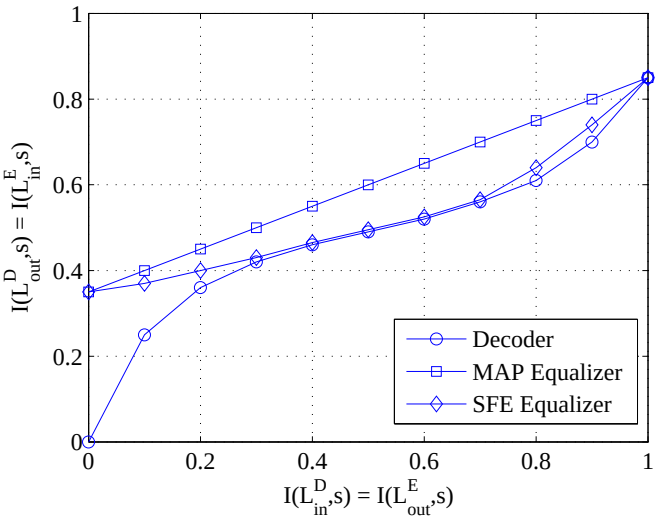


Figure 3.11: SFE-TE EXIT chart at $E_b/N_0 = 5.1$ dB for $\mathbf{h} = \{0.227, 0.46, 0.688, 0.46, 0.227\}$ [30].

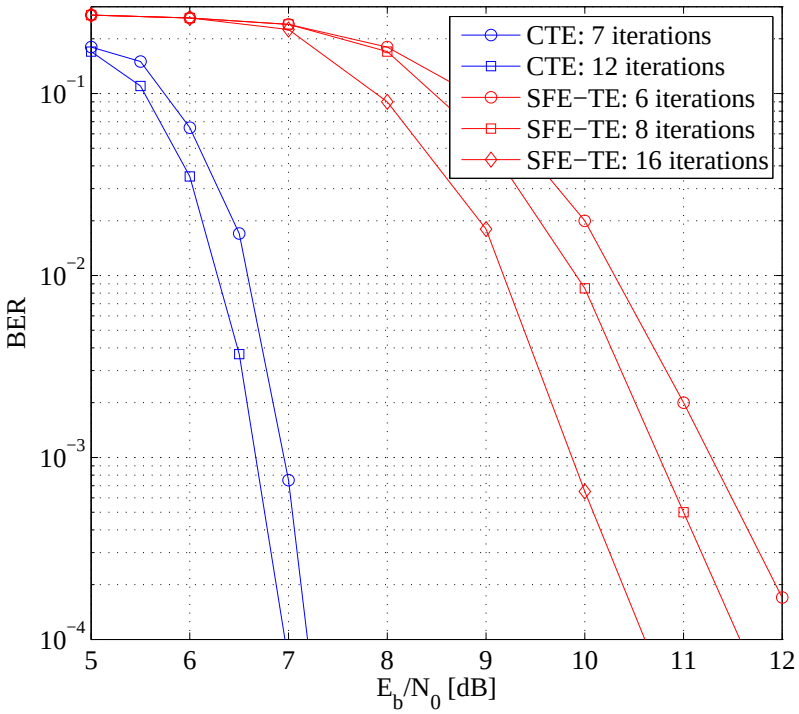


Figure 3.12: SFE-TE BER performance for $\mathbf{h} = \{0.23, 0.42, 0.52, 0.52, 0.42, 0.23\}$ [30].

3.5.3 SDFE

The soft-decision feedback turbo equalizer (SDFE-TE) proposed in [11] is simulated and compared to the SFE-TE and MMSE-TE, where the CIR is once again as in Fig. 3.12 and a rate $1/2$ convolutional encoder with octal generator polynomial $G = \{7,5\}$ is used to encode the information to $N_u = 10560$ encoded symbols, after which the encoded symbols are interleaved with a random interleaver. The filter parameters were chosen as follows: $N_1 = 9$, $N_2 = 5$, $N_3 = N_2 + M - 1$. Fig. 3.13 shows the BER performance of the SDFE-TE compared to that of the SFE-TE and the MMSE-TE for three and ten iterations respectively. Even though the performance of these turbo equalizers is comparable, it is clear from the EXIT chart in Fig. 3.14 that the SDFE-TE does not exhibit such favorable convergence properties as the MMSE-LE-TE. However, the SDFE-TE has much lower computational complexity than the MMSE-LE-TE.

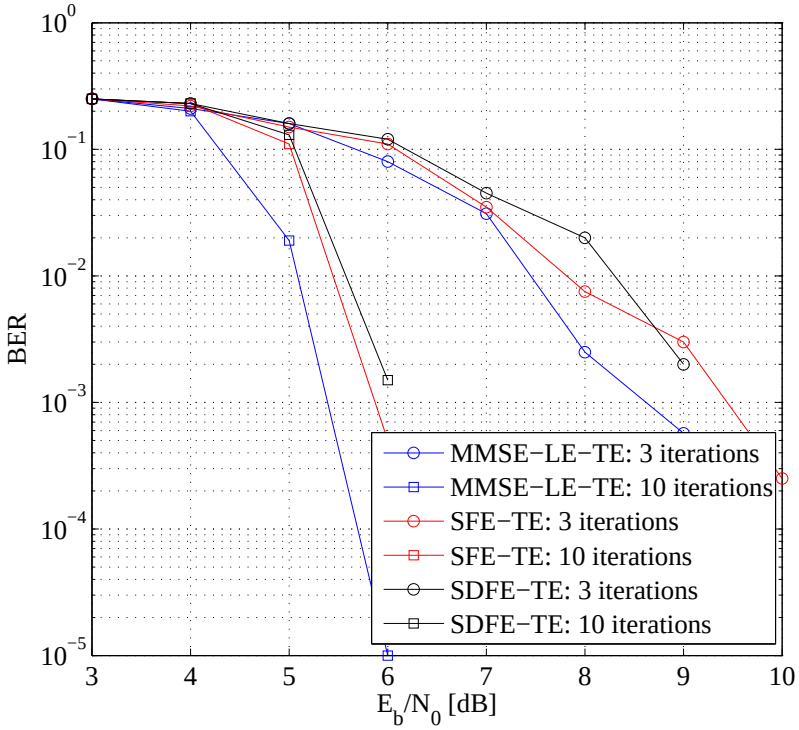


Figure 3.13: SDFE-TE, SFE-TE and MMSE-LE-TE BER performance [11].

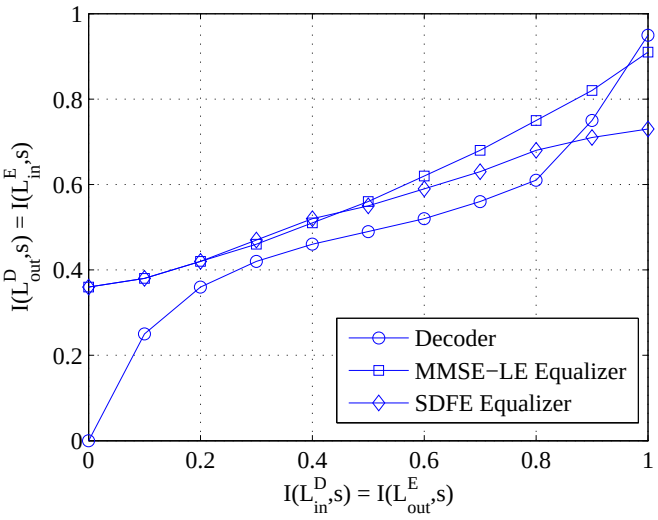


Figure 3.14: SDFE-TE EXIT chart at $E_b/N_0 = 5$ [11].

3.6 CONCLUDING REMARKS

In this chapter conventional turbo equalization was discussed and it was explained how the turbo equalizer iteratively exchanges information between the MAP equalizer and the MAP decoder in order to improve the BER performance with each iteration. It was discussed how the computational complexity of the CTE becomes excessive with moderate to long channel memory, and that the CTE is not useful in systems that transmit coded information through highly dispersive multipath channels.

A number of reduced complexity SISO equalizers were discussed, which can be used as an alternative to the optimal MAP equalizer in a turbo equalizer. Two MMSE SISO equalizers were discussed - the MMSE-LE and the MMSE-DFE - and their performance in a turbo equalizer was demonstrated via computer simulations from [12]. The MMSE equalizers have complexity that is quadratic in the channel memory length, but low-complexity alternatives do exist. Two other feedback based equalizers were discussed - the SFE and the SDFE - and it was shown that these equalizers have lower complexity than their MMSE counterparts, while maintaining acceptable performance. Their complexity is linear in the channel memory length, which makes them suitable for use in systems with extremely long memory. Simulation results from [10] and [11] demonstrated that these algorithms achieve performance that is comparable to that of the more complex MMSE-LE when used as SISO equalizers in a turbo equalizer.

In the following two chapters the author presents two unique approaches to perform iterative joint equalization and decoding by using two superstructures, namely a Bayesian belief network and a recurrent neural network. This paradigm is different from the paradigm discussed in this chapter, where information is iteratively exchanged between the equalizer and the decoder. When superstructures are used to perform iterative joint equalization and decoding, there are no restrictions on the structure of the interleaver and suboptimal equalizers are not resorted to in order to relieve the computational strain due to large channel memory. This approach combines the two joint equalization and decoding philosophies: The first was discussed in *Chapter 2*, where non-iterative joint equalization and decoding was performed on a super-trellis, while some restrictions were imposed on the structure of the interleaver. The second philosophy stems from the fact that, in order to have freedom when selecting and interleaver, the equalizer and the decoder must be separate SISO algorithms, which iteratively supply information to each other in order to improve the overall BER performance. Traditionally the use of superstructures for joint equalization and decoding had limitations with respect to the computational complexity, as discussed in *Chapter 2*, as well as the performance loss incurred by using depth-limited interleavers. As will be shown in *Chapter 4* and *Chapter 5*, these limitations are eradicated by employing two low complexity superstructures, enabling iterative joint equalization and decoding in systems transmitting randomly interleaved coded information through highly dispersive multipath channels.

CHAPTER 4

DYNAMIC BAYESIAN NETWORK TURBO EQUALIZER

During the past two decades much attention has been paid to approximate inference on factor graphs using the sum-product algorithm. Factor graphs provide a means by which complex functions can be factorized into the product of more simplistic functions, in order to infer posterior probabilistic information regarding hidden, or unknown variables [31, 32] for a particular application. Factor graphs include a myriad of graphical models used in the field of artificial intelligence and signal processing such as Bayesian networks, Markov random fields and Tanner graphs, and many well know algorithms such as the forward-backward algorithm, the turbo decoding algorithm and the Kalman filter, are all instances of factor graphs [31]. Factor graphs have also been extensively applied to the design of iterative receivers in wireless communication systems [17, 18, 33–46].

It was demonstrated in [17, 18] that turbo decoding can be performed by iteratively decoding the received codewords on a graph with cycles. Also, in [36–38] it was shown how low density parity check (LDPC) codes transmitted through multipath and partial response channels can be iteratively decoded, and therefore turbo equalized, on a factor graph. In all cases BP via the sum-product algorithm is used to calculate the approximate marginal posterior probabilities of the uncoded information symbols. Factor graphs adapted for use in wireless communication receiver algorithms, especially Turbo algorithms, usually have cycles due to a randomization effect designed to separate transmitted information temporally, ie. interleaving, to improve overall system performance. In order to reduce detection complexity, cycles have to be eliminated, and in order to remove cycles in a graph the junction tree algorithm is often used to combine nodes into supernodes, where each supernode represents a collection of original nodes [26, 31]. This combination of nodes results in an exponential

growth in the state space, and therefore the computational complexity. Apart from the very high computational complexity of this approach, it has been shown in [32, 35, 47–49] that fast, exact inference is not guaranteed on graphs with cycles.

In a wireless communication system transmitting coded information through a multipath communication channel, an interleaver is ofter used to mitigate the effect of burst errors by randomizing the occurrence of errors in a transmitted data block, as stated before. When a random or pseudo random interleaver is used, the Markov assumption, which states that the current state is only dependent on a finite history of previous states [16], is violated since the interleaver randomizes the encoded data according to some predetermined random permutation. The Markov assumption therefore fails and the turbo equalizer can no longer be modeled as a DAG to form a cycle-free decision tree.

In this chapter a low complexity near-optimal dynamic Bayesian network turbo equalizer (DBN-TE) is developed. The DBN-TE is modeled as a DAG, while relaxing the Markov assumption, by allowing weak dependencies on past and future states. Thus the DBN-TE model ensures that there is always one dominant connection between a given hidden state and its corresponding observations, while there may be several weak connections to past and future hidden states. The computational complexity of the DBN-TE is approximately quadratic in the coded data block length, exponential in decoder constraint length. Additional complexity is due to the channel memory, but is only approximately linear since it does not increase the size of the state space, but merely increases the summation terms in the sensor model. Its computational complexity is superior to that of the CTE, while being inferior to that of MMSE and DFE based LCTEs for moderate to large coded data block lengths. Results show that the performance of the DBN-TE closely matches that of a CTE in Rayleigh fading multipath channels, achieving full convergence after only a small number of iterations. Its performance in short static multipath channels is also comparable to that of the CTE and other LCTEs.

4.1 DYNAMIC BAYESIAN NETWORKS

A Bayesian network is a DAG consisting of nodes, where each node can be in a number of states [16]. Each of the possible states in turn has a corresponding probability indicating the likelihood of the node being in that state. Nodes are indicated by X_i and each node can assume M states $\{x_1, x_2, \ldots, x_M\}$. Conditional dependence between nodes is indicated by a directed link, or edge, between nodes. Fig. 4.1 shows a simple Bayesian network, where nodes X_1 and X_2 are respectively connected to nodes X_3 and X_4, which are in turn connected to node X_5. The connections between nodes in Fig. 4.1

means that X_1 has an effect on X_3, X_2 has an effect on X_4, and X_3 and X_4 have and effect on X_5. Therefore X_1 and X_2 have an indirect effect on X_5 via X_3 and X_4. Each node, except nodes X_1 and X_2 has a conditional distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.

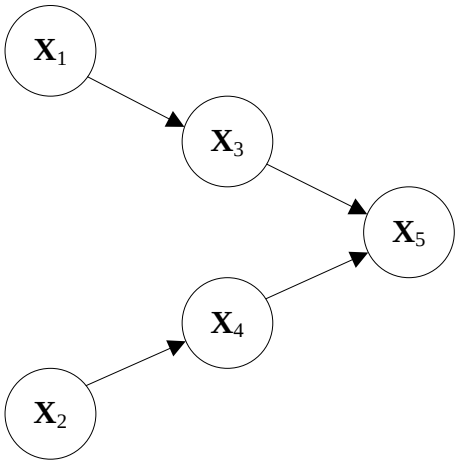


Figure 4.1: Bayesian network with five variable nodes.

4.1.1 Representing Joint Distributions

Given the conditional probability distributions for each variable, a Bayesian network is able to represent the full joint distribution for all the variables. A Bayesian network can either be viewed as a representation of the joint probability distribution of a given domain, or as an encoding of a collection of conditional independence statements [16, 50]. The joint distribution of all the variables in Fig. 4.1 can be determined as follows:

$$P(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4 = x_4, X_5 = x_5) = \prod_{i=1}^{5} P(x_i|parents(X_i)), \tag{4.1}$$

which can be written in short as

$$P(x_1, x_2, x_3, x_4, x_5) = \prod_{i=1}^{5} P(x_i|parents(X_i)), \tag{4.2}$$

Assuming that each node can assume $M = 4$ states, namely x_1, x_2, x_3 and x_4, the probability of any combination of state assignments can be calculated, assuming that the conditional probabilities are available. For instance, the probability that $X_1 = x_2$, $X_2 = x_4$, $X_3 = x_2$, $X_4 = x_1$ and $X_5 = x_3$ is calculated as

$$\begin{aligned} P(x_2,x_4,x_2,x_1,x_3) &= P(X_5=x_3|X_3=x_2)P(X_3=x_2|X_1=x_2)\dots \\ &\quad P(X_1=x_2)P(X_4=x_1|X_2=x_4)P(X_2=x_4). \end{aligned} \tag{4.3}$$

Note that X_1 and X_2 do not have conditional probabilities, since they are not affected by any other nodes.

4.1.2 Bayesian Network Construction

It is important to be able to construct the Bayesian network so as to represent the domain by means of the joint distribution. The product rule can be used to accomplish this, by rewriting the joint distribution in (4.1) for n variables [16],

$$P(x_1,\dots,x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i)), \tag{4.4}$$

in terms of the conditional probabilities such that

$$P(x_1,\dots,x_n) = P(x_n|x_{n-1},\dots,x_1)P(x_{n-1},\dots,x_1), \tag{4.5}$$

which, when each conjunctive probability is reduced to a conditional probability and smaller conjunction, factorizes to

$$\begin{aligned} P(x_1,\dots,x_n) &= P(x_n|x_{n-1},\dots,x_1)P(x_{n-1}|x_{n-2},\dots,x_1)\dots P(x_2|x_1)P(x_1) \\ &= \textstyle\prod_{i=1}^{n} P(x_i|x_{i-1},\dots,x_1) \end{aligned}. \tag{4.6}$$

From (4.6) it is clear that the Bayesian network is a correct representation of the domain if each node is conditionally independent from its predecessor. It is therefore important to select parents for each node so that this condition holds true.

4.1.3 Probabilistic Reasoning over Time

When dealing with systems that evolve over time it is necessary to be able to infer the joint distribution of any number of random variables, given all available information up to the current time instant. Keeping track of the joint distributions of the variables in past nodes, the belief state of the current node can be inferred from the observation at the current time instant. Let $\mathbf{X}_t$ be a set of unobservable state variables at time t, and let $\mathbf{E}_t$ be a set of observable evidence variables, where the observation at time t is $\mathbf{E}_t = \mathbf{e}_t$ for some set of values $\mathbf{e}_t$. In a Bayesian network that performs inference over

time, the *transition model* and the *sensor model* are key descriptors of the behavior of the system to be modeled. The transition model describes the probability of transitions between states at a given time, while the sensor model explains the process involved in producing the observed or evidence variables.

4.1.3.1 Transition Model

In order to construct a Bayesian network for temporal models, a description of the evolutionary nature of the system must be specified. This is done in the form of a transition model which describes the transitions between the different states of neighboring nodes. The transition model is given by $P(\mathbf{X}_t|\mathbf{X}_{t-1})$, which gives information regarding the probability of $\mathbf{X}_t$ being in a given state, given that $\mathbf{X}_{t-1}$ is in a certain state. In other words, the transition model specifies the probability distribution over the latest state variables, given the previous state variables. Here it is important to note that the current state is only dependent on a fixed number of previous states. This is called the *Markov assumption* [16]. In a first order Markov process, the current state only depends on the previous state ($P(\mathbf{X}_t|\mathbf{X}_{t-1})$), whereas the current state depends on two previous states in a second order Markov process ($P(\mathbf{X}_t|\mathbf{X}_{t-1},\mathbf{X}_{t-2})$). In his thesis the focus is on the former.

4.1.3.2 Sensor Model

The sensor model, or observation model, is responsible for testing the relevance of the observed evidence variables with respect to each individual state in $\mathbf{X_t}$, and producing a value for each state in $\mathbf{X_t}$ which is proportional to the probability of the state for which it is evaluated. The sensor model should faithfully model the process by which observed variables $\mathbf{E}_t$ are produced in order to best infer the probability of occurrence of each state in $\mathbf{X_t}$. The sensor model is given by $P(\mathbf{E}_t|\mathbf{X}_t)$, producing a probability for each state in $\mathbf{X_t}$, given the evidence $\mathbf{E}_t$.

4.1.3.3 Inference over Time

Fig. 4.2 shows a Bayesian network structure with four nodes $\mathbf{X}_t$, $\mathbf{X}_{t-1}$, $\mathbf{X}_{t-2}$ and $\mathbf{X}_{t-3}$, and four corresponding evidence variables $\mathbf{E}_t$, $\mathbf{E}_{t-1}$, $\mathbf{E}_{t-2}$ and $\mathbf{E}_{t-3}$. The nodes are connected by arrows, indicating the flow of time, and each node is connected to an evidence variable, with the arrow pointing towards the evidence variable since that state of the world causes the evidence variable to assume a certain value. In order to determine the probabilities of the state variables in $\mathbf{X}_t$, given all historic

information, one may write

$$P(\mathbf{X}_t, \mathbf{E}_t) = P(\mathbf{X}_0) \prod_{i=1}^{t} P(\mathbf{X}_i|\mathbf{X}_{i-1}) P(\mathbf{E}_i|\mathbf{X}_i), \tag{4.7}$$

where the first, second and third terms on the right-hand side are the initial state model, the transition model, and the sensor model respectively. Note that, in order for the system to start inferring future probability distributions, there must be an initial state model which is known *a priori*. Therefore, knowing the initial state model, the transition model and the sensor model, the joint probability distributions over all the states in $\mathbf{X}_t$ can be determined, given all past information, for any time instant t.

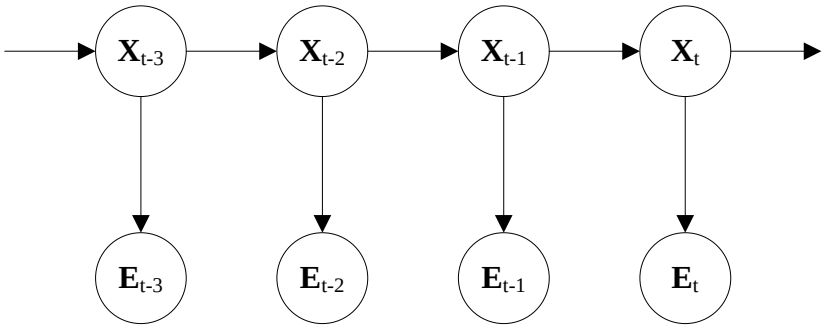


Figure 4.2: Bayesian network structure showing variable nodes and observed/evidence variables.

4.1.3.4 The Forward-backward Algorithm

The forward-backward algorithm computes the distribution over past and future states given evidence up to the present. It determines the exact MAP distribution $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $1 \leq k < t$, where $\mathbf{e}_{1:t}$ is a sequence of observed variables from time 1 to t. This is done by calculating two evidence "messages" - the forward message from 1 up to k and the backward message from t to $k+1$.

4.1.3.4.1 Forward message: The forward message computes the posterior distribution over future states, given all evidence up the current state. To compute the forward message, the current state is projected forward from time t to time $t+1$ and is then updated using the new evidence $\mathbf{e}_{t+1}$. To obtain the prediction of the next state it is necessary to condition on the current state $\mathbf{X}_t$, hence:

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{e}_{1:t}) \tag{4.8}$$

where α is a normalization constant, $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$ is obtained from the sensor model, $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)$ is the transition model and $P(\mathbf{x}_t|\mathbf{e}_{1:t})$ is the current state distribution. The forward message can be

computed recursively using (4.8).

4.1.3.4.2 Backward message: The backward message is computed in a similar fashion. It computes the posterior distribution over past states, given all future evidence up to the current state. Whereas the forward message is computed forwards from 1 to k, the backwards message is computed backwards from $k+1$ to t. Thus, the backwards message determines

$$\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \alpha \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1}) \tag{4.9}$$

where $\mathbf{P}(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})$ is obtained from the sensor model, $\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$ is the transition model and $P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})$ is the current state distribution. The backward message can be computed recursively using (4.9).

4.1.3.4.3 Forward-backward Message: Finally, by combining the forward and backward message, the posterior distribution over all states at any time instant $1 \leq k < t$ can be determined as

$$\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) = \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k). \tag{4.10}$$

Using the posterior distribution for each $\mathbf{X}_t$, the hidden variable can be estimated by marginalizing over all states $\{x_1, x_2, \ldots, x_M\}$, where M is the number of possible states.

4.2 MODELING A TURBO EQUALIZER AS A QUASI-DAG

Suppose a wireless communication system generates a sequence of source bits $\mathbf{s}$ of length N_u and $\mathbf{s}$ is encoded by a convolutional encoder of rate $R_c = 1/n$, producing a coded bit sequence $\mathbf{c}$ of length $N_c = N_u/R_c$. Now suppose that the coded bit sequence is interleaved using a random interleaver, which produces a bit sequence $\acute{\mathbf{c}}$ of length N_c, which is transmitted. The resulting symbol sequence to be transmitted is given by

$$\acute{\mathbf{c}} = \mathbf{J}\mathbf{G}^T\mathbf{s}, \tag{4.11}$$

where ^{T} denotes the transpose operation and $\mathbf{G}$ is an $N_u \times N_c$ matrix

$$\mathbf{G} = \begin{bmatrix} g_{1K} & \dots & g_{nK} & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & g_{1K} & \dots & 0 & 0 & 0 & 0 \\ g_{11} & \dots & g_{n1} & \vdots & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_{11} & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & g_{nK} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & \vdots & g_{1K} & \dots & g_{nK} \\ 0 & 0 & 0 & 0 & \dots & g_{n1} & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & g_{11} & \dots & g_{n1} \end{bmatrix} \tag{4.12}$$

representing the convolutional encoder, where

$$\mathbf{g} = \begin{bmatrix} g_{11} & \dots & g_{n1} \\ \vdots & \ddots & \vdots \\ g_{1K} & \dots & g_{nK} \end{bmatrix} \tag{4.13}$$

is the generator matrix of a rate $R_c = k/n$ ($k = 1$) convolutional encoder with constraint length K, and $\mathbf{J}$ is the $N_c \times N_c$ interleaver matrix. Now suppose the symbol sequence $\acute{\mathbf{c}}$ is transmitted over a single-carrier frequency-selective Rayleigh fading channel with a time-invariant impulse response $\mathbf{h}$ of length L, the received symbol sequence is given by

$$\mathbf{r} = \mathbf{H}\acute{\mathbf{c}} + \mathbf{n}, \tag{4.14}$$

where $\mathbf{H}$ is the $N_c \times N_c$ channel matrix with the CIR $\mathbf{h} = \{h_0, h_1, \dots, h_{L-1}\}^T$ on the diagonal such that

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & h_0 & \dots & 0 & 0 & 0 & 0 \\ h_{L-1} & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & h_{L-1} & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & h_0 & 0 & 0 \\ 0 & 0 & 0 & h_{L-1} & \dots & h_0 & 0 \\ 0 & 0 & 0 & 0 & h_{L-1} & \dots & h_0 \end{bmatrix} \tag{4.15}$$

and $\mathbf{n}$ is a complex Gaussian noise vector with $2N_c$ samples (N_c for real and N_c for imaginary) from the distribution $\mathcal{N}(0, \sigma^2)$.

Fig. 4.3 (a) shows a graphical model of the transmission model in (4.14), without noise, where it is assumed that $\mathbf{J} = \mathbf{I}$ where $\mathbf{I}$ is an $N_c \times N_c$ identity matrix (ie. no interleaving is performed) where $R_c = 1/3$ and $L = 2$. It shows that every uncoded bit s_k produces ${R_c}^{-1} = 3$ coded bits $c_{k'}$, $c_{k'+1}$ and $c_{k'+2}$, where $k' = ((k-1)/R_c) + 1$ (k runs from 1 to N_u and k' runs from 1 to N_c). Each received symbol can be expressed as

$$r_{k'} = \sum_{l=0}^{L-1} c_{k'-l} h_l, \tag{4.16}$$

where $\mathbf{h} = \{h_0, h_1\}$ is the CIR. Note that h_0 and h_1 are not shown in Fig. 4.3 (a). This joint equalization and decoding problem can be modeled as a DAG, and the forward-backward algorithm can be used to optimally estimate $\mathbf{c}$, and hence $\mathbf{s}$, with relative ease, since a one-to-one relationship exists between the observed variables $\mathbf{r}$ and the hidden variables $\mathbf{c}$. A relationship also exists between consecutive codewords (groups of n bits). Fig. 4.3 (a) also depicts the causality relationship between the hidden variables and the observed variables.

Now consider Fig. 4.3 (b). It shows a graphical model of the transmission model in (4.14), again without noise, but now $\mathbf{J}$ is a random $N_c \times N_c$ interleaver matrix and again $R_c = 1/3$ and $L = 2$. Each received symbol can be expressed as

$$r_{k'} = \sum_{l=0}^{L-1} \acute{c}_{k'-l} h_l, \tag{4.17}$$

where $\acute{c}_{k'}$ is the k'th interleaved symbol. It is clear from Fig. 4.3 (b) that there is no obvious relationship between the observed variables $\mathbf{r}$ and the hidden variables $\mathbf{c}$ and that the causality relationship in Fig. 4.3 (a) is destroyed by the randomization effect of the interleaver. Moreover, the relationship between consecutive codewords (groups of n bits) is also destroyed. This problem can therefore no longer be modeled as a DAG and exact inference is in fact impossible [32].

Deinterleaving the received sequence $\mathbf{r}$ will ensure that the one-to-one relationship between each element in $\mathbf{r}$ and $\mathbf{c}$ is restored, but only with respect to the first coefficient h_0 of the CIR $\mathbf{h}$. If h_0 is dominant and if $\mathbf{h}$ is sufficiently short, approximate inference is possible due to the negligible effect of h_1 to h_{L-1} on $\mathbf{r}$, but this is not normally the case. In a wireless communication system transmitting information through a realistic frequency-selective Rayleigh fading channel, h_0 cannot be guaranteed to be dominant and the contribution of h_1 to h_{L-1} is not negligible, and therefore this approach will fail. This has been simulated and verified by the author. Another viable alternative is to model the system as a loopy graph in order to perform approximate inference as in [49], but as stated before, exact inference is impossible [32,47–49] and full convergence is not guaranteed [51]. In loopy graphs convergence is normally achieved after many iterations.

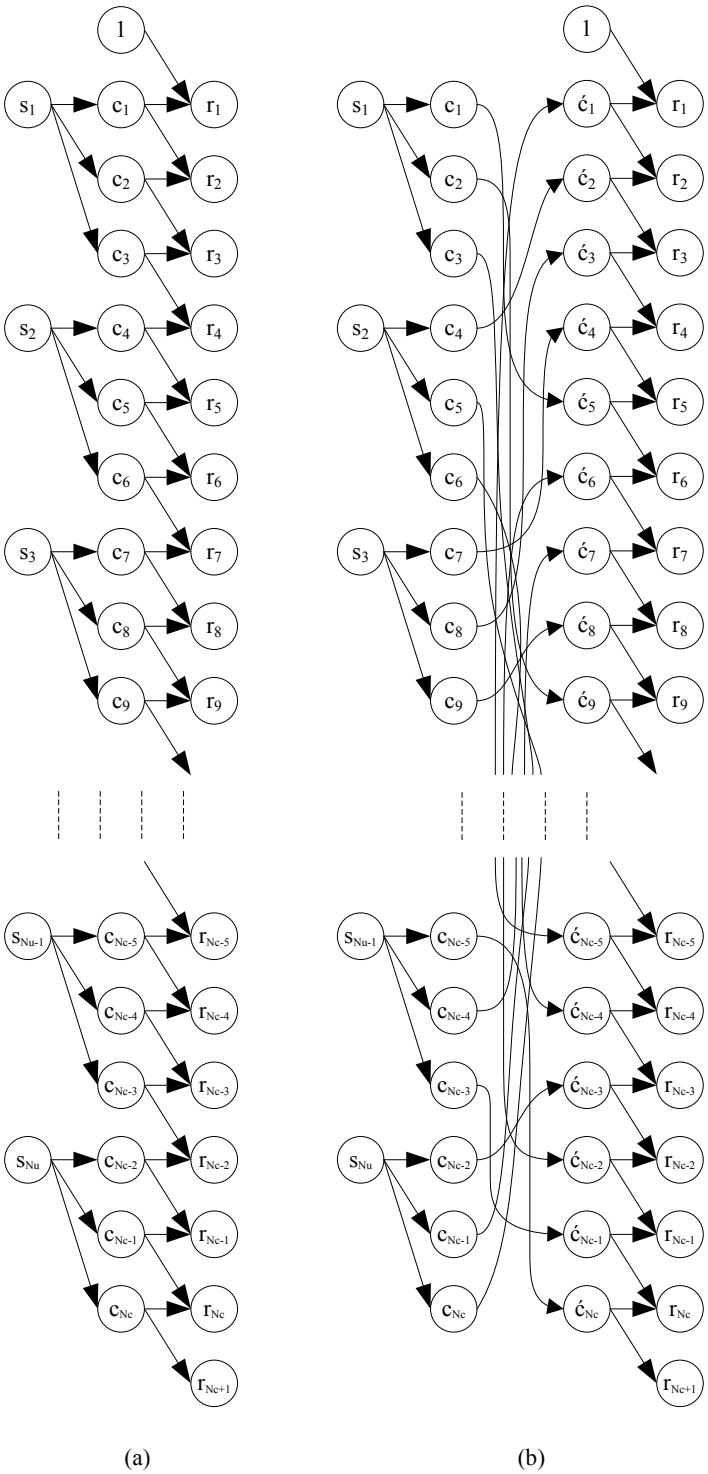


Figure 4.3: Graphical models of (4.14) without (a) and with (b) a random interleaver.

4.2.1 Prefiltering and Deinterleaving

For the DBN-TE to perform approximate inference a strong connection must exist between the observed variable and the hidden variable at time instant t, and weak connections must exist between the observed variable at time instant t and hidden variables at other time instants. The randomization effect of the interleaver must also be mitigated in order for the turbo equalizer to be modeled as a quasi-DAG so that a one-to-one relationship (dominant connection) can exist between the observed variable and the corresponding hidden variable at time instant t.

For this exposition assume that the coded symbols $\mathbf{c}$ are transmitted through a channel $\mathbf{Q} = \mathbf{HJ}$, where $\mathbf{H}$ is the channel matrix and $\mathbf{J}$ is the interleaver matrix as previously defined. Therefore (5.9) can be written as

$$\mathbf{r} = \mathbf{Qc} + \mathbf{n}. \tag{4.18}$$

4.2.1.1 Cholesky Minimum Phase Filtering

To ensure that the connection between the observed variable at time instant t and its corresponding hidden variable is dominant, and that the connections between the same observed variable and past and future neighboring hidden variables are weak, the energy must be concentrated in the first tap (h_0) of $\mathbf{h}$. This can be achieved by applying a Cholesky decomposition to the autocorrelation of the estimated channel matrix ($\mathbf{H}^H\mathbf{H}$), and then using the resulting composition to filter the received symbols $\mathbf{r}$ [52, 53].

The Cholesky decomposition is a decomposition of a matrix into the product of a lower triangular matrix and its conjugate transpose [54]. The matrix has to be Hermitian as well as positive-definite. An Hermitian matrix is a square matrix $\mathbf{A}$ containing complex values, that is equal to its own conjugate transpose. Also, an $N \times N$ Hermitian matrix $\mathbf{A}$ is said to be positive definite if $\mathbf{b}^H\mathbf{Ab}$ is real and positive for all non-zero complex vectors $\mathbf{b}$ [54].

Given an Hermitian, positive definite matrix $\mathbf{A}$, it can be factorized as

$$\mathbf{A} = \mathbf{CC}^H, \tag{4.19}$$

where $\mathbf{C}$ is the lower triangular matrix produced by the decomposition. The Cholesky decomposition is therefore the matrix analogue of taking the square root of a number.

Applying the Cholesky decomposition to the autocorrelation of the channel matrix as in [52, 53] yields

$$\mathbf{H}_{Chol} = Chol\left(\mathbf{H}^H\mathbf{H}\right) \tag{4.20}$$

where $\mathbf{H}_{Chol}$ is a lower triangular matrix resulting from the decomposition, which represents the new channel matrix with the energy concentrated in the first tap. To allow for $\mathbf{H}_{Chol}$ to be used as the new channel matrix, the received symbol sequence has to be transformed, or filtered, such that

$$\mathbf{r}_{filt} = \left(\mathbf{H}_{Chol}^H\right)^{-1}\mathbf{H}^H\mathbf{r}. \tag{4.21}$$

The transmission model in (4.18) is therefore adapted to include these modifications. Hence

$$\mathbf{r}_{filt} = \mathbf{HJc} + \mathbf{n}, \tag{4.22}$$

which is mathematically equivalent to

$$\mathbf{r} = \mathbf{H}_{Chol}\mathbf{Jc} + \mathbf{n}. \tag{4.23}$$

Therefore the effect of filtering the received symbol sequence $\mathbf{r}$ as in (4.21) is the same as transmitting the coded interleaved information through a channel $\mathbf{H}_{Chol}$ as in (4.23), where the first tap of the CIR $\mathbf{h}$ is dominant.

Fig. 4.5 and Fig. 4.4 show the CIR $\mathbf{h} = \{0.2294, 0.4588, 0.6882, 0.4588, 0.2294\}$, with and without Cholesky decomposition filtering. Fig. 4.6 and Fig. 4.7 show the frequency response of the average CIR, and Fig. 4.8 and Fig. 4.9 show the pole-zero plots of the average CIR. In Fig. 4.5 and Fig. 4.4 it can be seen that the CIR resulting from the factorized channel matrix is tapered such that the energy is concentrated in the leading CIR taps. However, the energy in the first tap is not the highest. From Fig. 4.7 and Fig. 4.6 it is clear that Cholesky decomposition filters the frequency response so as to remove the spectral nulls, which is responsible for noise enhancement and error propagation in MMSE and DFE equalizers, and finally, Fig. 4.9 and Fig. 4.8 indicate that Cholesky decomposition filtering moves the zeros from the unit circle towards the centre of the circle. However, the nulls are not close to the origin and there is still a zero on the unit circle, which will result in instability and ultimately unacceptable BER performance.

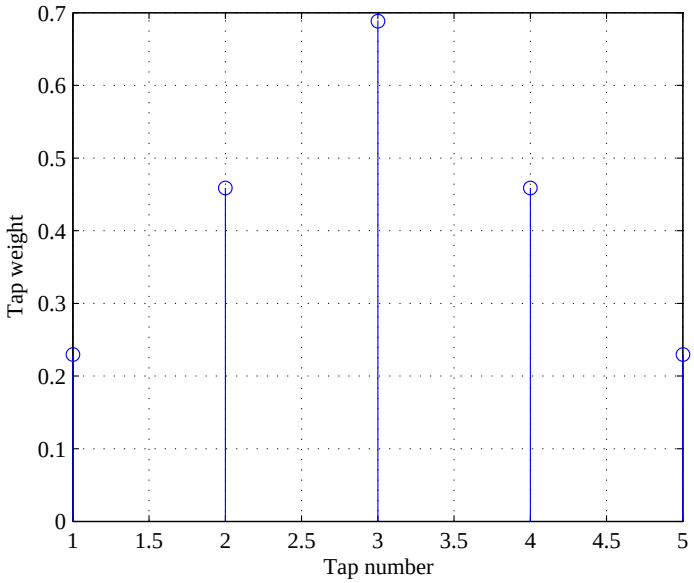


Figure 4.4: Average CIR for a system with $L = 6$ without Cholesky decomposition filtering.

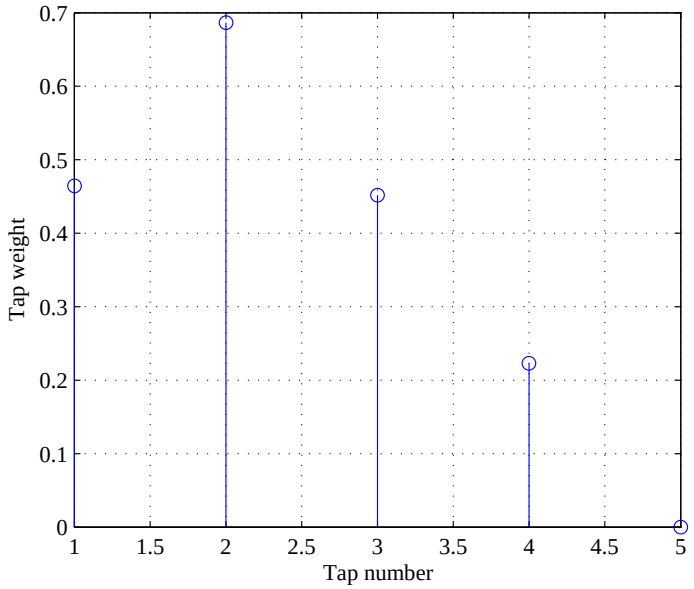


Figure 4.5: Average CIR for a system with $L = 6$ with Cholesky decomposition filtering.

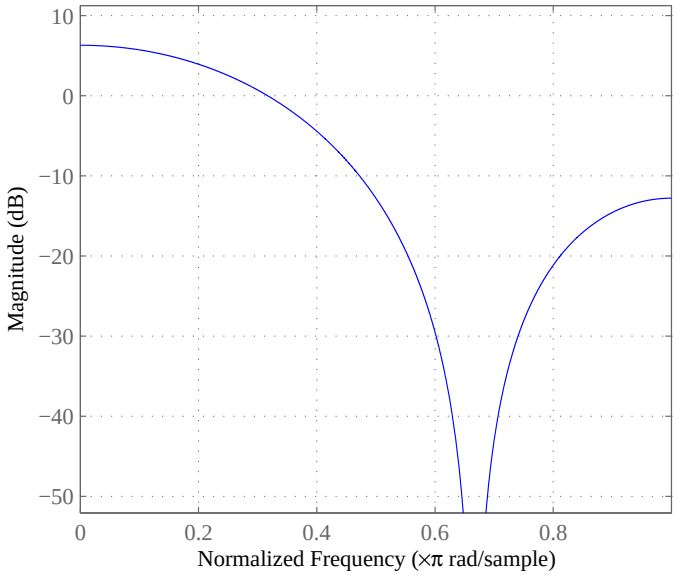


Figure 4.6: Frequency response of the average CIR for a system with $L = 6$ without Cholesky decomposition filtering.

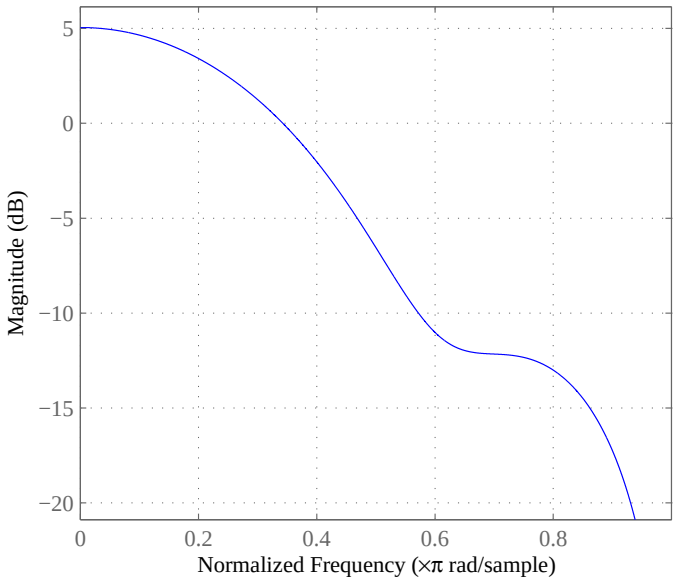


Figure 4.7: Frequency response of the average CIR for a system with $L = 6$ with Cholesky decomposition filtering.

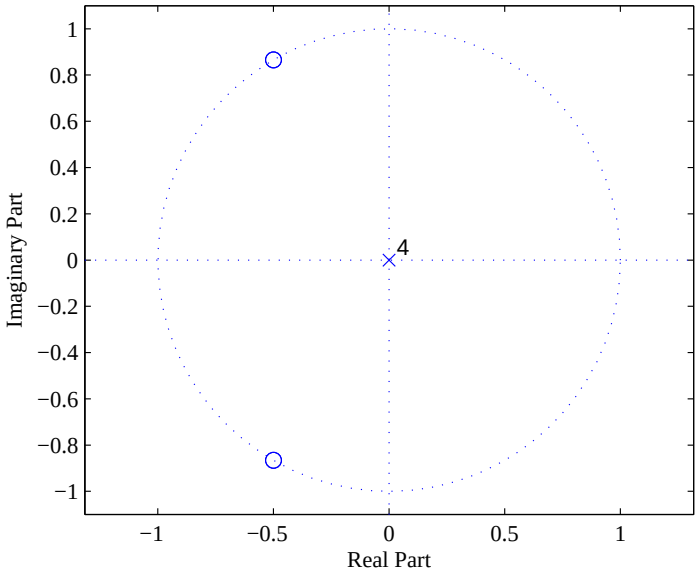


Figure 4.8: Pole-zero plot of the average CIR for a system with $L = 6$ without Cholesky decomposition filtering.

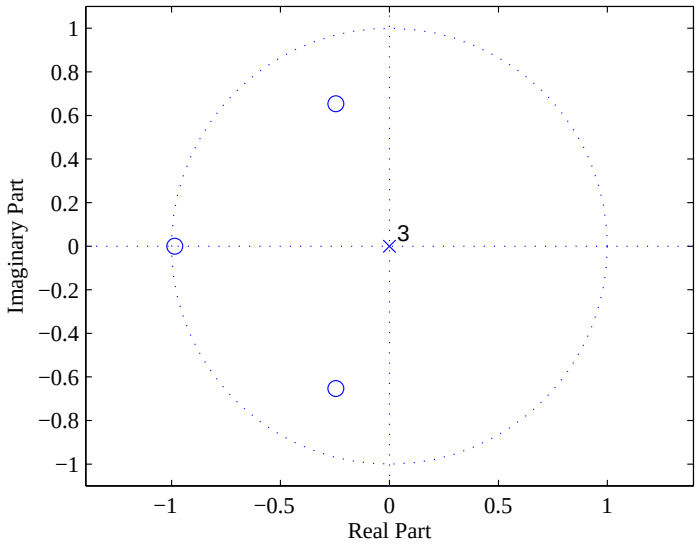


Figure 4.9: Pole-zero plot of the average CIR for a system with $L = 6$ with Cholesky decomposition filtering.

During simulation it became clear that the application of the Cholesky decomposition filtering to

render the channel in its minimum phase form has some drawbacks. Firstly it adds computational complexity that is cubic as well as quadratic in the coded data block length. The complexity of (4.20) is a maximum of $O(2N_c^3)$ and a minimum of $O(2N_c^{2.376})$, as the complexity of matrix multiplication and inversion can be reduced from $O(N^3)$ to $O(N^{2.376})$ by applying the Coppersmith-Winograd algorithm in [55]. However, (4.20) can be determined using a subset of $\mathbf{H}$, determined by the CIR length L.[1] Therefore, when using a $2L \times 2L$ subset of $\mathbf{H}$ the complexity of (4.20) reduces to a minimum of $O((2L)^{2.376})$ and a maximum of $O((2L)^3)$. Also the computational complexity of (4.21) is a minimum of $O(2N_c^{2.376} + N_c^2)$ and maximum of $O(2N_c^3 + N_c^2)$ for a fading channel, and trivial for a static channel.[2] The total computational complexity of minimum phase filtering via Cholesky decomposition is therefore a minimum of $O(2N_c^{2.376} + N_c^2 + (2L)^{2.376})$ and a maximum of $O(2N_c^3 + N_c^2 + (2L)^3)$.

Second, when simulations are performed in static multipath channels, the Cholesky decomposition sometimes fails to concentrate the energy sufficiently in the leading taps of the CIR, as is evident in Fig. 4.5, thus inhibiting the DBN-TE from achieving acceptable performance because of error propagation. However, when simulations are performed in fading multipath channels, the Cholesky decomposition sufficiently filters the channel to achieve full convergence and near-optimal performance. In order to make the DBN-TE competitive in static channels, an MMSE based minimum phase filter has to be implemented, which has complexity that is cubic in its filter length, where the filter length is usually a multiple of the unfiltered CIR. This prefilter has a computational complexity of $O(4L^3 + 4L^2 + N_c L)$, where the first and second terms are associated with the prefilter calculation, and the second term is associated with filtering the received symbol sequence with the resulting minimum phase filter.

It is however worth mentioning that computational complexity due to Cholesky decomposition filtering can also be reduced if frequency hopping is applied. During frequency hopping the channel changes owing to frequency-selective fading, and Cholesky decomposition and filtering are applied to smaller matrices. Fig. 4.10 shows the minimum and maximum numbers of calculations required for Cholesky decomposition and filtering for a system transmitting blocks of coded information of length from $N_c = 600$ to $N_c = 6000$ in intervals of 600, for a number of frequency hops from $F = 1$ to $F = 16$. From Fig. 4.10 it can be seen that frequency hopping has a favorable effect on the complexity of Cholesky decomposition and filtering, where the blue and red lines correspond to the respective

[1] A $2L \times 2L$ matrix is sufficient to determine $\mathbf{H}_{Chol}$.

[2] When the channel is static $\left(\mathbf{H}_{Chol}^H\right)^{-1}\mathbf{H}^H$ in (4.21) reduces to a positive time shift that is equal to the CIR length L. The complexity of (4.21) for a fading channel is used for the computational complexity calculations.

minimum and maximum complexities. This changes the minimum and maximum complexities to $O(2F(N_c/F)^{2.376} + F(N_c/F)^2 + (2L)^{2.376})$ (red) and $O(2F(N_c/F)^3 + F(N_c/F)^2 + (2L)^3)$ (blue) respectively.

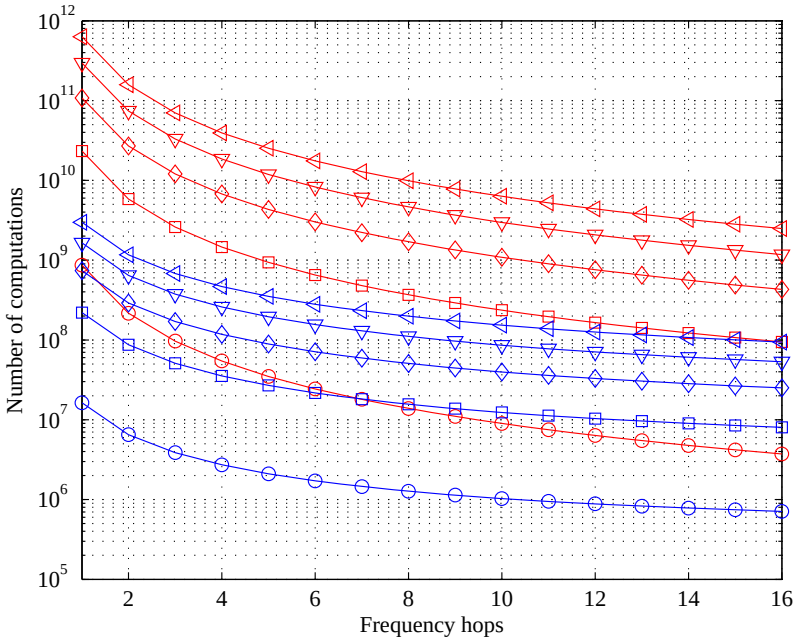


Figure 4.10: Effect of frequency hopping on the complexity of Cholesky decomposition and filtering.

4.2.1.2 MMSE-DF Minimum Phase Filtering

The MMSE-DF minimum phase filter is discussed according to [56, 57], and the MMSE-DF prefilter of the GSM simulator discussed in [57] is used in the simulation of the DBN-TE in static channels. Consider the receiver structure in Fig. 4.11 from [56], containing a matched filter, a feed-forward filter, a decision device and a feedback filter. The matched filter and an FIR filter constitute the prefilter. The output of the matched filter can be expressed in the z-domain as

$$Y(z) = C^*(z)C(z)D(z) + C^*N(z), \tag{4.24}$$

where $D(z)$ the is the transmitted symbols, $C(z)$ is the estimated CIR and $N(z)$ is the noise. Since the z-transform of the autocorrelation of $\mathbf{c}$ is nonnegative, there are no zeros in the power spectrum.

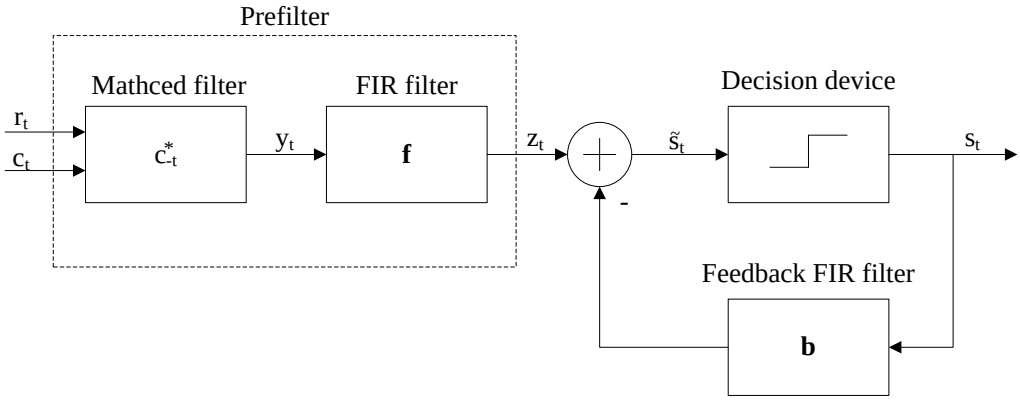


Figure 4.11: MMSE-DF prefilter [56] (Figure 7.2).

Therefore

$$C^*(z)C(z) = G^*(z)G(z)(\frac{1}{z^*}), \tag{4.25}$$

and $G(z)$ is causal and in minimum phase form, hence $G(z)(1/z^*)$ is anticausal and in maximum phase form. $G(z)$ may be found by assigning roots of $C^*(z)C(z)$ greater than 1 to the feed-forward filter as well as the remainder of the feedback filter.

4.2.1.2.1 Feedforward filter: The feed-forward filter should be chosen to cancel the precursor CIR with respect to time instant t. Therefore, the feed-forward filter ideally needs to have a z-domain representation

$$F(z) = 1/G(z)(\frac{1}{z^*}). \tag{4.26}$$

After the feedforward filter, the z-domain representation is

$$\begin{aligned} Z(z) &= F(z)G(z)G^*(\frac{1}{z^*})D(z) + F(z)C^*(z)N(z) && (4.27)\\ &= G(z)D(z) + \frac{C^*(z)}{G^*(\frac{1}{z^*})}N(z) && (4.28)\\ & && (4.29) \end{aligned}$$

in order for the prefiltered CIR $G(z)$ to be causal and in minimum phase form. However, $\frac{C^*(z)}{G^*(\frac{1}{z^*})}$ is an infinite impulse response (IIR) filter in the time domain, and therefore has to be approximated by an FIR filter. Since the FIR filter will have to be infinitely long, it is approximated by truncation, assuming that the IIR filter has an finite decaying impulse response [57].

The noise after the prefilter is given by $\frac{C^*(z)}{G^*(\frac{1}{z^*})}N(z)$ and is non-white. In order to simplify the operation

of the detector, the noise has to be whitened by a noise whitening filter.

4.2.1.2.2 Prefilter design: The prefilter is designed by using a single filter, consisting of the matched filter, the feed-forward filter and the noise whitening filter, a decision device (detector) and a feedback mechanism, as shown in Fig. 4.11. In order to maximize the energy in the leading feedback tap, the filter-detector combination is designed so that decisions made on s_t have zero delay.

The anticausal prefilter $\mathbf{f}$ is selected and is used to filter the received symbols such that

$$\mathbf{z} = \mathbf{f}^T \mathbf{r}, \tag{4.30}$$

where $\mathbf{z}$ has an impulse response $\mathbf{b}$. Therefore, z_t can be expressed as

$$z_t = \sum_{l=0}^{L} b_l s_{t-l} + n_t \tag{4.31}$$

where s_{t-l} is the $(t-l)$th transmitted symbol and n_t is a white Gaussian noise sample.

4.2.1.2.3 Choosing f and b: Past symbols are fed back using $\mathbf{b}$ in order to eliminate the ISI on the received symbols as in (4.31), assuming that the leading tap in $\mathbf{b}$ is $b_0 = 1$, and that $\tilde{s}_t$ is an estimate of s_t from a given modulation alphabet. Since the correct past symbols are fed back, the decision device can make hard decisions on s_t.

Therefore, the best choice for $\mathbf{f}$ and $\mathbf{b}$ will minimize the MMSE between s_t and $\tilde{s}_t$. That is

$$min\{E\{|s_t - \tilde{s}_t|^2\}\} = min\{E\{|\varepsilon_t|^2\}\} \tag{4.32}$$

ε_t can therefore be expressed as

$$\varepsilon_t = \mathbf{w}^T \mathbf{y} - s_t \tag{4.33}$$

where $\mathbf{w}$ and $\mathbf{y}$ are respectively given by

$$\mathbf{w} = \{f_0, f_1, \ldots, f_P, -b_1, \ldots, -b_L\}^T \tag{4.34}$$

and

$$\mathbf{y} = \{r_t, r_{t+1}, \ldots, r_{t+P}, s_{n-1}, \ldots, s_{n-L}\}^T. \tag{4.35}$$

The MMSE can therefore be written as

$$min\{E\{|\mathbf{w}^T \mathbf{y} - s_t|^2\}\} \tag{4.36}$$

and the solution to $\mathbf{w}$ is given by the Wiener-Hopf equation

$$E\{\mathbf{y}\mathbf{y}^H\}\mathbf{w}^* = E\{s_t^*\mathbf{y}\} \tag{4.37}$$

which jointly yields $\mathbf{f}$ and $\mathbf{b}$, assuming as before that $b_0 = 1$. The impulse response of the feedback filter $\mathbf{b}$ is the minimum phase equivalent impulse response (of $\mathbf{c}$) that is to be used together with the filtered signal $\mathbf{z}$ in the equalizer. The SNR will be maximized since b_0 is fixed to 1 while $E\{\|\varepsilon_t\|^2\}$ is minimized. In other words

$$SNR \propto \frac{\|\mathbf{b}\|^2}{E\{\|\varepsilon\|^2\}}. \tag{4.38}$$

4.2.1.2.4 Implementation: Referring to the Wiener-Hopf equation in (4.37), $E\{\mathbf{y}\mathbf{y}^H\}$ is written as

$$E\{\mathbf{y}\mathbf{y}^H\} = E\{\Omega\} \tag{4.39}$$

where Ω is given by

$$\Omega = \begin{bmatrix} \Omega_{11} & \Omega_{12} \\ \Omega_{21} & \Omega_{22} \end{bmatrix}, \tag{4.40}$$

and Ω_{11}, Ω_{12}, Ω_{21} and Ω_{22} are respectively given below:

$$\Omega_{11} = \begin{bmatrix} r_t r_t^* & r_t r_{t+1}^* & \dots & r_t r_{t+P}^* \\ r_{t+1} r_t^* & r_{t+1} r_{t+1}^* & \dots & r_{t+1} r_{t+P}^* \\ \dots & \dots & \dots & \dots \\ r_{t+P} r_t^* & r_{t+P} r_{t+1}^* & \dots & r_{t+P} r_{t+P}^* \end{bmatrix}, \tag{4.41}$$

$$\Omega_{12} = \begin{bmatrix} r_t s_{t-1}^* & r_t s_{t-2}^* & \dots & r_t s_{t-P-1}^* \\ r_{t+1} s_{t-1}^* & r_{t+1} s_{t-2}^* & \dots & r_{t+1} s_{t-P-1}^* \\ \dots & \dots & \dots & \dots \\ r_{t+P} s_{t-1}^* & r_{t+P} s_{t-2}^* & \dots & r_{t+P} s_{t-P-1}^* \end{bmatrix}, \tag{4.42}$$

$$\Omega_{21} = \begin{bmatrix} s_{t-1} r_t^* & s_{t-1} r_{t+1}^* & \dots & s_{t-1} r_{t+P}^* \\ s_{t-2} r_t^* & s_{t-2} r_{t+1}^* & \dots & s_{t-2} r_{t+P}^* \\ \dots & \dots & \dots & \dots \\ s_{t-L} r_t^* & s_{t-L} r_{t+1}^* & \dots & s_{t-L} r_{t+P}^* \end{bmatrix}, \tag{4.43}$$

and

$$\Omega_{21} = \begin{bmatrix} s_{t-1} s_{t-1}^* & s_{t-1} r_{t-2}^* & \dots & s_{t-1} s_{t-P-1}^* \\ s_{t-2} s_{t-1}^* & s_{t-2} r_{t-2}^* & \dots & s_{t-2} s_{t-P-1}^* \\ \dots & \dots & \dots & \dots \\ s_{t-L} s_{t-1}^* & s_{t-L} r_{t-2}^* & \dots & s_{t-L} s_{t-P-1}^* \end{bmatrix}, \tag{4.44}$$

where P is the number of feed-forward filter coefficients. Also, $E\{s_t^*\mathbf{y}\}$ in (4.37) is given by

$$E\{s_t^*\mathbf{y}\} = E\{s_t^* r_t, s_t^* r_{t+1}, \ldots, s_t^* r_{t+P}, s_t^* s_{t-1}^*, s_t^* s_{t-1}^*, \ldots, s_t^* s_{t-1}^*\}^T. \tag{4.45}$$

The feed-forward and feedback filters are determined by

$$\mathbf{w}^* = \Omega^{-1}\mathbf{h} \tag{4.46}$$

where $\mathbf{h}$ contains the estimated CIR $\mathbf{c}$ padded $P-L$ with zeros

$$\mathbf{f} = \{w_0, w_1, \ldots, w_{P-1}\} \tag{4.47}$$

and

$$\mathbf{b} = \{-w_P, -w_{P+1}, \ldots, -w_{P+L}\} \tag{4.48}$$

as given by (4.34). Finally, the received symbol sequence is filtered using the feedback filter as in (4.30) such that $\mathbf{z} = \mathbf{f}^T\mathbf{r}$. The minimum phase CIR $\mathbf{b}$ and the filtered received symbol sequence $\mathbf{z}$ are now used in the equalizer to determine the most likely transmitted symbols.

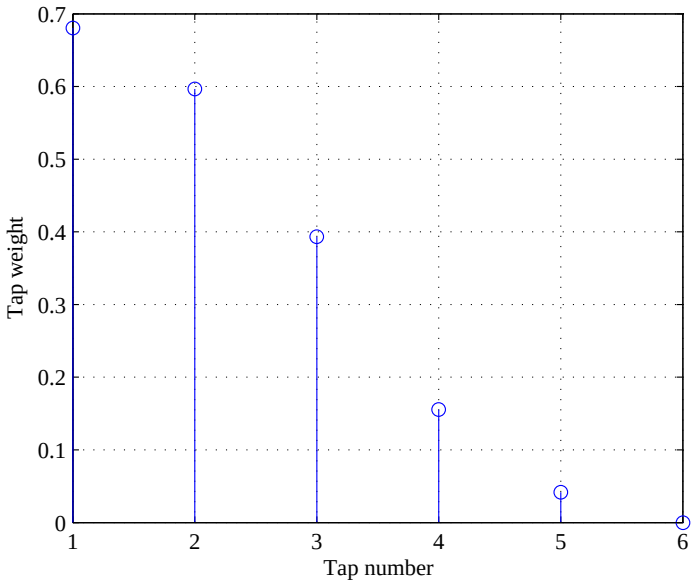


Figure 4.12: Average CIR for a system with $L = 6$ with MMSE-DF filtering.

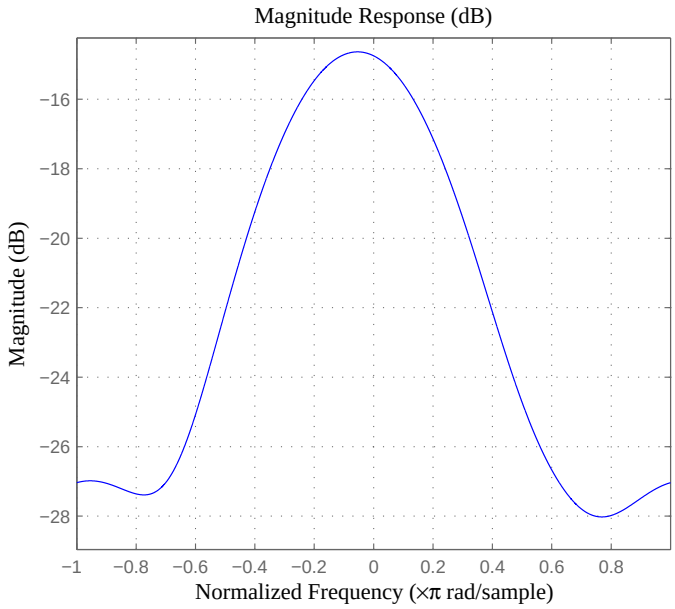


Figure 4.13: Frequency response of the average CIR for a system with MMSE-DF filtering.

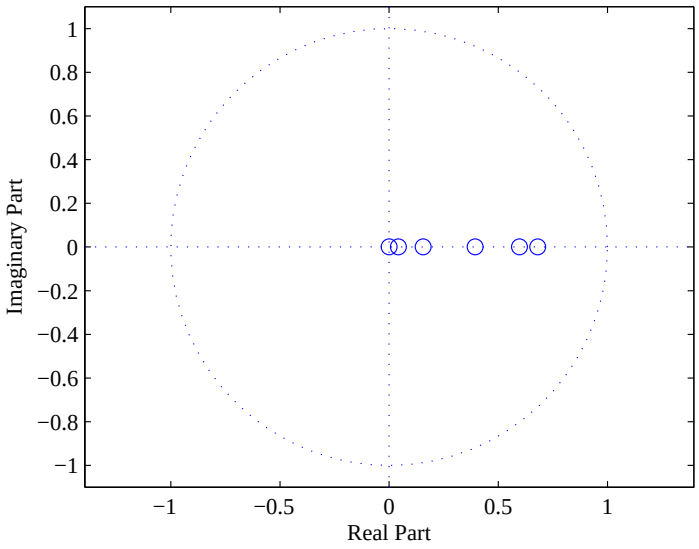


Figure 4.14: Pole-zero plot of the average CIR for a system with MMSE-DF filtering.

This concludes the derivation and discussion of the MMSE-DF prefilter used to filter the non-minimum phase CIR and the received symbols, before the DBN-TE is applied to jointly equalize and decode the received symbol sequence. Fig. 4.12 through to Fig. 4.14 show the prefiltered CIR,

its pole-zero plot and its frequency response for the same scenario as for Cholesky decomposition filtering, from which it is clear that the energy is concentrated in the leading CIR tap, that the zeros are well inside the unit circle, and that the spectral nulls have been removed.

4.2.1.3 Interleaver Mitigation

In order to model the turbo equalization problem as a DAG, or a quasi-DAG due to weak connections to past and future states, the randomization effect of the random interleaver must be mitigated. Fig. 4.15 and Fig. 4.16 show $|\mathbf{Q}|$ for systems with CIR lengths of $L = 1$ and $L = 3$ respectively, for a hypothetical system with parameters $N_u = 50$, $N_c = 150$, $R_c = 1/3$. It should be clear that any sequence $\mathbf{c}$ that is transmitted through a channel $\mathbf{Q}$, as described in (4.18), will be subject to randomization.

In order to reverse the randomization effect of the interleaver while ensuring a dominant connection between the observed variables and their corresponding hidden variables, one leans on the fact that the product of the transpose of any random interleaver matrix with the interleaver matrix itself, will produce an identity matrix. That is

$$\mathbf{J}^T\mathbf{J} = \mathbf{I}, \tag{4.49}$$

thus removing the randomness of the interleaver. To reverse the effect of the interleaver in the transmission model in (4.18) is not straightforward, since the coded information is interleaved ($\mathbf{J}$) and then transformed by the channel ($\mathbf{H}$), since $\mathbf{Q} = \mathbf{HJ}$. In order to reverse the effect of the interleaver on the received symbol sequence, the autocorrelation of the combined interleaver/channel matrix $\mathbf{Q}$ is determined, and used to filter the received symbol sequence. To mitigate the effect of the interleaver, the following transformation is applied to $\mathbf{r}$:

$$\mathbf{Q}^H\mathbf{r} = \mathbf{Q}^H\mathbf{Q}\mathbf{c} + \mathbf{Q}^H\mathbf{n}, \tag{4.50}$$

which is equivalent to transmitting the coded symbol sequence $\mathbf{c}$ through a channel $\mathbf{U}$, where

$$\mathbf{U} = \mathbf{Q}^H\mathbf{Q}, \tag{4.51}$$

so that

$$\mathbf{Q}^H\mathbf{r} = \mathbf{U}\mathbf{c} + \mathbf{Q}^H\mathbf{n}. \tag{4.52}$$

Fig. 4.17 and Fig. 4.18 show $|\mathbf{U}|$ for systems with CIR lengths of $L = 1$ and $L = 3$ respectively. It is clear that this transformation mitigates the randomness exhibited in $\mathbf{Q}$, since the new "chan-

nel" $\mathbf{U}$ is diagonally dominant. The one-to-one relationship between the observed variables and the corresponding hidden variables is therefore restored. The computational complexity of interleaver mitagation is quadratic in the coded data block length ($O(N_c^2)$) because of the filtering of the received symbol sequence ($\mathbf{Q}^H\mathbf{r}$ in (4.52)), and cubic in the coded data block length ($O(N_c^3)$) because of the autocorrelation of the interleaver/channel matrix ($\mathbf{Q}^H\mathbf{Q}$ in (4.51)). The computational complexity of interleaver mitigation is therefore a maximum of $O(N_c^3+N_c^2)$ and a minimum of $O(N_c^{2.376}+N_c^2)$. Unfortunately no complexity reduction is possible by making use of frequency hopping.

Therefore, by applying a Cholesky decomposition to the correlation of the channel matrix and filtering $\mathbf{r}$ accordingly, before performing the transformation in (4.52) to mitigate the effect of the interleaver, all the conditions are met to model the turbo equalizer as a quasi-DAG with dominant connections between the observed variables and their corresponding hidden variables.[3] Cholesky decomposition and filtering ensures that a dominant connection exists between the observed variable and the hidden variable at time instant t, and that weak connections exist between the observed variable at time instant t and the hidden variable at other time instants, while the transformation in (4.52) mitigates the randomization effect of the interleaver so that a one-to-one relationship may exist between each observed variable and its corresponding hidden variable. The minimum computational complexity of the prefiltering phase (minimum phase filtering and interleaver mitigation) is therefore $O(N_c^{2.376}+N_c^2+N_cL+4L^2)$, and the maximum complexity is $O(5N_c^3+2N_c^2)$. These maximum and minimum complexities will be used in the calculation of the overall complexity of the DBN-TE.

[3] As stated before, a minimum phase prefilter can also be used to produce a minimum phase equivalent CIR.

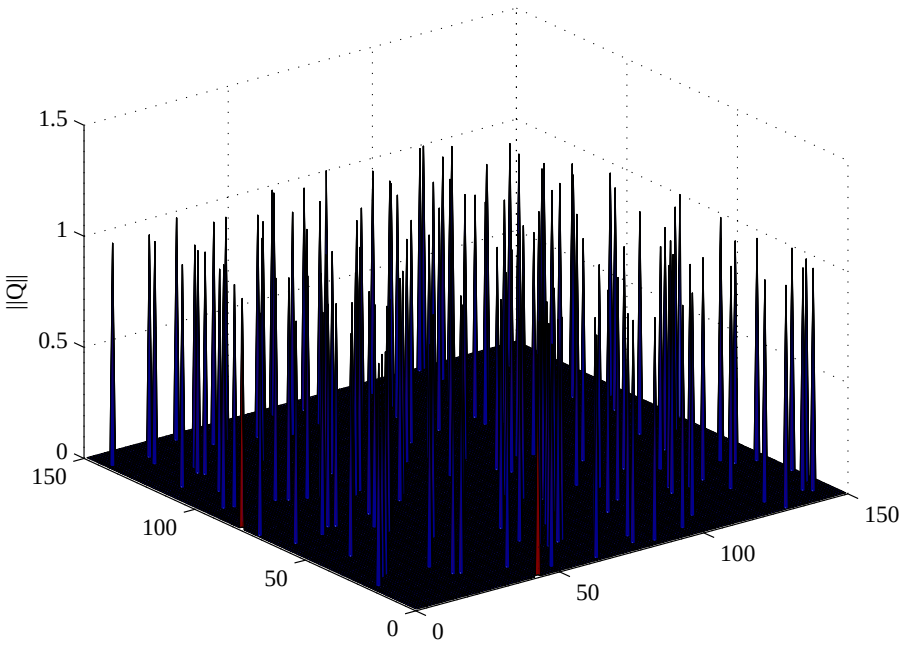


Figure 4.15: $|\mathbf{Q}|$ for a system with $L = 1$ CIR coefficients.

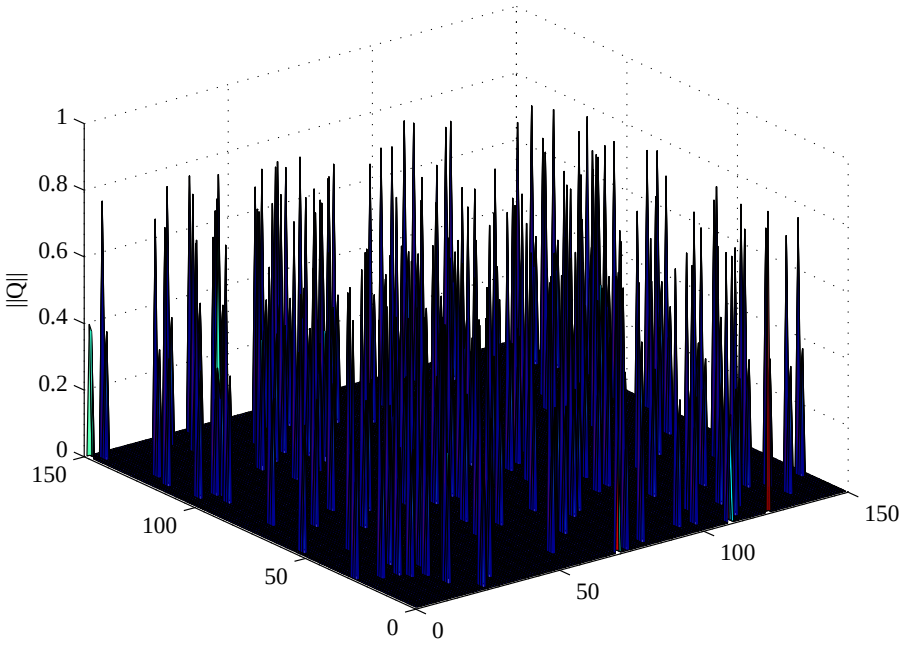


Figure 4.16: $|\mathbf{Q}|$ for a system with $L = 3$ CIR coefficients.

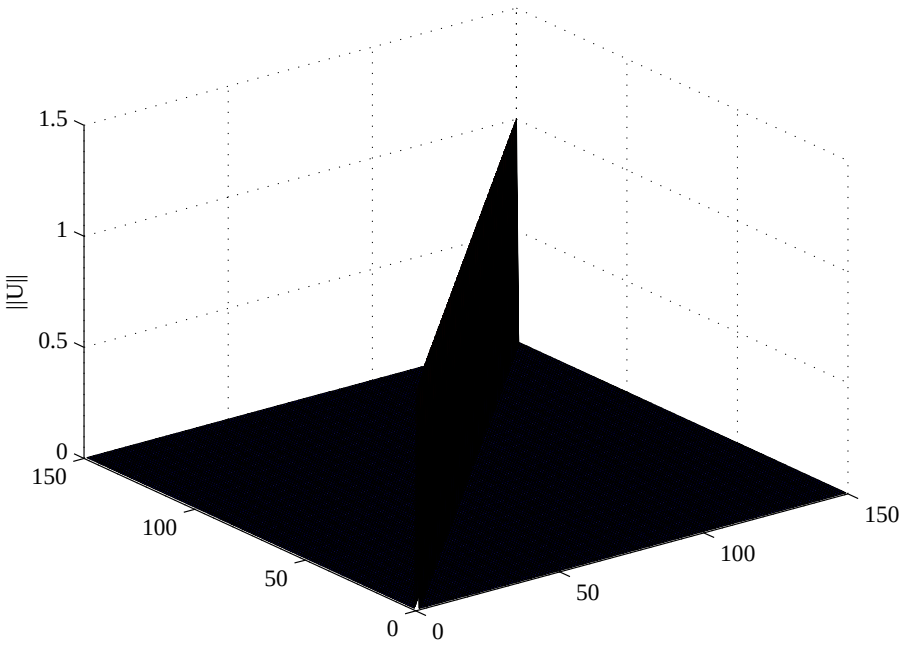


Figure 4.17: $|\mathbf{U}|$ for a system with $L = 1$ CIR coefficients.

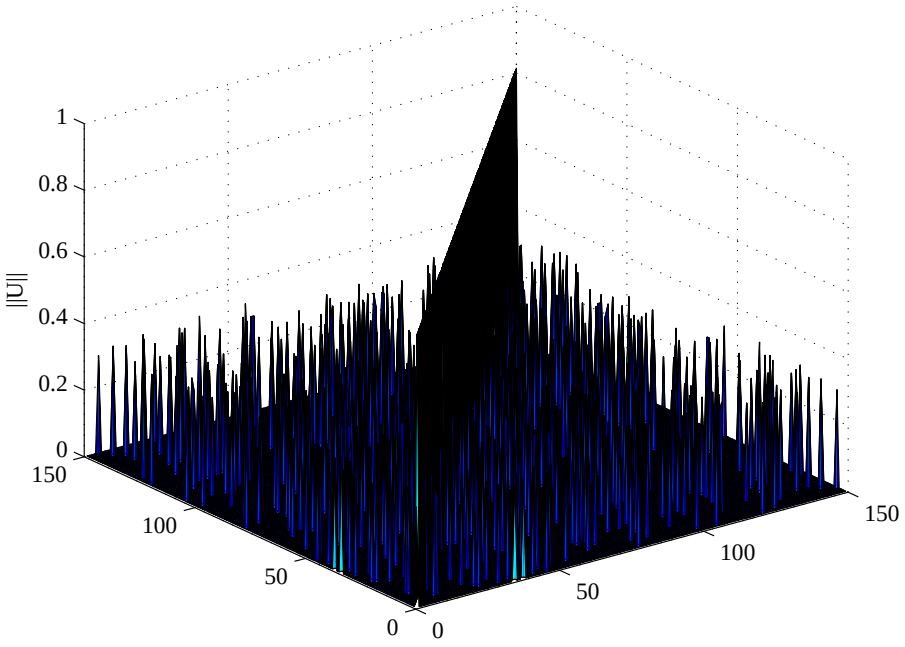


Figure 4.18: $|\mathbf{U}|$ for a system with $L = 3$ CIR coefficients.

4.3 THE DBN-TE ALGORITHM

After making preparations for the turbo equalization problem to be modeled as a quasi-DAG, as explained in the previous section, the DBN-TE algorithm can be executed. This author assumes a system with an uncoded block length of N_u, using the rate $R_c = 1/3$, constraint length $K = 3$, convolutional encoder in Fig. 4.19 to produce N_c bits, where $N_c = N_u/R_c$. The coded bits are interleaved with a random interleaver and passed through a multipath channel with a CIR of length L.

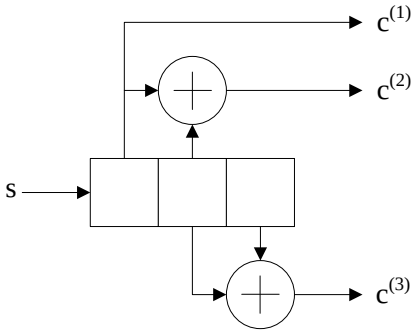


Figure 4.19: Rate $R_c = 1/3$ convolutional encoder.

4.3.1 Graph Construction

The graph is constructed to model the possible outputs of a convolutional encoder in much the same way as a trellis is constructed in a conventional MAP decoder. For the DBN-TE graph the number of states per time instant t is equal to the number of possible *state transitions*, given by $M = 2^K$, where K is the encoder constraint length. This is different from the number of states in a MAP decoder, which is equal to the number of possible *states*, given by $M = 2^{K-1}$. In *Section 4.4* it will be shown how the number of states can be reduced to $M = 2^{K-1}$. The number of time instants in the DBN-TE graph is equal to the number of uncoded bits N_u, which is also equal to the number of codewords. Fig. 4.20 (a) and (b) show the graphical model of a DBN-TE for the first and subsequent iterations respectively, where the dashed lines in Fig. 9 (b) depict weak connections to past and future states due to ISI. Each $\mathbf{X_t}$ on the graph contains a set of M state transitions $x_t^{(m)}$, where $t = 1, 2, ..., N_u$ and $m = 1, 2, ..., M$.

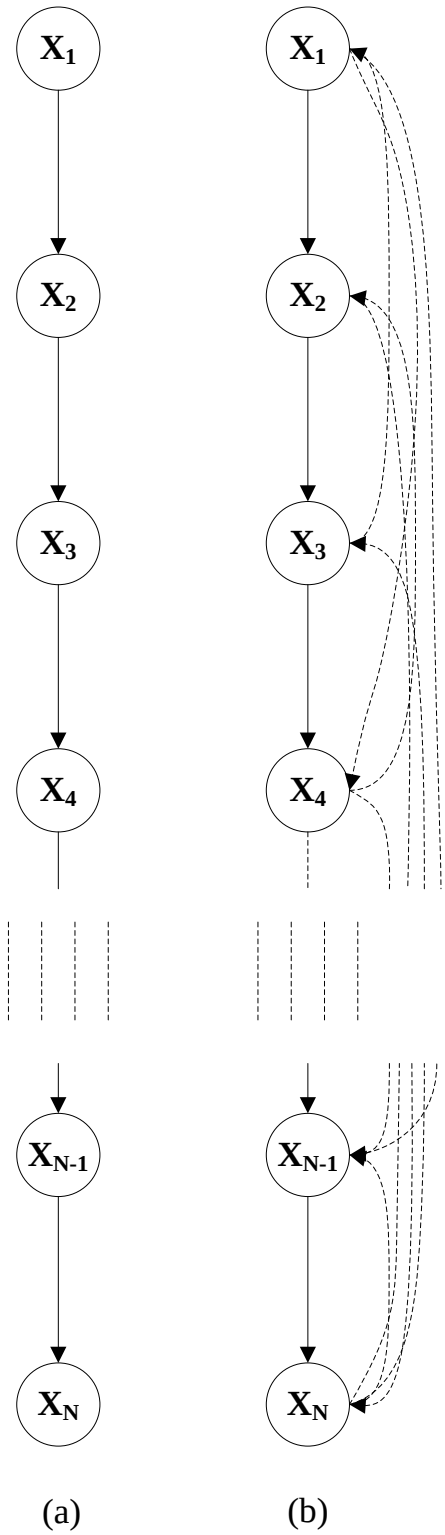


Figure 4.20: Graphical models for the (a) first iteration and (b) subsequent iterations.

During the first iteration no coded bit estimates $\tilde{\mathbf{c}}$ are available, so the graphical model is a pure DAG because the current state is only dependent on the previous state. Hence only $U_{t,t}$ is used in the cost function of the sensor model, where $U_{t,t}$ is a coefficient on the diagonal of the new channel matrix $\mathbf{U}$ in (4.52). After the first iteration, estimates of the coded bits $\tilde{\mathbf{c}}$ are produced and can therefore be used in subsequent iterations. During subsequent iterations then, $U_{t,u}$ and $U_{t,v}$ are also considered in the cost function of the sensor model, where $u = 1,2,...,t-1$ and $v = t+1,t+2,...,N_u$.

4.3.2 State Transition Output Table

The output associated with each state transition is also tabulated using the encoder state diagram in Fig. 4.21. The output produced by each state transition $x_t^{(m)}$, $m = 1,2,...,8$, is determined by loading the bit-values of the current state into the leftmost $K-1=2$ fields of the convolutional encoder in Fig. 4.19, and then placing a 0 and a 1 respectively on the input of the encoder, each producing a new codeword $c^{(1)}c^{(2)}c^{(3)}$ at the output. This process is followed exhaustively and tabulated. Table 4.1 shows the state transition outputs of the encoder in Fig. 4.19 that results from moving from one state to the next.

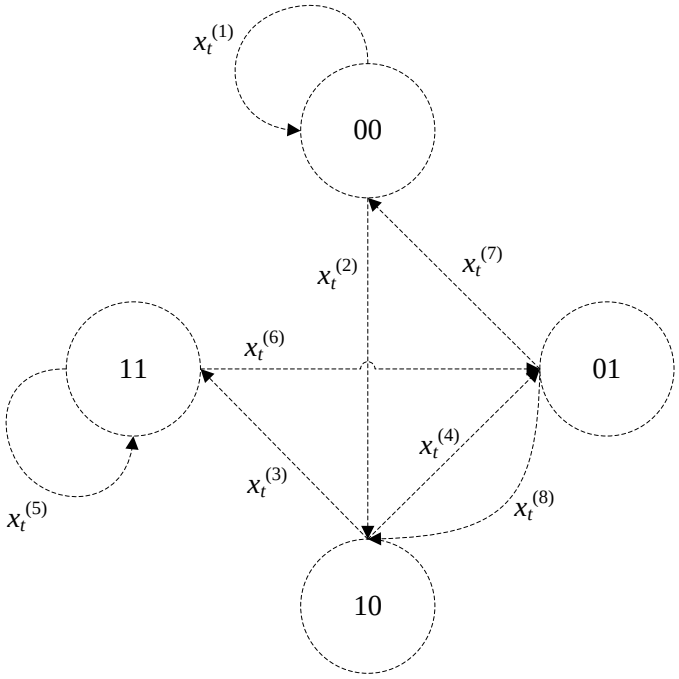


Figure 4.21: Convolutional encoder state diagram.

Table 4.1: State transition output table

	$c^{(1)}$	$c^{(2)}$	$c^{(3)}$
$x_t^{(1)}$	0	0	0
$x_t^{(2)}$	1	1	0
$x_t^{(3)}$	1	0	1
$x_t^{(4)}$	0	1	1
$x_t^{(5)}$	1	0	0
$x_t^{(6)}$	0	1	0
$x_t^{(7)}$	0	0	1
$x_t^{(8)}$	1	1	1

4.3.3 Transition Probability Table

The DBN-TE depends on a transition model to describe the permissible state transitions. This is constructed by examining the encoder state transition diagram in Fig. 4.21 and noting the possible state transitions. The solid lines and dashed lines indicate state transitions caused by ones and zeros at the input of the encoder. It is clear from Fig. 4.21 that only two state transitions emanate from any given state transition (one caused by a 1 at the input and one caused by a 0 at the input). Table 4.2 shows the transition probabilities of the encoder in Fig. 4.19, of which the state transition diagram is shown in Fig. 4.21.

4.3.4 Sensor Model

The conditional probabilities obtained from the sensor model for the respective forward- and backward messages, $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$ and $\mathbf{P}(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})$ are determined by calculating a metric between the observed variable $\mathbf{e}_t$ and the hidden variable $x_t^{(m)}$, where $m = 1,2,...,M$. The observed variable $\mathbf{e}_t$ consists of ${R_c}^{-1}$ received symbols $r_{t'+1}$ to $r_{t'+{R_c}^{-1}}$, where $t' = ((t-1)/R_c)+1$ (t runs from 1 to N_u and t' runs from 1 to N_c), and the hidden variable $x_t^{(m)}$ consists of the output ($c^{(1)}$, $c^{(2)}$ and $c^{(3)}$) associated with state transition $x_t^{(m)}$ in Table 4.1.

During the first iteration, only the dominant connection $U_{t',t'}$ between the observed variable $\mathbf{e}_t$ and the hidden variable $\mathbf{x}_t$ is used in the cost calculation, since no coded bit estimates $\tilde{\mathbf{c}}$ are available at

Table 4.2: Transition probability table

	$x_{t+1}^{(1)}$	$x_{t+1}^{(2)}$	$x_{t+1}^{(3)}$	$x_{t+1}^{(4)}$	$x_{t+1}^{(5)}$	$x_{t+1}^{(6)}$	$x_{t+1}^{(7)}$	$x_{t+1}^{(8)}$
$x_t^{(1)}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0	0
$x_t^{(2)}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0
$x_t^{(3)}$	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
$x_t^{(4)}$	0	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$
$x_t^{(5)}$	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
$x_t^{(6)}$	0	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$
$x_t^{(7)}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0	0
$x_t^{(8)}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0

that point. Recall that $\mathbf{U}$ is the new channel matrix brought about by performing the transformation in (4.50), resulting in the new transmission model in (4.52) having the desired properties to model the system as a quasi-DAG. During the first iteration the system can therefore be modeled as a pure DAG, as if no ISI occurred. Given a hidden variable or state transition output x_{t+1}^n, its associated bits (as tabulated in Table 4.1) are used together with observed variables / received symbols $r_{t'+1}$ to $r_{t'+{R_c}^{-1}}$, where $t' = ((t-1)/R_c)+1$ and ${R_c}^{-1}$ is the number of encoder output bits, to calculate the cost of the nth state transition at time instant t

$$\Delta_t^n = \sum_{j=0}^{R^{-1}-1} |r_{t'+j} - U_{t'+j,t'+j} c_n^{j+1} \beta(i)|^2, \tag{4.53}$$

where $t = 1,2,...,N_u$ runs over time for the uncoded bit estimates and $\beta(.)$ is a function that produces an optimization scaling factor for each iteration i. $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$ in (4.8) is determined by

$$\mathbf{P}(\mathbf{e}_{t+1}|x_{t+1}^n) = \alpha \exp(-\Delta_t^n/2\sigma^2) \tag{4.54}$$

where α is a normalization constant and σ is the noise standard deviation.

During subsequent iterations, LLR estimates of the uncoded bits are available, since the first set of LLRs are produced after the first iteration. Therefore the system can be modeled as a quasi-DAG due to the fact that a dominant connection exists between the observed variable $\mathbf{e}_t$ and the hidden variable $\mathbf{x}_t$ and weak connections between the observed variable $\mathbf{e}_t$ and other hidden variables. Thus the rest of the coefficients $U_{t',1}$ to U_{t',N_c} (and not only $U_{t',t'}$) are also used in the cost calculation. Analogous to the first iteration, the output bits associated with a given state transition x_{t+1}^n are used together with

the observed variables $r_{t'+1}$ to $r_{t'+R_c{}^{-1}}$ *as well as* the LLR estimates $\tilde{\mathbf{c}}$ of the uncoded bits to calculate the cost of the nth state transition at time instant t. Therefore, the cost of the nth state transition for subsequent iterations at time instant t is given by

$$\Delta_t^n = \sum_{j=0}^{R_c{}^{-1}+1} |r_{t'+j} - U_{t'+j,t'+j} c_n^{j+1} - \sum_{v=1, v\neq t', |U_{t'+j,v}|>0}^{N_c} U_{t'+j,v} \tilde{c}_v \beta(i)|^2. \tag{4.55}$$

The last term in (4.55) contains the ISI terms that must be subtracted from the received symbols in order to minimize Δ_t^n so that $\mathbf{P}(\mathbf{e}_{t+1}|x_{t+1}^n)$, determined as in (4.54), can be maximized.

4.3.5 Computing LLR Estimates

After the forward and backward messages have been combined as in (4.10), the LLRs for each uncoded bit is determined from the graph. $R_c{}^{-1}$ LLR vectors of length N_u are determined - each one corresponding to one output bit of the encoder - after which they are multiplexed to form one vector of length N_c containing the LLR estimates $\tilde{\mathbf{c}}$ of the coded bits $\mathbf{c}$. With reference to the state transitions in Table 4.1, the LLRs for the convolutional encoder in Fig. 4.19 are determined as follows:

$$\tilde{\mathbf{c}}^{(1)} = \tanh\left(\log\left(\frac{\sum_{j=1,c^{(1)}=1}^{M} \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}{\sum_{j=1,c^{(1)}=0}^{M} \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}\right)\right) \tag{4.56}$$

$$\tilde{\mathbf{c}}^{(2)} = \tanh\left(\log\left(\frac{\sum_{j=1,c^{(2)}=1}^{M} \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}{\sum_{j=1,c^{(2)}=0}^{M} \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}\right)\right) \tag{4.57}$$

$$\tilde{\mathbf{c}}^{(3)} = \tanh\left(\log\left(\frac{\sum_{j=1,c^{(3)}=1}^{M} \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}{\sum_{j=1,c^{(3)}=0}^{M} \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}\right)\right). \tag{4.58}$$

The final LLR vector is constructed by multiplexing the respective LLR vectors such that

$$\tilde{\mathbf{c}} = \{\tilde{c}_1^{(1)}, \tilde{c}_1^{(2)}, c_1^{(3)}, \tilde{c}_2^{(1)}, \tilde{c}_2^{(2)}, \tilde{c}_2^{(3)}, \ldots, \tilde{c}_{N_u}^{(1)}, \tilde{c}_{N_u}^{(2)}, \tilde{c}_{N_u}^{(3)}\} \tag{4.59}$$

which is used in (4.55) in the next DBN-TE iteration.

4.3.6 Dynamic LLR Updates

It was shown in *Section 4.3.5* that the LLR estimates are only calculated after the forward and backward messages have been combined. The forward and backward messages are normally combined once each message has been determined across the entire received data block, from which LLR estimates are calculated to be used in the next iteration.

What if LLR estimates can be calculated while the forward and backward messages are being formed? Since the DBN-TE iteratively improves upon estimates in previous iterations, will it not be beneficial to provide new LLR estimates as soon as they become available? Fig. 4.22 shows the progression of the forward and backward messages in three stages. In Fig. 4.22 (a) the forward message (left to right) and backward message (right to left) are being constructed, but have not reached the center of the graph. In Fig. 4.22 (b) the forward and backward messages have reached the center of the graph and are now overlapping by one time instant. In Fig. 4.22 (c) the forward and backward messages are almost completely determined and are overlapping by a large number of time instants.

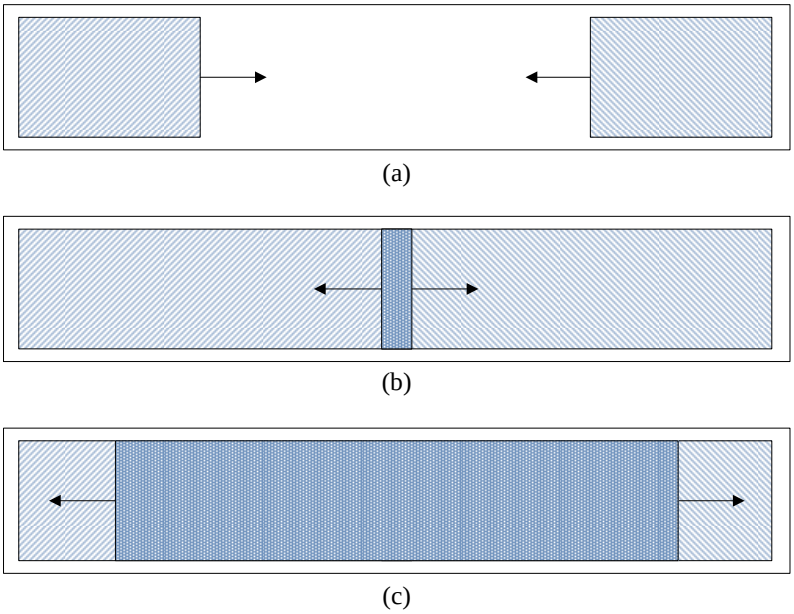


Figure 4.22: Three stages of the forward, backward, and forward-backward message.

From Fig. 4.22 it is clear that the forward-backward message, and hence the LLR estimates, can be determined as soon as the forward and backward messages overlap, but only for the time instants for which they overlap. Since the DBN-TE relies on past and future state estimates, it is crucial that accurate information be provided as soon as it becomes available. By calculating the LLR estimates at the earliest, the DBN-TE can use these new LLR estimates in the *current iteration*, and not only in the *next iteration*. This modification does not affect the computational complexity of the DBN-TE at all, but provides a means to improve its performance. Instead of calculating the LLR estimates at the end of an iteration, each LLR is calculated as soon as the required information becomes available.

4.3.7 Iterative System

The DBN-TE produces estimates regarding the uncoded source information at its output, and improves the quality of these estimates with each iteration. During the first iteration the DBN-TE only relies on the dominant connection between the observation and the hidden variables at a given time instant, but starts using estimates from past and future time instants as they become available, as explained in *Section 4.3.6*. To summarize the DBN-TE algorithm:

1. Set up the $M \times N_u$ graph, where $M = 2^{K-1}$ is the number of states per time instant, and N_u is the uncoded data block length. This step also involves the calculation of the transition probability table and the state transition table, based on the structure of the convolutional encoder.

2. Using the sensor model, calculate the forward and backward messages. Calculate the forward-backward message, and hence the corresponding LLRs, as soon as the forward and the backward messages overlap.

3. Repeat the previous step until the iteration number equals the predefined number of iterations Z.

4. When Z iterations are completed, determine the coded bit estimates from the LLRs.

Fig. 4.23 and Fig. 4.24 show the output of the DBN-TE for a coded data block length of $N_c = 600$, a CIR length of $L = 2$, $Z = 5$ iterations, at a E_b/N_0 of 4 dB, with and without dynamic LLR updates. Fig. 4.23 shows how the LLR estimates are recalculated after each iteration. The first LLR estimates are only available after the first iteration and can therefore only be used during the second iteration. In Fig. 4.24 it is clear that LLR estimates are available already during the first iteration, since LLR estimates are calculated from the forward-backward message as soon as the forward message and the backward message overlap. This allows for LLR estimates to be used in the second half of the first iteration.

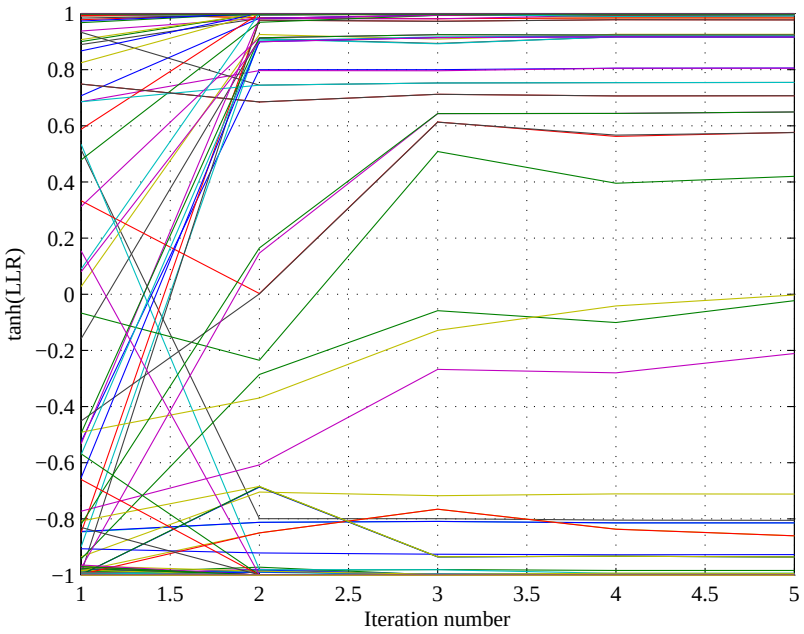


Figure 4.23: DBN-TE LLR convergence without dynamic LLR updates.

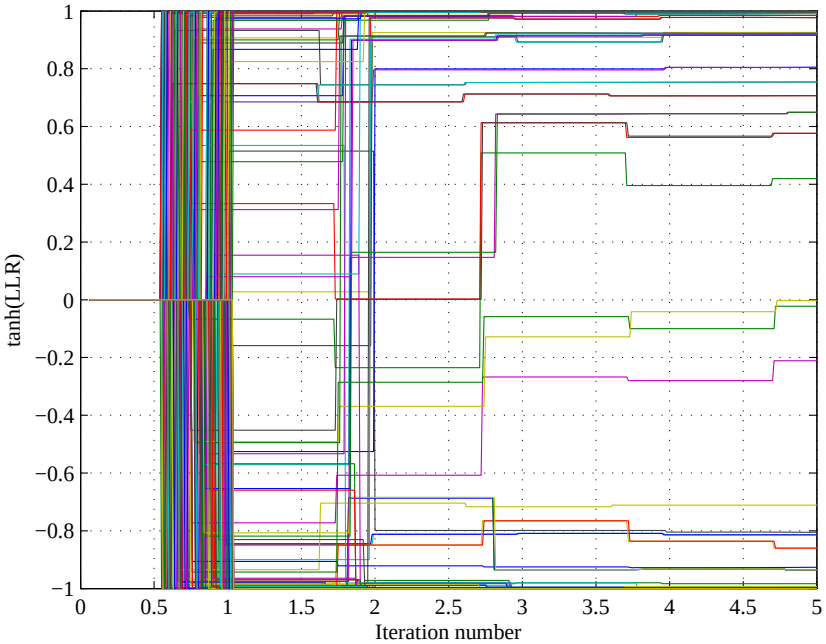


Figure 4.24: DBN-TE LLR convergence with dynamic LLR updates.

4.4 COMPLEXITY REDUCTION

The derivation of the DBN-TE was performed using a DBN framework by assuming that each state transition and its resulting encoder output is treated as a unique state. Whilst this approach is more descriptive of the problem domain, another approach may be followed in order to reduce the number of required states per time instant by half, therefore effectively reducing the computational complexity by half. Instead of the state $\mathbf{X}_t$ being represented by the *state transitions*, one rather uses the *encoder states* in the encoder state diagram in Fig. 4.21 to represent each state $\mathbf{X}_t$.

From Fig. 4.21 it is clear that each encoder state has two parent states (which may cause a given state) and two child states (which may be caused by a given state). Therefore, given any encoder state in Fig. 4.21, the output produced by a transition from each child state to that state can be uniquely determined from the state transition output table in Table 4.1.

The number of states per time instant is therefore reduced from $M = 2^K$ to $M = 2^{K-1}$, where each state $\mathbf{X}_t$ represents two encoder outputs. For the forward message the encoder output associated with the current state depends on the two previous states in $\mathbf{X}_{t-1}$. Similarly, for the backward message the encoder output associated with the current state depends on the two next states in $\mathbf{X}_{t+1}$. For clarity, Table 4.3 and Table 4.4 show the state transition output tables from the perspective of the forward message as well as the backward message. Even though Table 4.3 and Table 4.4 contain the same information, the state transitions are reshuffled in order to highlight current state $\mathbf{X}_t$ and the corresponding previous states $\mathbf{X}_{t-1}$ and next states $\mathbf{X}_{t+1}$, together with their corresponding outputs. The only difference between the DBN-TE with $M = 2^K$ states and the DBN-TE with $M = 2^{K-1}$ states is that the latter uses *one* state to represent *two* state transitions, whereas the former uses *one* state to represent *a single* state transition. The computational complexity of the reduced complexity DBN-TE is therefore halved. The performance of the DBN-TE with this complexity reduction remains the same as before. LLRs are also computed by noting the relevant positions of ones and zeros in the state transition table and marginalizing by summing over all states at a given time instant t.

Table 4.3: State transition output table for the forward message

X_{t-1}	X_t	$\mathbf{c}_t$
00	00	000
01		001
00	10	110
01		111
10	01	011
11		100
10	11	101
11		010

Table 4.4: State transition output table for the backward message

X_t	X_{t+1}	$\mathbf{c}_t$
00	00	000
	10	110
10	01	011
	11	101
01	00	001
	10	111
11	01	100
	11	010

4.5 COMPUTATIONAL COMPLEXITY ANALYSIS

4.5.1 DBN-TE vs CTE

The computational complexity of the DBN-TE and the CTE are presented in this section. The complexity equations were derived by counting the number of computations needed to perform turbo equalization. The computational complexity of the DBN-TE without a prefilter (PF), together with the minimum and maximum complexities with Cholesky decomposition filtering (CPF) and MMSE-

DF filtering (MPF), was determined as

$$CC_{DBN-TE} \approx O(Z(N_c M_d Q k)), \tag{4.60}$$

$$CC_{DBN-TE,CPF} \approx O(Z(N_c M_d Q k) + 2F(N_c/F)^i + F(N_c/F)^2 + (2L)^3 + N_c^i + N_c^2), \tag{4.61}$$

where $i = 2.376$ when optimized matrix inversion and multiplication are used, and $i = 3$ when normal operations are used. Furthermore

$$CC_{DBN-TE,MPF} \approx O(Z(N_c M_d Q k) + N_c^i + N_c^2 + 4L^3 + 4L^2 + N_c L), \tag{4.62}$$

where Z is the number of turbo iterations, M_d is the number of decoder states determined by 2^{K-1} where K is the encoder constraint length, Q is the number of interfering symbols which can be approximated by $Q \approx 2L-1$, R_c is the code rate, k is the number of encoder output bits, and F is the number of frequency hops. The approximation for Q was obtained empirically by calculating the average number of interfering symbols in the new channel $\mathbf{U}$ in (4.51) for a given original channel length L, after the transformation in (4.50) had been applied. The complexity of the CTE was determined as

$$CC_{CTE} \approx O(Z(N_c M_e L + N_c M_d k)), \tag{4.63}$$

where M_e is the number of equalizer states determined by 2^{L-1} for BPSK modulation.

Fig. 4.25 and Fig. 4.26 shows the computational complexity graphs of the DBN-TE and the CTE, normalized by the number of coded transmitted symbols, for CIR lengths from $L = 1$ to $L = 25$, where $Z = 5$, $R_c = 1/3$, $K = 3$ and $N_u = 400$ ($N_c = 1200$), and where the number of frequencies used for frequency hopping are $F = 1$ and $F = 8$ respectively. Fig. 4.25 and Fig. 4.26 shows the same information as Fig. 4.25 and Fig. 4.26, but for $N_u = 1600$ ($N_c = 4800$).

From Fig. 4.25 it can be seen that the computational complexity of the DBN-TE is much lower than that of the CTE when no prefiltering is performed, without which the DBN-TE will fail. However, when prefiltering via the MMSE-DF is performed the minimum complexity is the same as that of the CTE for $L \approx 6$, and the corresponding maximum complexity meets that of the CTE at $L \approx 13$. Also, when prefiltering via the Cholesky decomposition is performed, the minimum and maximum complexities are equivalent to the complexity of the CTE at $L \approx 8$ and $L \approx 15$ respectively. In all cases the DBN-TE complexity with prefiltering remains approximately constant as the CIR length increases. Fig. 4.26 shows that when frequency hopping is employed, the maximum complexities of the DBN-TE with Cholesky and MMSE prefiltering are approximately equal to their corresponding

minimum complexities, highlighting the complexity reduction with an increase in the number of frequencies used for frequency hopping.

From Fig. 4.27 and Fig. 4.28 it can be seen that the same trends identified in Fig. 4.25 and Fig. 4.26 continue. In Fig. 4.27 the minimum and maximum complexities of the DBN-TE when using Cholesky prefiltering reaches a breakeven point with respect to the CTE at $L \approx 10$ and $L \approx 17$ respectively, while the minimum and maximum complexities when using MMSE-DF prefiltering reaches a break-even point with the CTE at $L \approx 12$ and $L \approx 18$. It is also clear from Fig. 4.28 that the maximum complexities of the DBN-TE using Cholesky and MMSE-DF prefiltering are approximately equal to their corresponding minimum complexities when the number of frequencies is increased.

From Fig. 4.25 through to Fig. 4.28 it was seen that the DBN-TE has complexity that is approximately quadratic (at best) with respect to the coded data clock length (and cubic at the worst), and it is approximately independent of the CIR length. It is also evident that the minimum computational complexities, when using Cholesky and MMSE-DF prefiltering, are very favorable compared to those of the CTE for moderate coded data block length. Apart from the fact that frequency hopping provides a means whereby BER improvement can be achieved in multipath fading channels, it is also clear that the use of frequency hopping reduces complexity. To conclude the computational complexity analysis of the DBN-TE, Fig. 4.29 shows its various complexities for extremely long CIRs with $N_u = 800$ ($N_c = 2400$) and $F = 1$, from which it is clear that it maintains its constant complexity with growth in channel memory. This makes the DBN-TE a very attractive choice for use in systems transmitting convolutionally coded information through higly dispersive multipath channels.

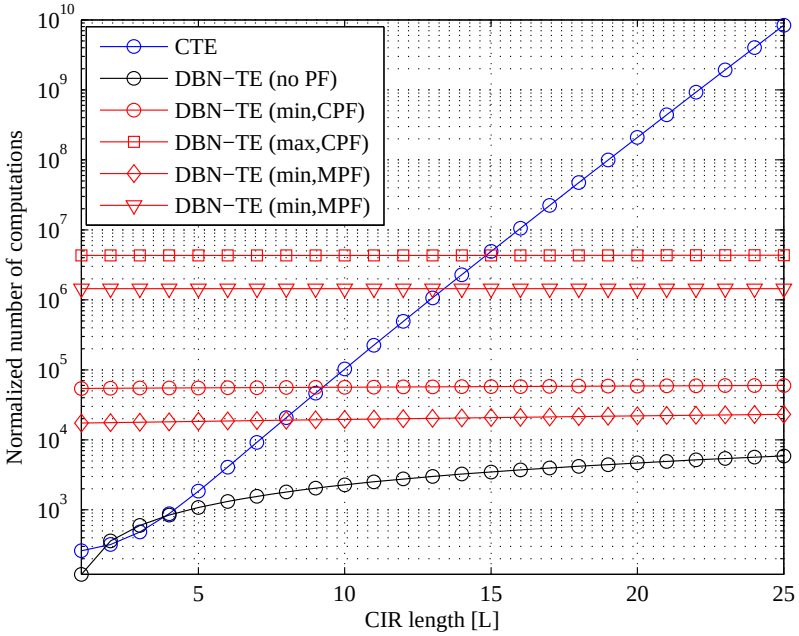


Figure 4.25: DBN-TE and CTE normalized computational complexity for different CIR lengths for $N_u = 400$ and $F = 1$.

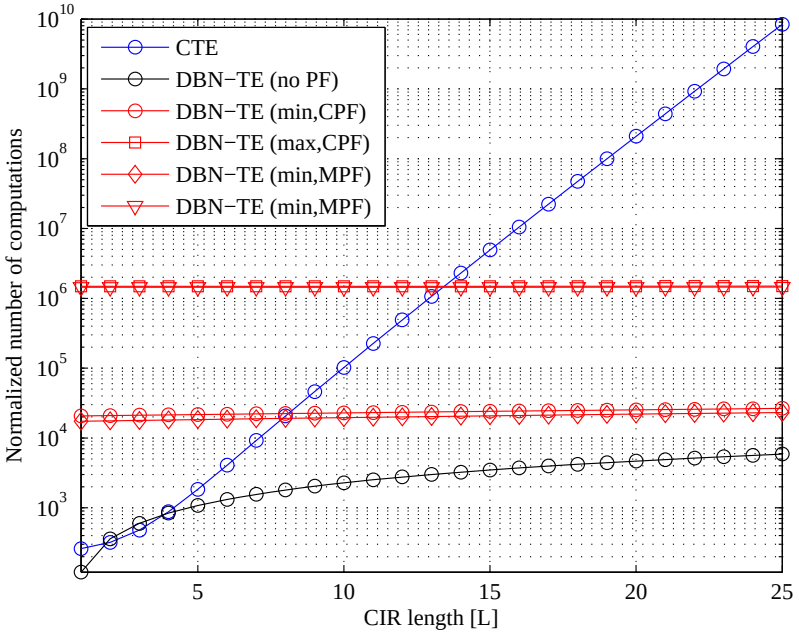


Figure 4.26: DBN-TE and CTE normalized computational complexity for different CIR lengths for $N_u = 400$ and $F = 8$.

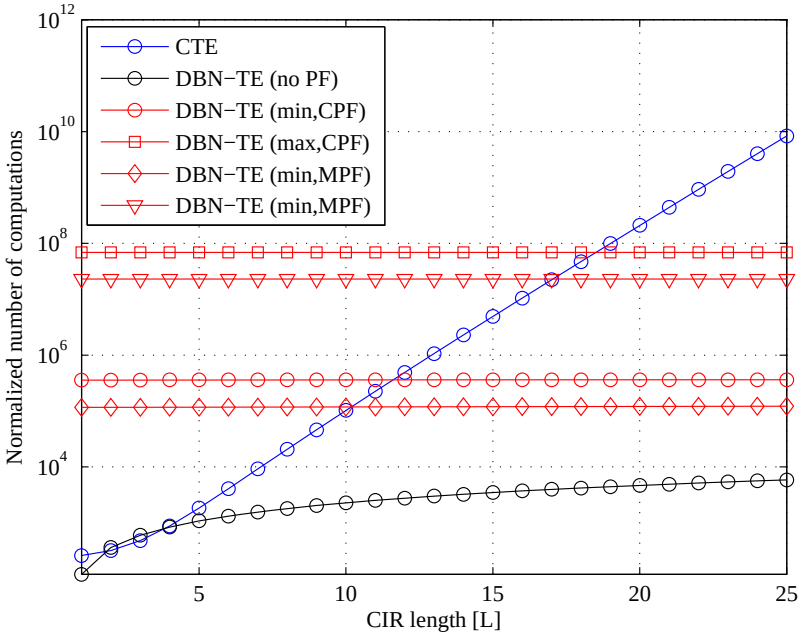


Figure 4.27: DBN-TE and CTE normalized computational complexity for different CIR lengths for $N_u = 1600$ and $F = 1$.

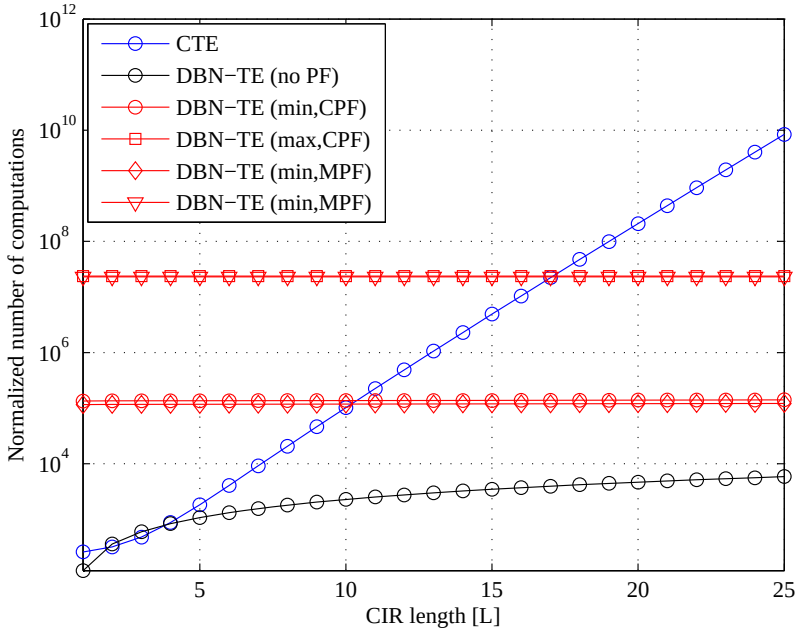


Figure 4.28: DBN-TE and CTE normalized computational complexity for different CIR lengths for $N_u = 1600$ and $F = 8$.

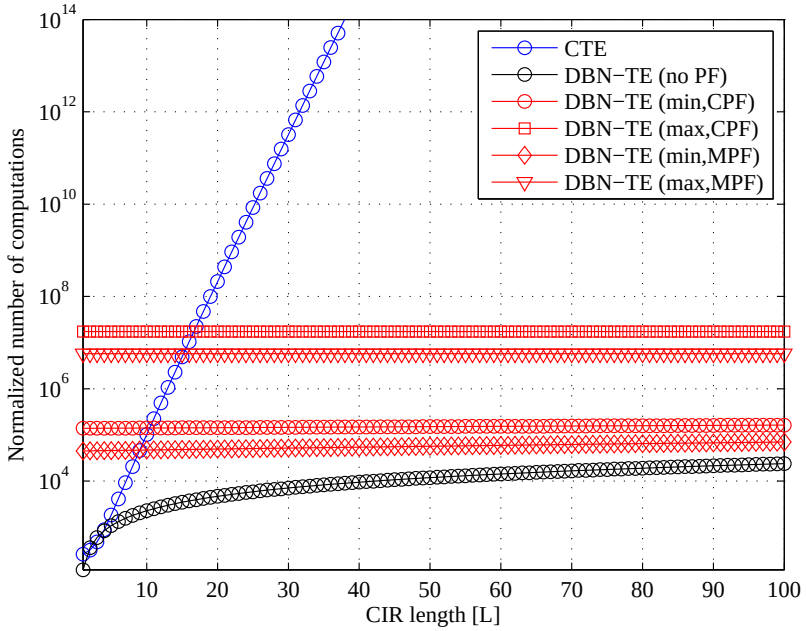


Figure 4.29: DBN-TE and CTE normalized computational complexity for very long CIR lengths for $N_u = 800$ and $F = 1$.

4.5.2 DBN-TE vs LCTEs

Fig. 4.30 shows the minimum computational complexities of the DBN-TE with prefiltering (Cholesky and MMSE-DF), the MMSE-LE/DFE-TE and the SFE/SDFE-TE, using the same parameters as before, for uncoded data block lengths of $N_u = 200$, $N_u = 400$ and $N_u = 800$ and $F = 1$, assuming the computational complexities of the MMSE-LE/DFE (in (3.62) and (3.64)) and the SFE/SDFE (in (3.66) and (3.68)). Assuming a MAP decoder computational complexity of $O(N_c M_d k)$, the resulting LCTE computational complexities are expressed as

$$CC_{MMSE-LE/DFE-TE} = O(Z(N_c(N^2+L^2)+N_c M_d k)) \tag{4.64}$$

and

$$CC_{SFE/SDFE-TE} = O(Z(N_c(N+L)+N_c M_d k)), \tag{4.65}$$

which are shown together with those of the DBN-TE in Fig. 4.30 for uncoded data block lengths of $N_u = 200$, $N_u = 400$ and $N_u = 800$ anf $F = 1$. From Fig. 4.30 it is once again clear that the computational complexity of the DBN-TE remains approximately constant as the CIR length increases, and that the complexity increases as the data block length increases.

In general the computational complexity of the DBN-TE is inferior to that of the MMSE-LE-TE and MMSE-DFE-TE. The DBN-TE complexity employing a prefilter via the MMSE-DF filter reaches a breakeven point with the MMSE-LE/DFE-TE at $L \approx 20$ for $N_u = 200$, at $L \approx 30$ for $N_u = 400$, and at $L \approx 50$ for $N_u = 800$. The high computational complexity of the DBN-TE is due to interleaver mitigation, which requires matrix operations that have quadratic to cubic complexity in the coded data block length N_c. Even so, the computational complexity of the DBN-TE is favorable when compared to that of the CTE for moderate data block lengths.

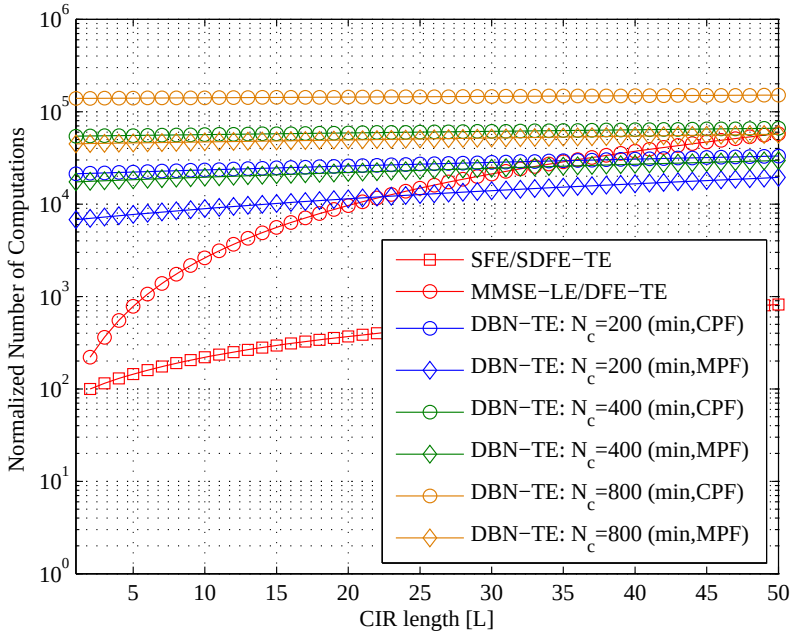


Figure 4.30: DBN-TE, MMSE-LE/DFE and SFE/SDFE normalized computational complexity for different CIR lengths for $N_u = 200$, $N_u = 400$ and $N_u = 800$.

4.6 SIMULATION RESULTS

The DBN-TE is evaluated in static and fading multipath channels. Various simulations are firstly performed in order to compare the performance of the DBN-TE to that of the CTE in [5, 6]. Second, simulations are performed in order to compare the DBN-TE performance to the performance of, among others, the various LCTEs discussed in *Chapter 3*. The simulation environment is described in detail in *Appendix A*. For the first set of simulations, the following standard simulation parameter settings were used:

Parameter	**Setting**
Modulation	BPSK
Interleaver	Random
Uncoded block length (N_u)	400
Coded block length (N_c)	1200
Channel	Fading/Static
CIR length (L)	Varying
Channel Estimation	No
Channel State Information	Perfect CSI
Number of pilots	-
Frequency hopping	Yes
Number of freq. hops	8
Number of CTE iterations (Z_{CTE})	5
Number of DBN-TE iterations (Z_{DBN-TE})	5
Mobile speed (v)	0 km/h
PDP	Uniform ($\mathbf{h}^T\mathbf{h} = 1$)
Minimum phase filtering	Cholesky (fading) / MMSE-DF (static)

In order to evaluate the effect of each parameter on the performance of the DBN-TE, the following simulations were performed by fixing all the parameters and varying one parameter at a time.

4.6.1 DBN-TE vs CTE

4.6.1.1 Optimization

In order to evaluate the performance of the DBN-TE using dynamic LLR updates, the system was simulated for fading (at 0 km/h mobile speed) and static channels with and without dynamic LLR updates. Fig. 4.31 and Fig. 4.32 show the performance of the DBN-TE with and without fading (slow fading), for a channel length of $L = 3$. In Fig. 4.31 it can be seen that the BER performance is best when dynamic updates are performed. From Fig. 4.32 it is clear that dynamic LLR updates provide a significant performance gain for this fading channel. Fig. 4.33 and Fig. 4.34 show the performance of the DBN-TE with and without fading, for a channel length of $L = 5$. Fig. 4.33 shows that the BER performance does not increase with an increase in E_b/N_0 unless dynamic LLR updates are applied. It is clear that dynamic LLR updates provides and increase in performance. In Fig. 4.34 it is clear once again that dynamic LLR updates alone provide improved BER performance. From the results presented here it can be concluded that optimization via dynamic LLR updates allows for an improvement in BER performance. The DBN-TE will therefore henceforward be simulated using dynamic LLR updates for optimization.

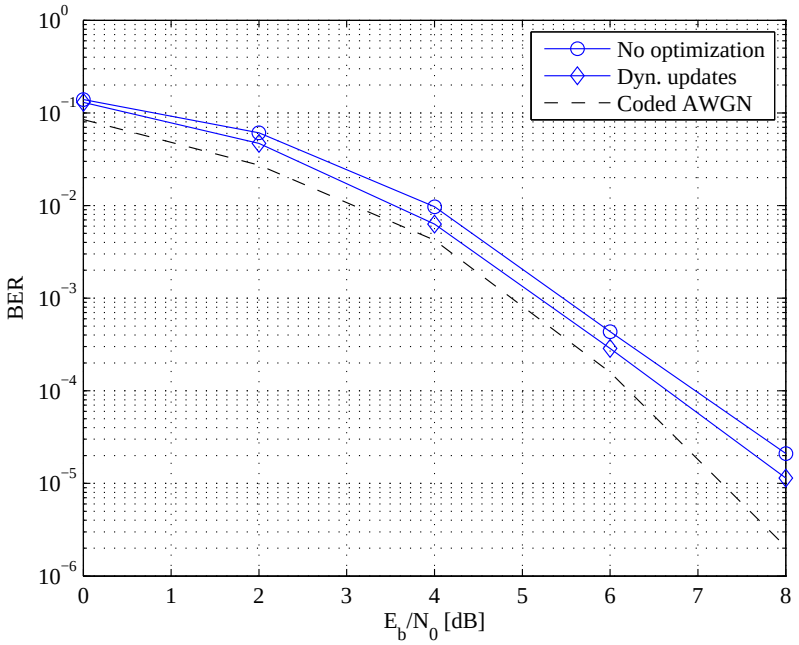


Figure 4.31: DBN-TE performance with and without dynamic LLR updates in a static channel with $L = 3$.

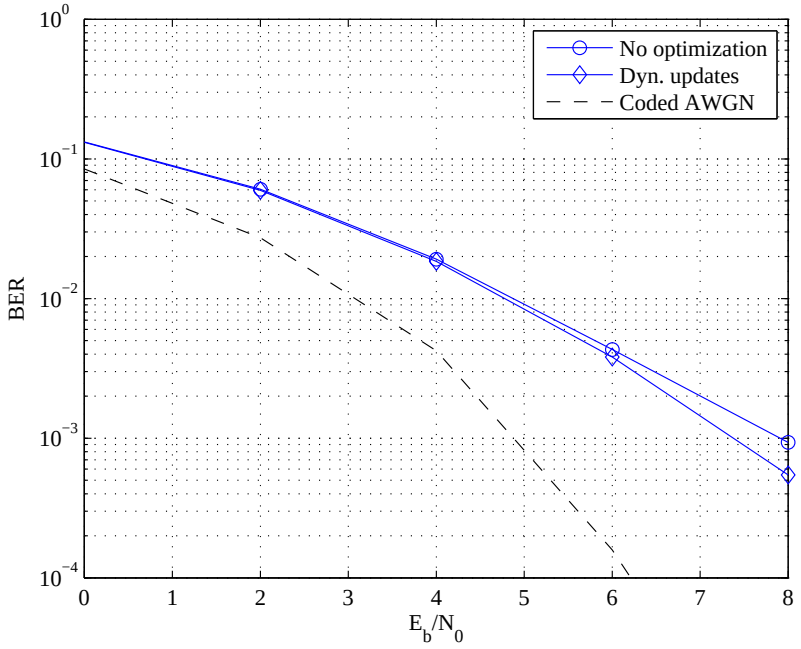


Figure 4.32: DBN-TE performance with and without dynamic LLR updates in a fading channel with $L = 3$.

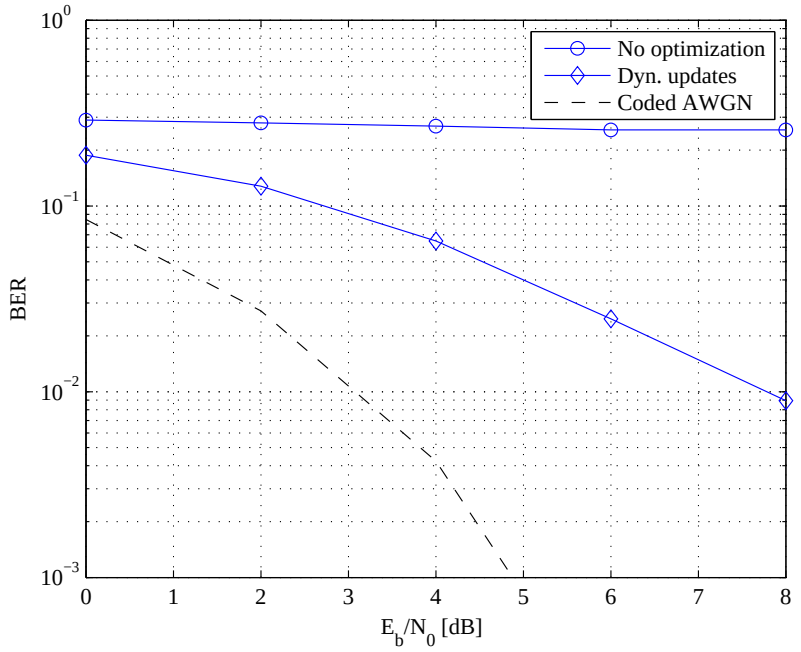


Figure 4.33: DBN-TE performance with and without dynamic LLR updates in a static channel with $L = 5$.

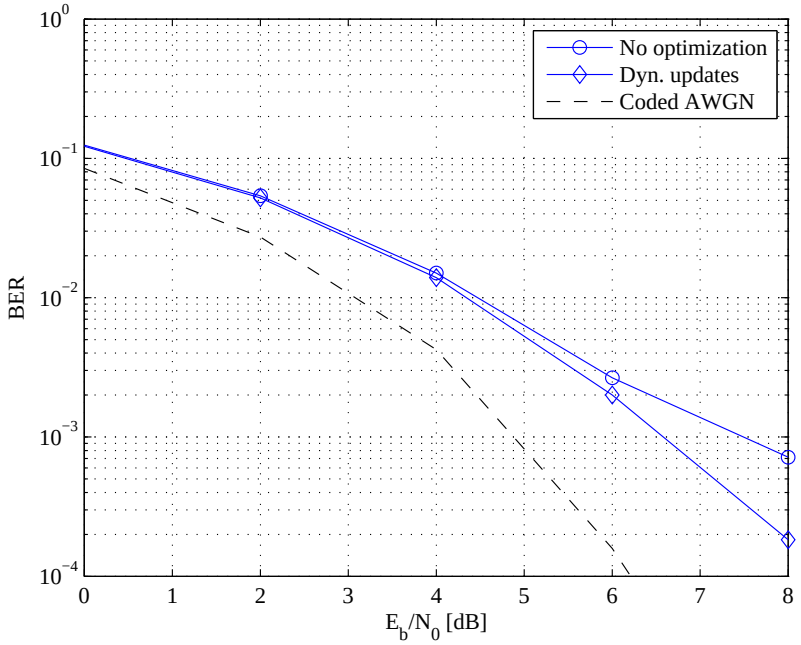


Figure 4.34: DBN-TE performance with and without dynamic LLR updates in a fading channel with $L = 5$.

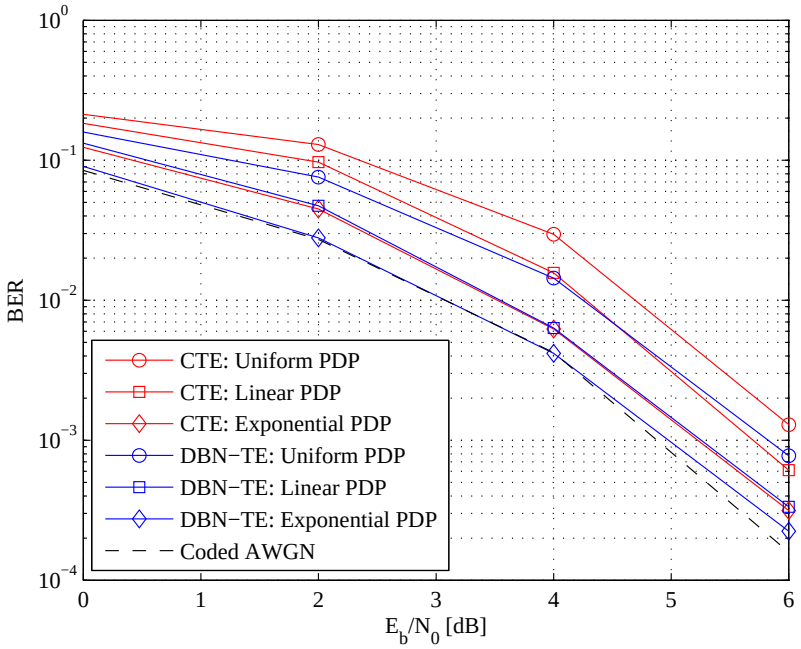


Figure 4.35: DBN-TE and CTE performance for various PDPs in a static channel with $L = 4$.

4.6.1.2 Power Delay Profiles

The DBN-TE is evaluated against the CTE using uniform, linear and exponential PDPs, for both fading and static channels for CIR lengths of $L = 4$ and $L = 8$. Fig. 4.35 shows the performance of the DBN-TE and the CTE in a static channel for $L = 4$ for the various PDPs, while Fig. 4.36 shows the results achieved in a fading channel of the same length. Also, Fig. 4.37 shows the performance of the DBN-TE and the CTE in a static channel for $L = 8$, while Fig. 4.38 shows the results achieved in a fading channel of length $L = 8$.

From Fig. 4.35 it is clear that the DBN-TE outperforms the CTE for all PDPs when the channel is fairly short and static. Fig. 4.36 shows that the performance of the DBN-TE is comparable to that of the CTE in short fading channels for all three PDPs. From Fig. 4.37 it can be seen that the DBN-TE performs worse than the CTE for all but one of the PDPs in static channels of length $L = 8$, while the DBN-TE outperforms the CTE in fading channels of the same length for all PDPs. The underperformance of the DBN-TE in longer static channels is due to the fact that there is too much energy in the latter CIR taps and that the soft feedback mechanism of the DBN-TE fails under these conditions. The DBN-TE is therefore not suitable in static channels that are too long, where the first tap (h_0) of the CIR is not sufficiently dominant.

From these simulations it is clear that the DBN-TE is suitable for use in short static or fading channels, using all three PDPs considered. The DBN-TE is also suitable for use in longer fading channels, but fails in longer static channels because of energy in the latter CIR taps of these channels that cause error propagation via the feedback mechanism inherent in the DBN-TE.

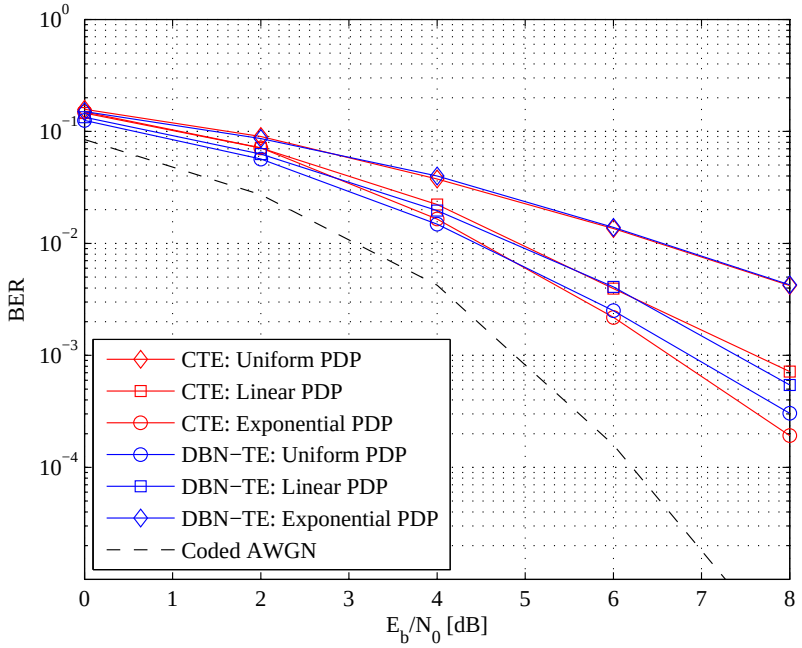


Figure 4.36: DBN-TE and CTE performance for various PDPs in a fading channel with $L = 4$.

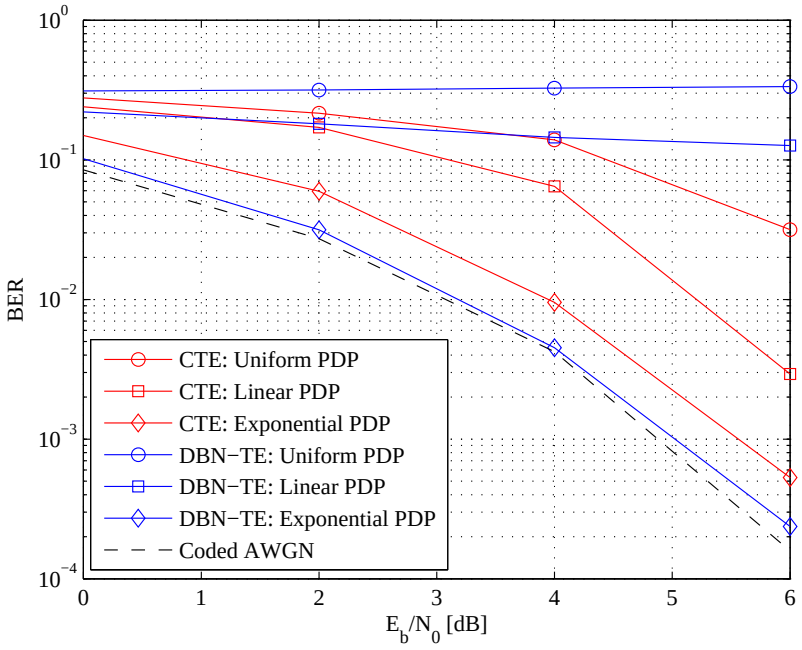


Figure 4.37: DBN-TE and CTE performance for various PDPs in a static channel with $L = 8$.

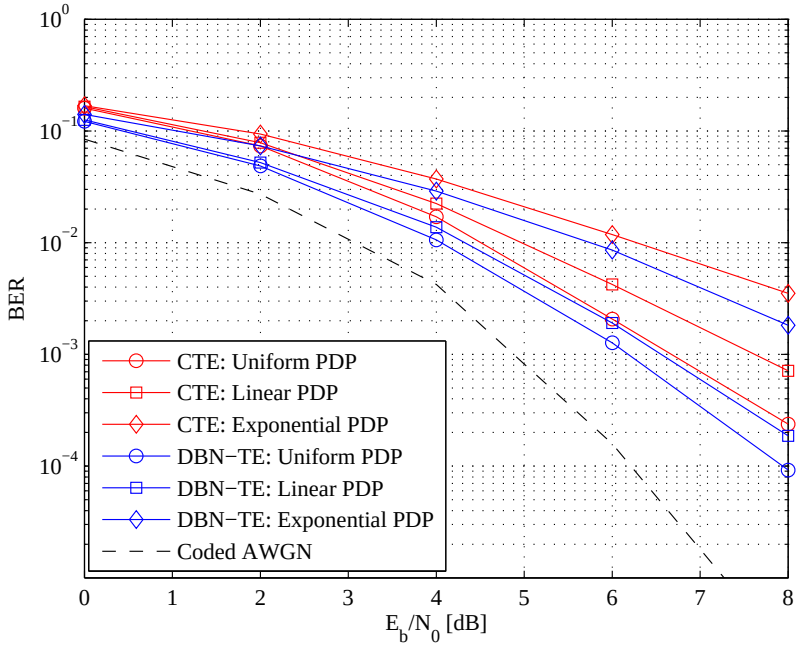


Figure 4.38: DBN-TE and CTE performance for various PDPs in a fading channel with $L = 8$.

4.6.1.3 Mobile Speeds

The DBN-TE and the CTE are evaluated for different mobile speeds in a fading channel with lengths of $L = 4$ and $L = 8$ respectively. Fig. 4.39 shows the performance of the DBN-TE and the CTE for mobile speeds from 20 km/h to 110 km/h in intervals of 20 km/h for $L = 4$, and Fig. 4.40 shows the results in a channel with length $L = 8$.

From Fig. 4.39 it is clear that the CTE outperforms the DBN-TE for $L = 4$, and that the DBN-TE performs the same as the CTE at a mobile speed of 50 km/h. However, at a mobile speed of 80 km/h and 100 km/h the DBN-TE outperforms the CTE by some margin. At higher mobile speeds the CSI is not reliable, since the CIR cannot be assumed to be time invariant owing to fast fading. It seems, however, that the DBN-TE is more resilient against imperfect channel information. Fig. 4.40 shows the performance of the DBN-TE and the CTE for a longer fading channel of length $L = 8$, where the DBN-TE consistently outperforms the CTE. As seen before, the DBN-TE outperforms the CTE in longer fading channels, while the improvement in performance of the DBN-TE over the CTE increases with an increase in mobile speed.

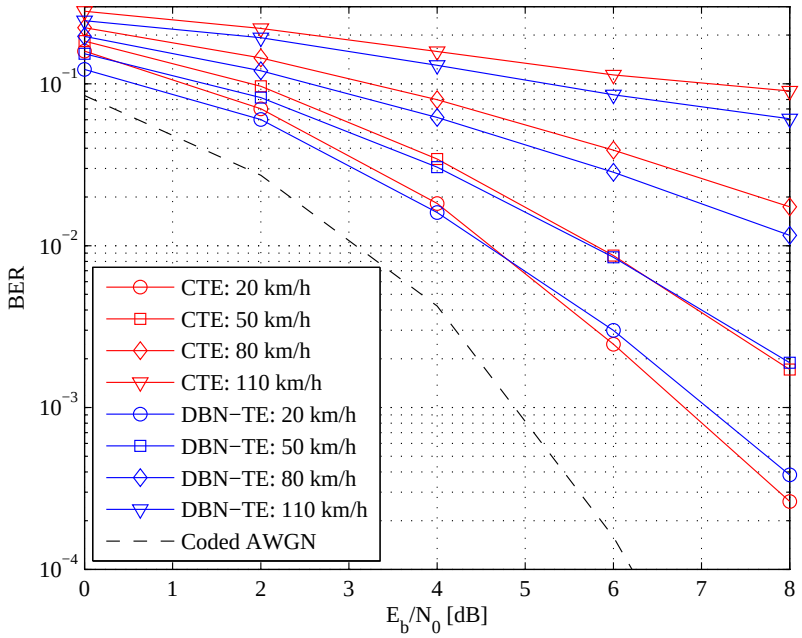


Figure 4.39: DBN-TE and CTE performance for various mobile speeds in a fading channel with $L = 4$.

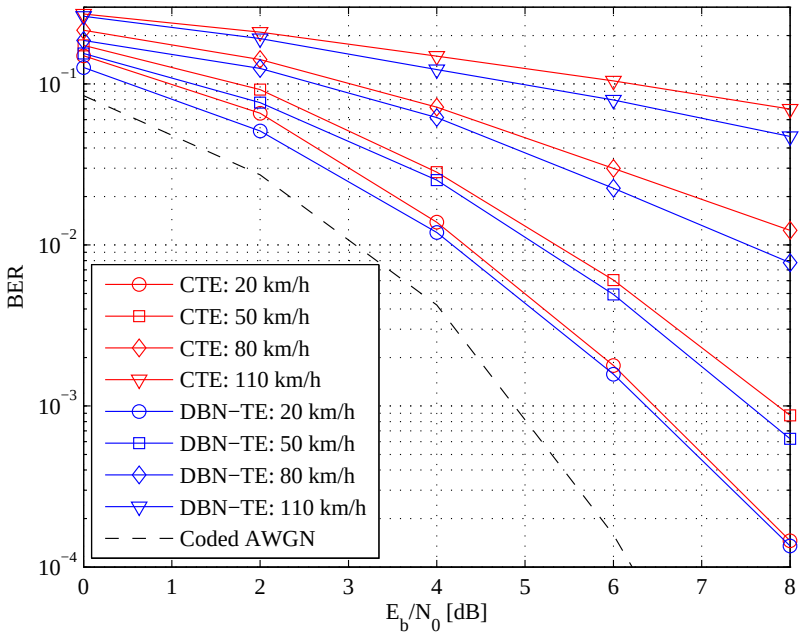


Figure 4.40: DBN-TE and CTE performance for various mobile speeds in a fading channel with $L = 8$.

From these simulations it is concluded that the DBN-TE is useful in systems operating in a mobile fading environment, and the performance of the DBN-TE compared to that of the CTE increases with an increase in mobile speed.

4.6.1.4 Frequency Hops

In order to determine the effect of the number of frequency hops on the performance of the DBN-TE and the CTE, they are evaluated in fading channels of length $L = 4$ and $L = 8$ using $F = 2$, $F = 4$ and $F = 8$ frequency hops. Fig. 4.41 shows the DBN-TE and CTE performance for the various frequency hops in a fading channel of length $L = 4$, and Fig. 4.42 shows the performance for $L = 8$.

In Fig. 4.41 it can be seen that the performance of both the DBN-TE and the CTE improves with an increase in the number of frequency hops, while the CTE outperforms the DBN-TE, as expected for a short channel like this. Fig. 4.42 show the DBN-TE and CTE performance in a length $L = 8$ fading channel, from which it is clear that the performance of both the CTE and the DBN-TE also improves with an increase in the number of frequency hops. Here, however, the DBN-TE outperforms the CTE when $F = 8$ frequency hops are used.

From these results it is clear that the performance of the DBN-TE is comparable to that of the CTE in systems where the frequency is hopped a sufficient number of times during each transmitted data block. This will ensure that the CIR changes with each frequency hopped segment, allowing the DBN-TE to yield better inference than when a single CIR is present if the frequency is not hopped. The DBN-TE is therefore able to exploit the frequency diversity introduced by frequency hopping.

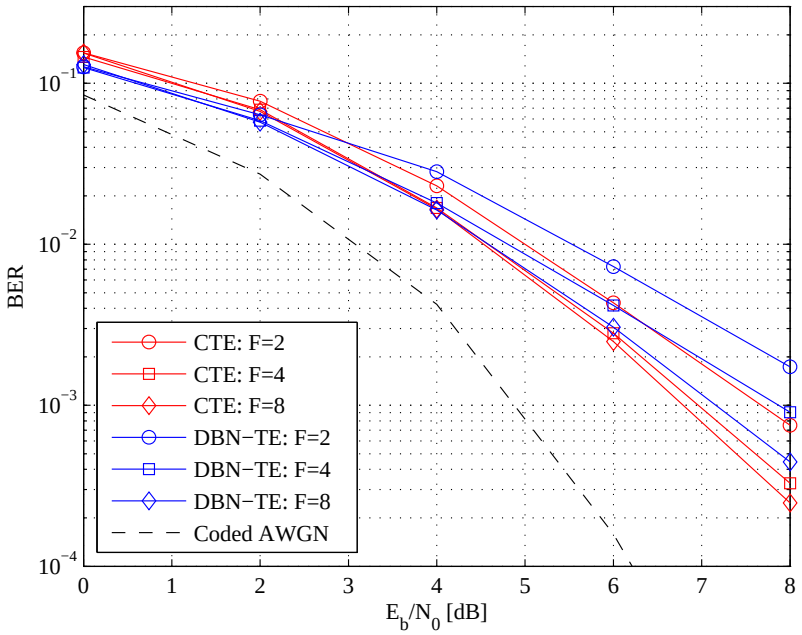


Figure 4.41: DBN-TE and CTE performance for various frequency hops in a fading channel with $L = 4$.

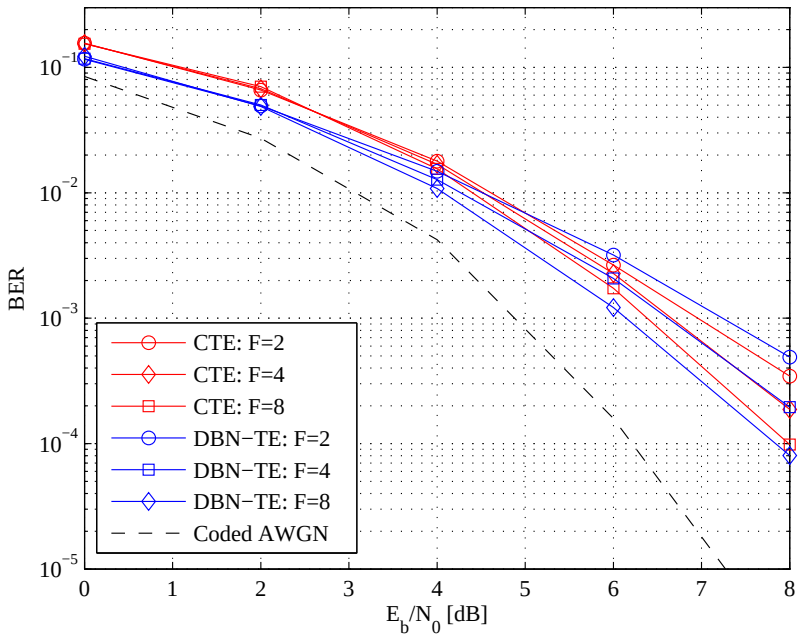


Figure 4.42: DBN-TE and CTE performance for various frequency hops in a fading channel with $L = 8$.

4.6.1.5 Block Lengths

The effect of the block length on the performance of the DBN-TE and the CTE is evaluated by simulating the system for various coded data block lengths. Since a random interleaver is used, it is expected that the performance will increase with an increase in data block length. Simulations are performed in a static channel with a length of $L = 3$ and a fading channel with a length of $L = 5$ where the coded data block lengths are $N_c = 300$, $N_c = 1200$ and $N_c = 4800$ respectively.

Fig. 4.43 and Fig. 4.44 show the performance of the DBN-TE and the CTE for various coded block lengths, for a static channel of length $L = 3$ and a fading channel of length $L = 5$ respectively. In both cases it is clear that the performance improves with an increase in the coded data block length owing to the random distribution of neighboring bits in the coded bit sequence. Fig. 4.43 shows that the DBN-TE outperforms the CTE, especially at low E_b/N_0 levels, and achieves near matched-filter performance for $N_c = 4800$. From Fig. 4.44 it can also be seen that the DBN-TE outperforms the CTE.

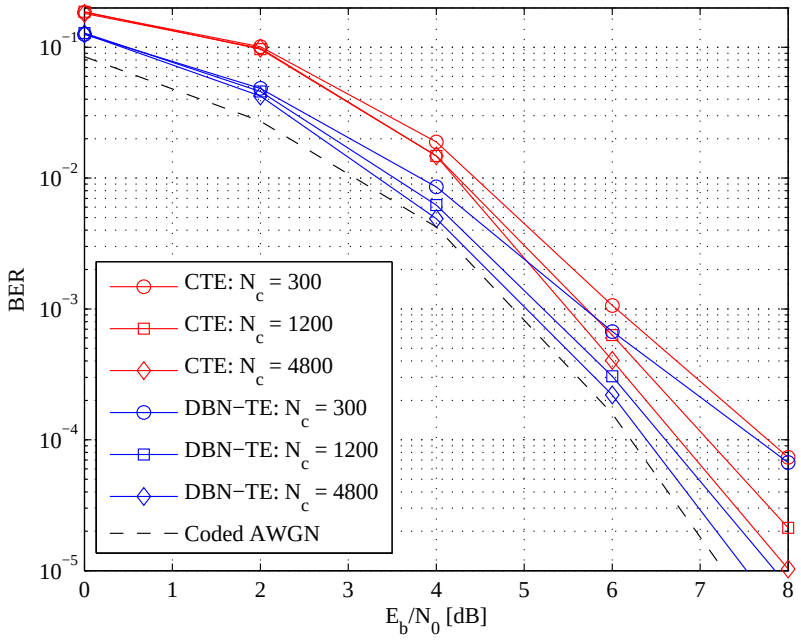


Figure 4.43: DBN-TE and CTE performance for various coded data block lengths in a static channel with $L = 3$.

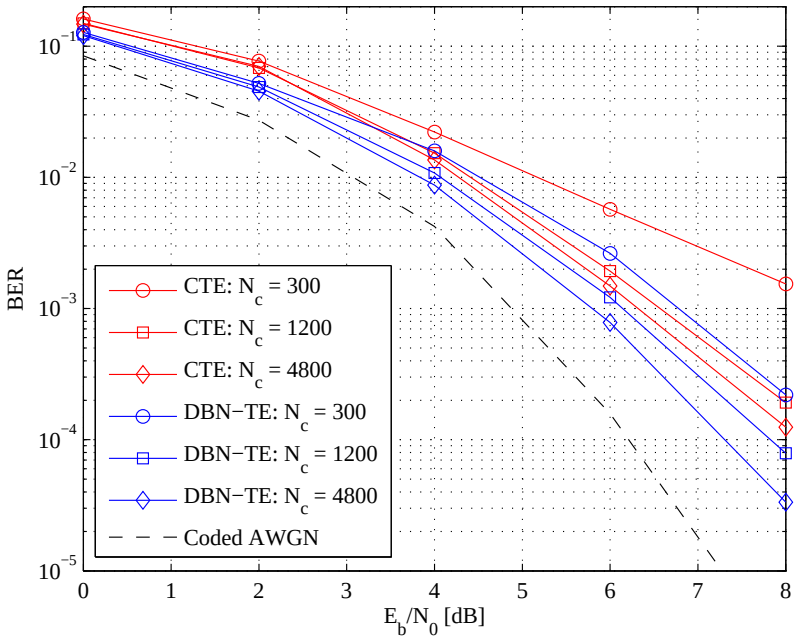


Figure 4.44: DBN-TE and CTE performance for various coded data block lengths in a fading channel with $L = 5$.

From these simulations it is concluded that the DBN-TE is able to exploit the temporal separation of the coded information bits in order to perform near-optimal inference. It seems that it is able to do so more effectively than the CTE, thus achieving better performance for both scenarios.

4.6.1.6 Training Symbols

In a practical wireless communication system perfect CSI is not available, and hence the channel has to be estimated using training or pilot symbols, which are usually placed in the center of the transmitted data block, and are known at the transmitter and the receiver. In order to test the resilience of the DBN-TE against channel estimation errors, it is evaluated alongside the CTE in a system where the channel is estimated using a least squares (LS) channel estimator, as explained in *Appendix A*, using various numbers of training symbols. The DBN-TE and the CTE are simulated in fading channels of length $L = 4$ and $L = 8$ where the number of training symbols are $P = 4L$, $P = 6L$ and $P = 8L$, and the results are compared to the performance achieved when perfect CSI is available.

Fig. 4.45 and Fig. 4.46 show the performance of the DBN-TE and the CTE in fading channels of

length $L = 4$ and $L = 6$ respectively for various numbers of training symbols used for channel estimation. From Fig. 4.45 it can be seen that the DBN-TE performs worse than the CTE for smaller numbers of training symbols for $L = 4$, and only performs slightly worse than the CTE for $P = 8L$ training symbols. Fig. 4.46 shows that the DBN-TE only performs worse than the CTE for $P = 4L$ training symbols when $L = 8$, after which the DBN-TE outperforms the CTE by 0.5 dB.

From these simulations it is concluded that the DBN-TE is able to achieve acceptable performance in the presence of channel estimation errors, while its performance is comparable to that of the CTE.

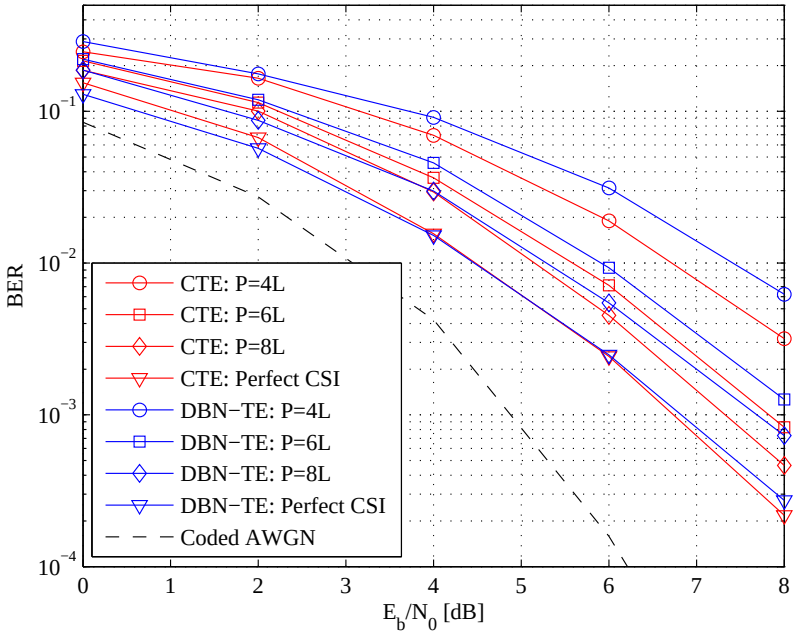


Figure 4.45: DBN-TE and CTE performance for various numbers of training symbols in a fading channel with $L = 4$.

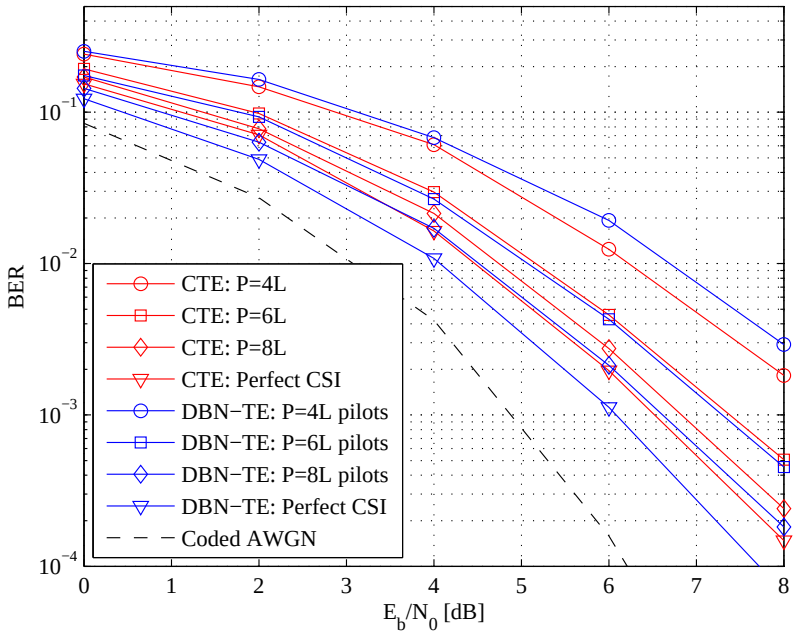


Figure 4.46: DBN-TE and CTE performance for various numbers of training symbols in a fading channel with $L = 8$.

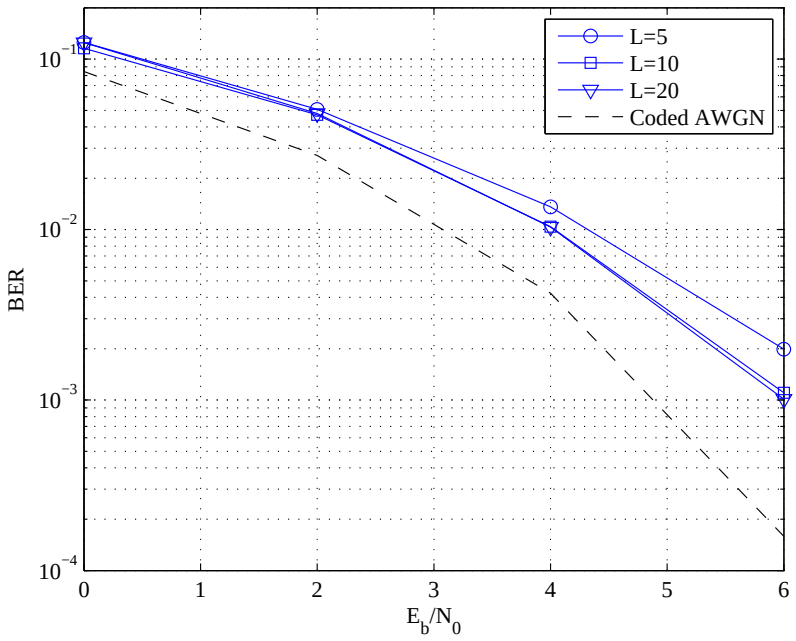


Figure 4.47: DBN-TE performance in long fading channels.

4.6.1.7 Long Memory

Because of its low computational complexity the DBN-TE is able to jointly equalize and decode information transmitted though highly dispersive multipath channels. The DBN-TE is therefore simulated for channels of length $L = 5$, $L = 10$ and $L = 20$. Fig. 4.47 shows the performance of the DBN-TE in these long channels. It is clear from Fig. 4.47 that the DBN-TE effectively jointly equalizes and decodes the information in highly dispersive fading channels, but that the performance does not increase for CIR lengths beyond $L = 10$. This might be remedied by using larger coded data blocks.

4.6.2 DBN-TE vs LCTEs

In this section the DBN-TE is evaluated against LCTEs, which use a low complexity equalizer as a replacement for the optimal, but complex, MAP equalizer. The LCTEs considered are formed by replacing the MAP equalizer with the MMSE-LE, MMSE-DFE, SFE and the SDFE discussed in *Chapter 3*, here named MMSE-LE-TE, MMSE-DFE-TE, SFE-TE and the SDFE-TE respectively. The performance of the DBN-TE is also compared to that of a few other LCTEs in the literature.

4.6.2.1 DBN-TE vs MMSE-LE/DFE-TE

The DBN-TE is evaluated against the MMSE-LE-TE and the MMSE-DFE-TE in [58], where BPSK modulated information is transmitted through a static multipath channel of length $L = 3$ with a CIR of $\mathbf{h} = \{0.408, 0.815, 0.408\}$. A rate $R_c = 1/2$, constraint length $K = 3$ recursive systematic convolutional encoder with generator $\mathbf{G} = [7, 5]$ is used, and the encoded information is randomly interleaved in blocks of length $N_c = 512$ before transmission. The MMSE-LE-TE and the MMSE-DFE-TE are iterated until full convergence is achieved, and the DBN-TE is iterated $Z = 10$ times, first using the Cholesky decomposition (Chol) to concentrate the energy in the first CIR tap, and secondly using the MMSE-DF prefilter (MMSE). Fig. 4.48 shows the performance of the turbo equalizers, with Cholesky decomposition filtering and MMSE-DF prefiltering. It is clear that the DBN-TE performance is comparable to that of the MMSE-LE-TE when Cholesky decomposition filtering is used, and outperforms the MMSE-LE-TE when MMSE-DF prefiltering is used.

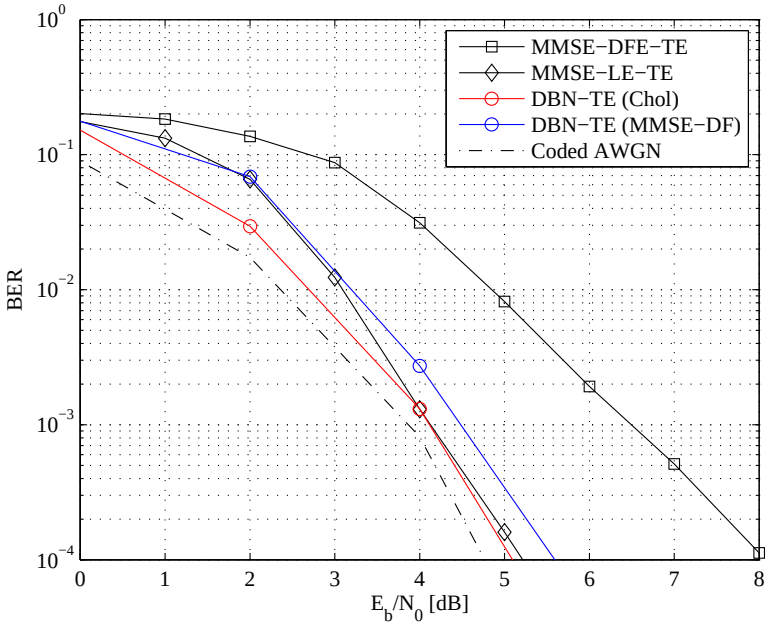


Figure 4.48: DBN-TE performance compared to that of MMSE-LE-TE and MMSE-DFE-TE in [58].

4.6.2.2 DBN-TE vs BDFE-TE and iBDFE-TE

The DBN-TE performance is also compared to that of the turbo equalizer using low complexity SISO block decision feedback equalizer (BDFE) proposed in [52]. The uncoded information is encoded using a rate $R_c = 1/2$ convolutional encoder with generators $\mathbf{G} = [23, 57]$ and with constraint length $K = 5$. Blocks of $N_c = 2048$ coded information is randomly interleaved and transmitted through a fading channel with a CIR according the typical urban channel profile in [59]. The channel is constant for each data block and varies between data blocks. For each data block the channel energy is normalized as in [52]. Fig. 4.49 shows the performance of the DBN-TE compared to that of the BDFE-TE-TE and the MMSE-LE-TE for $Z = 1$, $Z = 2$ and $Z = 5$ iterations. It is clear from Fig. 4.49 that the performance of the DBN-TE is comparable to that of the MMSE-LE-TE as well as the BDFE-TE in [52].

Fig. 4.50 shows the performance of a turbo equalizer using an improved BDFE-TE proposed in [53] (iBDFE-TE) for the identical simulation parameters as in Fig. 4.49. The only difference is that the generators are $\mathbf{G} = [23, 57]$ in the simulation setup in [53], and that QPSK modulation is used instead of BPSK. Fig. 4.50 also shows the performances of the DBN-TE for BPSK modulation using the same parameters as in [53]. It is clear from Fig. 4.50 that the performance of the DBN-TE is slightly

better than that of the improved DBFE proposed in [53].

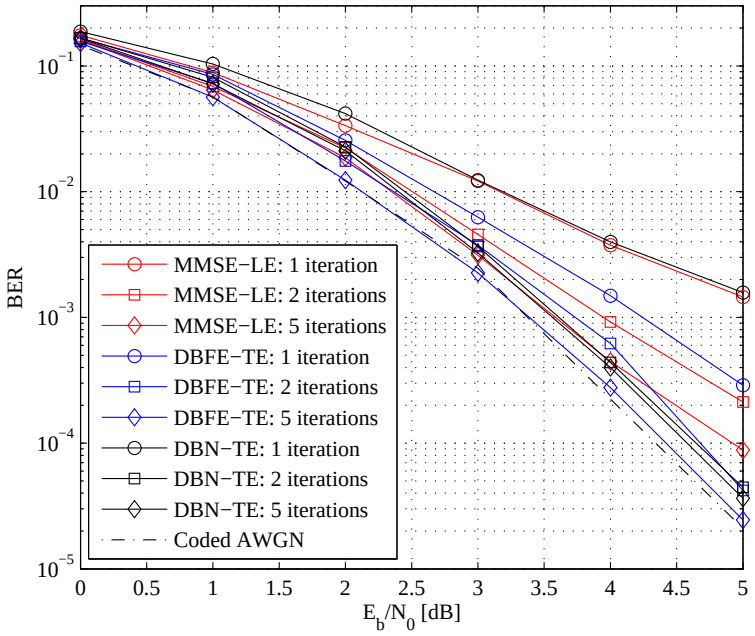


Figure 4.49: DBN-TE performance compared to that of MMSE-LE-TE [58] and BDFE-TE [52].

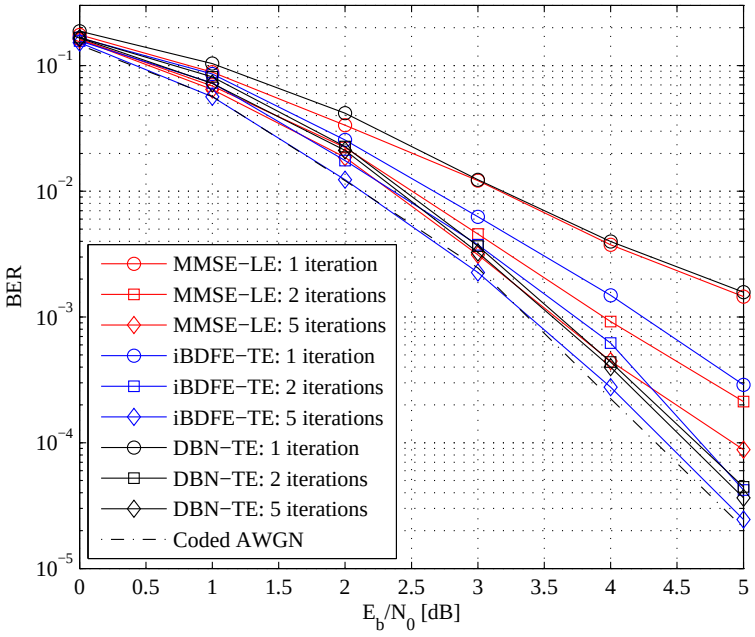


Figure 4.50: DBN-TE performance compared to that of MMSE-LE-TE and improved iBDFE-TE [53] .

4.6.2.3 DBN-TE vs MMSE-LE-TE and PDA-TE

The performance of the DBN-TE is compared to that of the MMSE-LE-TE and another LCTE, the probabilistic data association turbo equalizer (PDA-TE), where the MAP equalizer is replaced with a low complexity PDA equalizer proposed in [60]. The system encodes information using a rate $R_c = 1/2$, constraint length $K = 3$, convolutional encoder with generator $\mathbf{G} = [7,5]$ to encode uncoded blocks of length $N_u = 256$, and is then interleaved with a random interleaver. The information is transmitted through a channel $\mathbf{h} = \{0.408, 0.815, 0.408\}$ of length $L = 3$. To evaluate the performance of the DBN-TE, Cholesky decomposition filtering was used. Fig. 4.51 shows the performance of the DBN-TE compared to that of the MMSE-LE-TE and the PDA-TE for $Z = 0$, $Z = 1$ and $Z = 2$ iterations. The DBN-TE performance is also shown for $Z = 3$ iterations. It is clear that the performance of the DBN-TE is comparable to that of the MMSE-LE-TE and the PDA-TE at $Z = 2$ iterations.

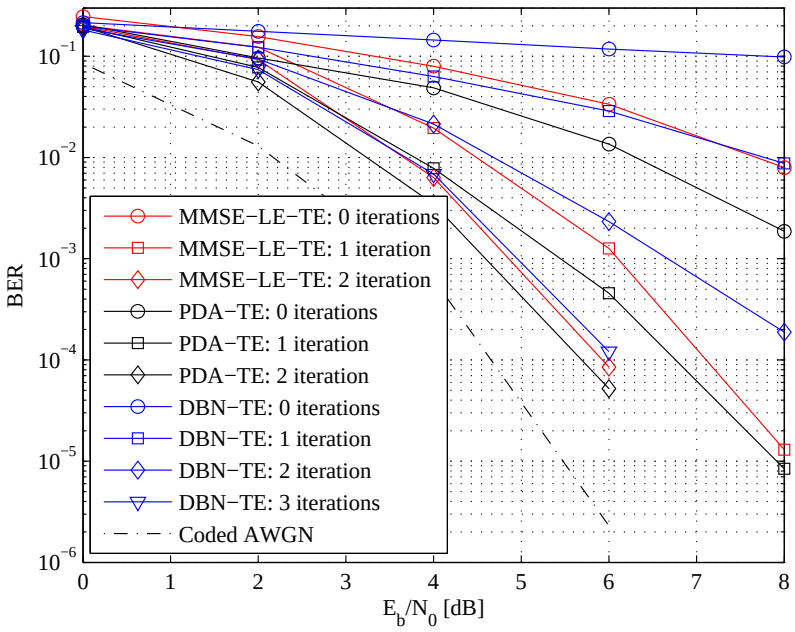


Figure 4.51: DBN-TE performance compared to that of MMSE-LE-TE in [58] and PDA-TE [60].

4.6.2.4 DBN-TE vs SISO-DFE-TE and SISO Bi-DFE-TE

The performance of the DBN-TE is compared to that of the SISO DFE-TE and the SISO bi-directional DFE-TE (Bi-DFE-TE) proposed in [61]. The information is encoded using a rate $R_c = 1/2$ recursive systematic convolutional encoder with constraint length $K = 3$, where the generators $\mathbf{G} = [7,5]$ are

used. Coded data blocks of length $N_c = 2048$ are randomly interleaved and transmitted through a channel $\mathbf{h} = \{0.2294, 0.4588, 0.6882, 0.4588, 0.2294\}$ of length $L = 5$. As before, the MMSE-DF prefilter is used for the DBN-TE. Fig. 4.53 shows the performance of the DBN-TE compared to that of the SISO-DFE-TE and the BiD-DFE-TE, from which it is clear that the DBN-TE performs better at low E_b/N_0 levels, but does not achieve the same asymptotic performance as the SISO-DFE-TE and the BiD-DFE-TE. The performance of the DBN-TE is still acceptable.

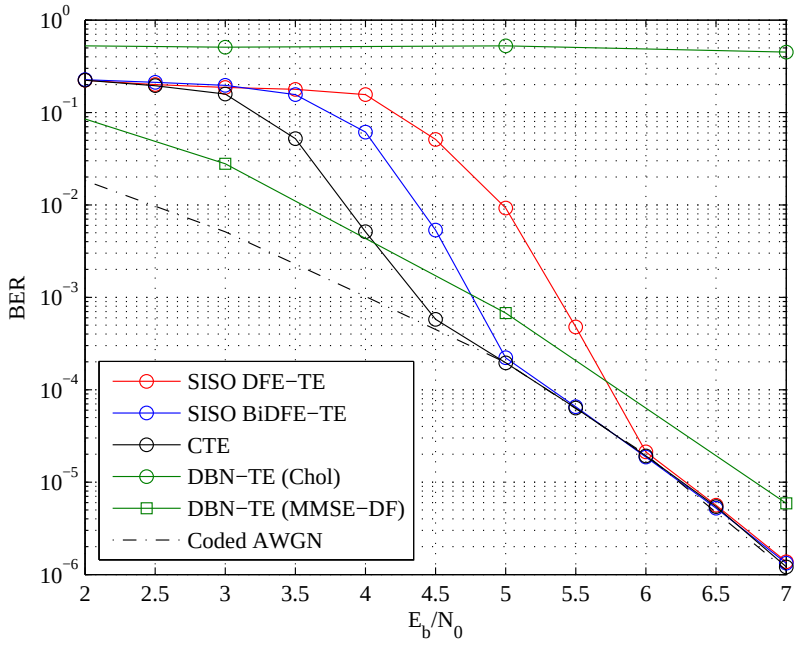


Figure 4.52: DBN-TE performance compared to that of Bi-DFE-TE and SISO-DFE-TE [61].

4.6.2.5 DBN-TE vs SISO-DFE-TE and SFE-TE

The DBN-TE's performance is compared to that of the SISO-DFE-TE developed in [62] and the SFE-TE in [10]. The information is encoded using a rate $R_c = 1/2$ convolutional encoder with constraint length $K = 3$, where once again the generators $\mathbf{G} = [7, 5]$ are used. The coded data block length is $N_c = 10560$ and a random interleaver is used. The coded information is transmitted through a channel $\mathbf{h} = \{0.2294, 0.4588, 0.6882, 0.4588, 0.2294\}$. As before, the MMSE-DF prefilter is used for the DBN-TE. Fig. 4.53 shows the performance of the DBN-TE compared to that of the SISO-DFE-TE and SFE-TE. As in Fig. 4.53, the DBN-TE performs comparably against the SISO-DFE-TE and the SFE-TE, with better performance at low E_b/N_0 levels and worse performance at higher E_b/N_0 levels.

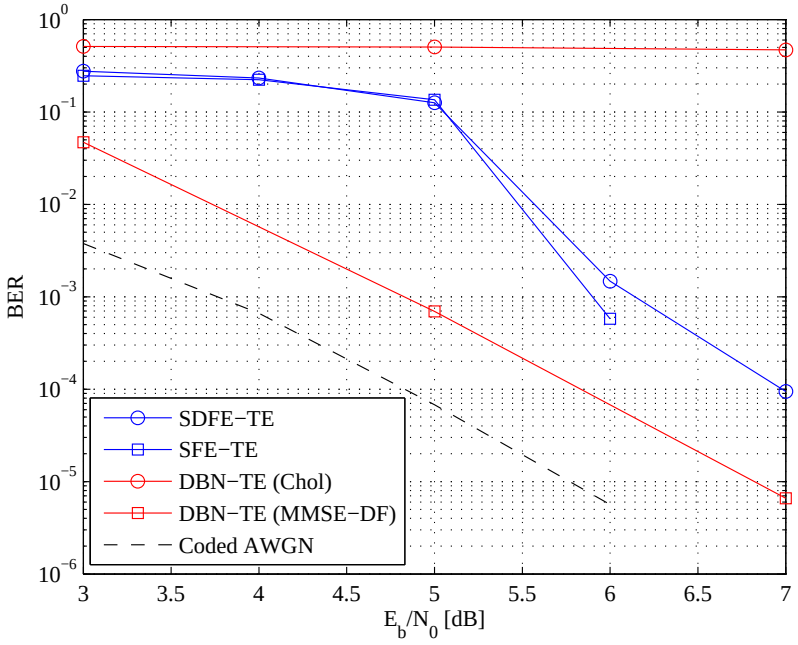


Figure 4.53: DBN-TE performance compared to that of SISO-DFE-TE [62] and SFE [10].

4.7 CONCLUDING REMARKS

In this chapter a turbo equalizer modeled on a dynamic Bayesian network was presented. The resulting turbo equalizer was named the DBN-TE, which uses BP via the forward-backward algorithm together with a soft-feedback mechanism to jointly equalize and decode the received information in order to estimate the uncoded information symbols. The DBN-TE works on the basis of approximate inference on a quasi-DAG, with dominant connections between the observed variables and their corresponding hidden variables as well as weak connections to past and future state variables. A minimum phase filter in the form of a Cholesky decomposition filter or a MMSE-DF prefilter is applied in order to concentrate the energy in the leading CIR tap, so as to ensure a dominant connection between the observed variables and their corresponding hidden variables. In order to restore the one-to-one relationship between the coded transmitted symbols and the leading CIR tap, the effect of the interleaver is eliminated by a transform on the interleaved channel matrix and a corresponding filtering of the received symbol sequence.

It was shown that the performance of the new DBN-TE closely matches that of the CTE in fading channels, with and without perfect CSI knowledge, and in some cases the DBN-TE outperforms the

CTE. It was also demonstrated that the DBN-TE performs favorably compared to other LCTEs in the literature, in static and fading channels, although at higher computational cost. The computational complexity of the DBN-TE is approximately quadratic in the coded data block length (worst case is cubic), exponential in the encoder constraint length, and approximately independent of channel memory length. Although the complexity of the DBN-TE is higher than that of the MMSE and DFE based turbo equalizers, it is still superior to that of the CTE for systems with long memory.

CHAPTER 5

HOPFIELD NEURAL NETWORK TURBO EQUALIZER

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 [16], and 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 primitive 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 [16].

Because of their ability to perform classification and pattern recognition based on partial information, as well as their low complexity processing capabilities, neural networks have also been used in designing wireless communication receiver algorithms. Neural networks have been used for equalization [63–66], multiuser detection in code division multiple access (CDMA) systems [67–71] and decoding [72, 73]. Of particular interest in this thesis is the HNN, a recurrent or feedback neural network, which has also been used in the aforementioned applications [19, 21, 22, 74–86].

In [19, 20] the author of this thesis proposed an MLSE equalizer which is able to equalize M-QAM modulated signals in systems with extremely long memory. The complexity of the equalizer proposed in [19, 20] is approximately quadratic in the data block length[1] and approximately independent of the

[1] It is explained how the complexity of matrix multiplication can be reduced via more efficient algorithms.

channel memory length. Its superior computational complexity is due to the high parallelism of its underlying neural network structure. It uses the HNN structure, which enables fast parallel processing of information between neurons, producing ML sequence estimates at the output. It was shown in [19,20] that the performance of the HNN MLSE equalizer closely matches that of the Viterbi MLSE equalizer in short channels, and near-optimally recombines the energy spread across the channel in order to achieve near-matched filter performance when the channel is extremely long.

The HNN has also been shown by several authors to be able to decode balanced check codes [21,22]. These codes, together with methods for encoding and decoding, were first proposed in [87], but it was later shown in [21, 22] that single codeword decoding can also be performed using the HNN. The ability of the HNN to detect binary patterns allows it to determine the ML codeword from a predefined set of codewords. In this chapter it is shown that the HNN ML decoder can be extended to allow for the ML estimation of a *sequence* of balanced check codes. It is therefore extendable to an MLSE decoder.

In this chapter a novel turbo equalizer is developed by combining the HNN MLSE equalizer developed in [19, 20] and an HNN MLSE decoder (used to decode balanced codes), resulting in the Hopfield Neural Network Turbo Equalizer (HNN-TE), which can be used as replacement for a CTE in systems with extremely long memory, where the coded symbols are interleaved before transmission through the multipath channel. The HNN-TE is able to equalize and decode balanced codes in systems with extremely long memory, since the computational complexity is nearly independent of the channel memory length. Like the HNN MLSE equalizer in [19, 20], its superior complexity characteristics are due to the high parallelism of its underlying neural network structure.

5.1 THE HOPFIELD NEURAL NETWORK

The 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 [88]. This circuit is shown in Fig. 5.1. 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

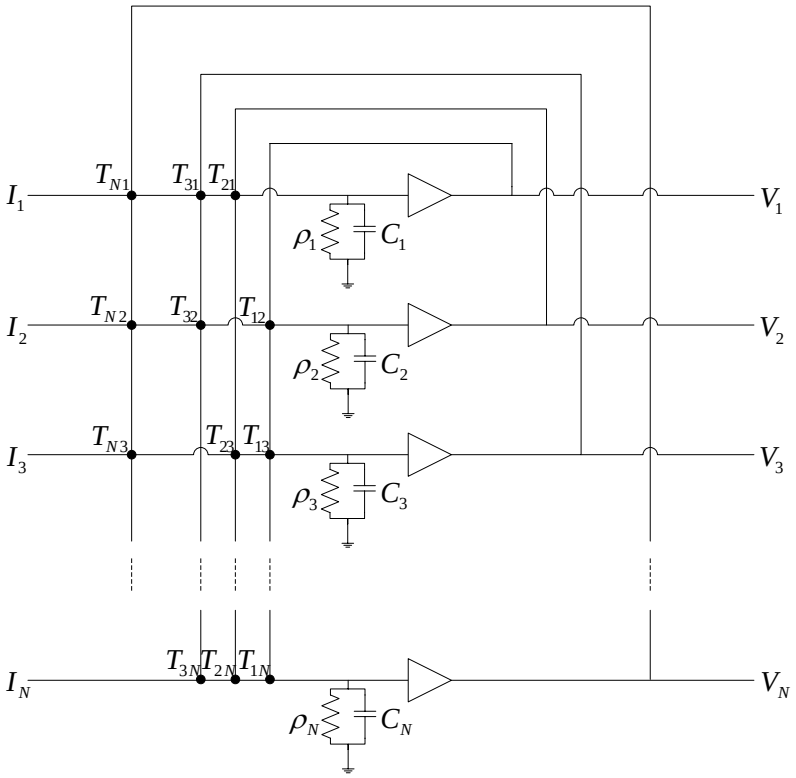


Figure 5.1: Hopfield neural network circuit diagram.

the dynamics of the system is given by [88]

$$\begin{aligned} C_i \frac{du_i}{dt} &= -\frac{u_i}{\tau_i} + \sum_j T_{ij} I_i + I_i \\ V_i &= g(u_i) \end{aligned} \tag{5.1}$$

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[2] represents the negative part of the activation function $g(u)$. The activation function is shown in Fig. 5.2. It was shown in [88] that the stable state of this circuit network can be found by minimizing the function

$$\mathscr{L} = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} T_{ij} V_i V_j - \sum_{i=1}^{N} V_i I_i \tag{5.2}$$

[2]Negative outputs are not shown here.

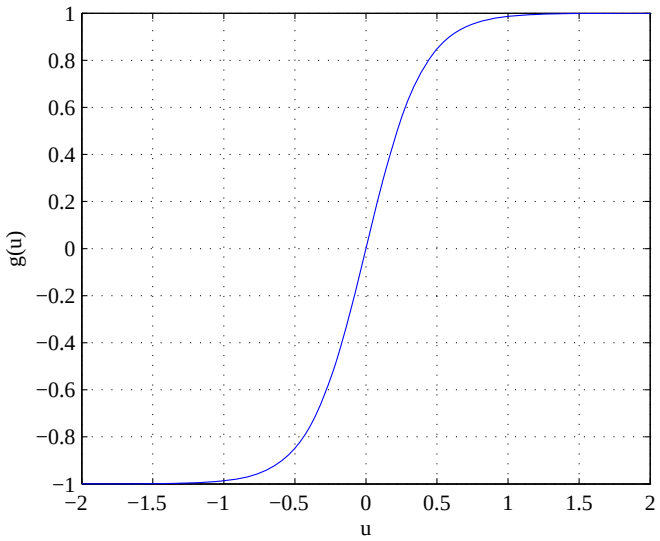


Figure 5.2: Activation function.

provided that $T_{ij} = T_{ji}$ and $T_{ii} = 0$, implying that T is symmetric around the diagonal and its diagonal is zero [88]. 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 [88]. 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 (5.2) the system equations in (5.1) are solved iteratively until the outputs V_1-V_N settle.

Hopfield also showed that this kind of network can be used to solve the traveling salesman problem (TSP). This problem is of a class called NP-complete, the class of nondeterministic 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 [56]. 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.

The HNN was first proposed in [88] and it was shown in [89] that the HNN can be used to solve combinatorial optimization problems as well as pattern recognition problems. In [88] Tank and Hopfield derived an energy function and showed how the HNN can be used to minimize this energy function,

thus producing near-ML sequence estimates at the output of the neurons. To enable the HNN to solve an optimization problem, the cost function of that problem is mapped to the HNN energy function, whereafter the HNN iteratively minimizes its energy function and performs near-MLSE. Also, to enable the HNN to solve a binary pattern recognition problem, the autocorrelation matrix of the set of patterns is used as the weights between the HNN neurons, while the noisy pattern to be recognized is used as the input to the HNN. Again, the HNN iteratively performs pattern recognition in order to produce the near-ML pattern at the output of the HNN.

5.1.1 Energy Function

The Hopfield energy function can be written as [19, 20, 88]

$$\mathscr{L} = -\frac{1}{2}\mathbf{s}^T\mathbf{X}\mathbf{s} - \mathbf{I}^T\mathbf{s}, \tag{5.3}$$

where $\mathbf{I}$ is a column vector with N elements, $\mathbf{X}$ is an $N \times N$ matrix. Assuming that $\mathbf{s}$, $\mathbf{I}$ and $\mathbf{X}$ contain complex values, these variables can be written as [19, 20, 88]

$$\begin{aligned} \mathbf{s} &= \mathbf{s}_i + j\mathbf{s}_q, \\ \mathbf{I} &= \mathbf{I}_i + j\mathbf{I}_q, \\ \mathbf{X} &= \mathbf{X}_i + j\mathbf{X}_q, \end{aligned} \tag{5.4}$$

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 the cross-correlation matrix of the complex received symbols such that

$$\mathbf{X}^H = \mathbf{X}_i^T - j\mathbf{X}_q^T = \mathbf{X}_i + j\mathbf{X}_q, \tag{5.5}$$

implying that it is Hermitian. Therefore $\mathbf{X}_i^T = \mathbf{X}_i$ is symmetric and $\mathbf{X}_q^T = -\mathbf{X}_q$ is skew symmetric [19, 20]. By using the symmetric properties of $\mathbf{X}_i$ and $\mathbf{X}_q$, (5.3) can be expanded and rewritten as

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

which in turn can be rewritten as [19, 20]

$$\mathscr{L} = -\frac{1}{2}[\mathbf{s}_i^T|\mathbf{s}_q^T]\begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i \end{bmatrix}\begin{bmatrix} \mathbf{s}_i \\ \mathbf{s}_q \end{bmatrix} - [\mathbf{I}_i^T|\mathbf{I}_q^T]\begin{bmatrix} \mathbf{s}_i \\ \mathbf{s}_q \end{bmatrix}. \tag{5.6}$$

It is clear that (5.6) is in the form of (5.3), where the variables in (5.3) are substituted as follows:

$$\begin{aligned} \mathbf{s}^T &= [\mathbf{s}_i^T|\mathbf{s}_q^T], \\ \mathbf{I}^T &= [\mathbf{I}_i^T|\mathbf{I}_q^T], \\ \mathbf{X} &= \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i \end{bmatrix}. \end{aligned} \tag{5.7}$$

Equation (5.6) is used to derive the HNN MLSE equalizer, HNN decoder, and eventually the HNN-TE.

5.1.2 Iterative System

The HNN minimizes the energy function (5.3) with the following iterative system:

$$\begin{aligned} \mathbf{u}^{(i)} &= \mathbf{X}\mathbf{s}^{(i)} + \mathbf{I} \\ \mathbf{s}^{(i+1)} &= g\left(\beta(i)\mathbf{u}^{(i)}\right), \end{aligned} \tag{5.8}$$

where $\mathbf{u} = \{u_1, u_2, \cdots, u_N\}^T$ is the internal state of the HNN, $\mathbf{s} = \{s_1, s_2, \cdots, s_N\}^T$ is the vector of estimated symbols, $g(.)$ is the decision function associated with each neuron and i indicates the iteration number. $\beta(.)$ is a function used for optimization as in [19, 20].

The estimated symbol vector $\mathbf{s}^T = [\mathbf{s}_i^T|\mathbf{s}_q^T]$ is updated with each iteration. $\mathbf{I}^T = [\mathbf{I}_i^T|\mathbf{I}_q^T]$ contains the best blind estimate of $\mathbf{s}$ via energy recombination, and is therefore used as input to the network, while $\mathbf{X} = \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i \end{bmatrix}$ contains the cross-correlation information of the transmitted symbols. The system produces the MLSE estimates in $\mathbf{s}$ after Z iterations.

5.2 HOPFIELD NEURAL NETWORK TURBO EQUALIZER

Turbo Equalizers are used in multipath communication systems that make use of encoders, usually convolutional encoders, to encode the source symbol sequence $\mathbf{s}$ of length N_u (using some generator matrix $\mathbf{G}$) at a rate R_c to produce coded information symbols $\mathbf{c}$ of length $N_c = N_u/R_c$, after which the coded symbols $\mathbf{c}$ are interleaved with a random interleaver before modulation and transmission. The interleaved coded symbols $\acute{\mathbf{c}}$ are transmitted through a multipath channel with a CIR length of L, causing inter-symbol interference among adjacent transmitted symbols at the receiver. At the receiver the received inter-symbol ISI corrupted coded symbols are matched filtered and used as input to the

turbo equalizer. The received symbol sequence is given by

$$\mathbf{r} = \mathbf{H}\acute{\mathbf{c}} + \mathbf{n}, \tag{5.9}$$

where $\mathbf{n}$ is a vector containing complex Gaussian noise samples and $\acute{\mathbf{c}}$ is the interleaved coded symbols given by

$$\acute{\mathbf{c}} = \mathbf{J}\mathbf{G}^T\mathbf{s}, \tag{5.10}$$

where $\mathbf{J}$ is an $N_c \times N_c$ interleaver matrix, and $\mathbf{H}$ is the $N_c \times N_c$ channel matrix

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & h_0 & \dots & 0 & 0 & 0 & 0 \\ h_{L-1} & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & h_{L-1} & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & h_0 & 0 & 0 \\ 0 & 0 & 0 & h_{L-1} & \dots & h_0 & 0 \\ 0 & 0 & 0 & 0 & h_{L-1} & \dots & h_0 \end{bmatrix}. \tag{5.11}$$

In this section the derivation of the HNN-TE is discussed, by first deriving its constituent parts - the HNN MLSE equalizer and the HNN MLSE decoder - and then showing how the HNN-TE is finally realized by combining the two.

5.2.1 HNN MLSE Equalizer

The HNN MLSE equalizer was developed by the author in [19, 20]. The HNN MLSE equalizer was applied to single-carrier M-QAM modulated systems with extremely long memory, where the CIR length was as long as $L = 250$, even though this was not the limit. The ability of the HNN MLSE equalizer to equalize signals in systems with highly dispersive channels is due to the fact that its complexity grows quadratically with an increase in transmitted data block size, and that it is approximately independent of the channel memory length. The HNN MLSE equalizer developed in [19, 20] will subsequently be presented, without spending time on the derivation.

It was shown in [19, 20] that the correlation matrices $\mathbf{X}_i$ and $\mathbf{X}_q$ in (5.7), for a single carrier system transmitting a data block of length N through a multipath channel of length L with the data block initiated and terminated by $L-1$ known tail symbols, with values 1 for BPSK modulation and $\frac{1}{\sqrt{2}} +$

$j\frac{1}{\sqrt{2}}$ for M-QAM modulation, can be determined by

$$\mathbf{X}_i = -\begin{bmatrix} 0 & \alpha_1 & \dots & \alpha_{L-1} & \dots & 0 \\ \alpha_1 & 0 & \alpha_1 & \dots & \ddots & \vdots \\ \vdots & \alpha_1 & 0 & \ddots & \vdots & \alpha_{L-1} \\ \alpha_{L-1} & \vdots & \ddots & \ddots & \alpha_1 & \vdots \\ \vdots & \ddots & \dots & \alpha_1 & 0 & \alpha_1 \\ 0 & \ddots & \alpha_{L-1} & \dots & \alpha_1 & 0 \end{bmatrix} \tag{5.12}$$

and

$$\mathbf{X}_q = -\begin{bmatrix} 0 & \gamma_1 & \dots & \gamma_{L-1} & \dots & 0 \\ \gamma_1 & 0 & \gamma_1 & \dots & \ddots & \vdots \\ \vdots & \gamma_1 & 0 & \ddots & \vdots & \gamma_{L-1} \\ \gamma_{L-1} & \vdots & \ddots & \ddots & \gamma_1 & \vdots \\ \vdots & \ddots & \dots & \gamma_1 & 0 & \gamma_1 \\ 0 & \ddots & \gamma_{L-1} & \dots & \gamma_1 & 0 \end{bmatrix} \tag{5.13}$$

where $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_{L-1}\}$ and $\gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_{L-1}\}$ are respectively determined by

$$\alpha_k = \sum_{j=0}^{L-k-1} h_j^{(i)} h_{j+k}^{(i)} + \sum_{j=0}^{L-k-1} h_j^{(q)} h_{j+k}^{(q)}, \tag{5.14}$$

and

$$\gamma_k = \sum_{j=0}^{L-k-1} h_j^{(q)} h_{j+k}^{(i)} - \sum_{j=0}^{L-k-1} h_j^{(i)} h_{j+k}^{(q)}, \tag{5.15}$$

where $k = 1, 2, 3, ..., L-1$ and i and q denote the in-phase and quadrature components of the CIR coefficients.

Upon inspection it is easy to see from (5.12) through to (5.15) that $\mathbf{X}_i$ and $\mathbf{X}_q$ can be determined using the respective in-phase and quadrature components of the $N \times N$ channel matrix, with the in-phase and quadrature components of the CIR, $\mathbf{h}^{(i)} = \{h_0^{(i)}, h_1^{(i)}, \dots, h_{L-1}^{(i)}\}^T$ and $\mathbf{h}^{(q)} = \{h_0^{(q)}, h_1^{(q)}, \dots, h_{L-1}^{(q)}\}^T$, on the diagonals such that

$$\mathbf{H}^{(i)} = \begin{bmatrix} h_0^{(i)} & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & h_0^{(i)} & \dots & 0 & 0 & 0 & 0 \\ h_{L-1}^{(i)} & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & h_{L-1}^{(i)} & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & h_0^{(i)} & 0 & 0 \\ 0 & 0 & 0 & h_{L-1}^{(i)} & \dots & h_0^{(i)} & 0 \\ 0 & 0 & 0 & 0 & h_{L-1}^{(i)} & \dots & h_0^{(i)} \end{bmatrix} \tag{5.16}$$

and

$$\mathbf{H}^{(q)} = \begin{bmatrix} h_0^{(q)} & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & h_0^{(q)} & \dots & 0 & 0 & 0 & 0 \\ h_{L-1}^{(q)} & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & h_{L-1}^{(q)} & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & h_0^{(q)} & 0 & 0 \\ 0 & 0 & 0 & h_{L-1}^{(q)} & \dots & h_0^{(q)} & 0 \\ 0 & 0 & 0 & 0 & h_{L-1}^{(q)} & \dots & h_0^{(q)} \end{bmatrix}. \tag{5.17}$$

Using $\mathbf{H}^{(i)}$ and $\mathbf{H}^{(q)}$ the correlation matrices in (5.12) and (5.13) can be determined by

$$\mathbf{X}_i = -\left(\mathbf{H}^{(i)T}\mathbf{H}^{(i)} + \mathbf{H}^{(q)T}\mathbf{H}^{(q)}\right) \tag{5.18}$$

which is simply

$$\mathbf{X}_i = -Re\{\mathbf{H}^T\mathbf{H}\}. \tag{5.19}$$

Also

$$\mathbf{X}_q = -\left(\mathbf{H}^{(q)T}\mathbf{H}^{(i)} - \mathbf{H}^{(i)T}\mathbf{H}^{(q)}\right)^T, \tag{5.20}$$

which is

$$\mathbf{X}_q = -Im\{\mathbf{H}^T\mathbf{H}\}. \tag{5.21}$$

$\mathbf{X}_i$ and $\mathbf{X}_q$ are then used to construct the combined correlation matrix in (5.7).

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i \end{bmatrix}. \tag{5.22}$$

It was also shown in [19,20] that the input vectors $\mathbf{I}_i$ and $\mathbf{I}_q$ in (5.7) are determined by

$$\mathbf{I}_i = \begin{bmatrix} \lambda_1 - \rho(\alpha_1 + \gamma_1 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \lambda_2 - \rho(\alpha_2 + \gamma_2 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \lambda_3 - \rho(\alpha_3 + \gamma_3 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \vdots \quad \vdots \quad \vdots \\ \lambda_{L-1} - \rho(\alpha_{L-1} + \gamma_{L-1}) \\ \lambda_L \\ \vdots \quad \vdots \quad \vdots \\ \lambda_{N-L+1} \\ \lambda_{N-L+2} - \rho(\alpha_{L-1} - \gamma_{L-1}) \\ \vdots \quad \vdots \quad \vdots \\ \lambda_{N-2} - \rho(\alpha_3 - \gamma_3 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \lambda_{N-1} - \rho(\alpha_2 - \gamma_2 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \lambda_N - \rho(\alpha_1 - \gamma_1 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \end{bmatrix} \tag{5.23}$$

and

$$\mathbf{I}_q = \begin{bmatrix} \omega_1 - \rho(\alpha_1 - \gamma_1 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \omega_2 - \rho(\alpha_2 - \gamma_2 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \omega_3 - \rho(\alpha_3 - \gamma_3 + \ldots + \alpha_{L-1} - \gamma_{L-1}) \\ \vdots \quad \vdots \quad \vdots \\ \omega_{L-1} - \rho(\alpha_{L-1} - \gamma_{L-1}) \\ \omega_L \\ \vdots \quad \vdots \quad \vdots \\ \omega_{N-L+1} \\ \omega_{N-L+2} - \rho(\alpha_{L-1} + \gamma_{L-1}) \\ \vdots \quad \vdots \quad \vdots \\ \omega_{N-2} - \rho(\alpha_3 + \gamma_3 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \omega_{N-1} - \rho(\alpha_2 + \gamma_2 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \\ \omega_N - \rho(\alpha_1 + \gamma_1 + \ldots + \alpha_{L-1} + \gamma_{L-1}) \end{bmatrix}, \tag{5.24}$$

where $\rho = 1/\sqrt{2}$ for M-QAM modulation, $\rho = 1$ in $\mathbf{I}_i$ and $\rho = 0$ in $\mathbf{I}_q$ for BPSK modulation, and $\mathbf{\Lambda} = \{\lambda_1, \lambda_2, \cdots, \lambda_N\}$ is determined by

$$\lambda_k = \sum_{j=0}^{L-1} r_{j+k}^{(i)} h_j^{(i)} + \sum_{j=0}^{L-1} r_{j+k}^{(q)} h_j^{(q)}, \tag{5.25}$$

and $\mathbf{\Omega} = \{\omega_1, \omega_2, \cdots, \omega_N\}$ is determined by

$$\omega_k = \sum_{j=0}^{L-1} r_{j+k}^{(q)} h_j^{(i)} - \sum_{j=0}^{L-1} r_{j+k}^{(i)} h_j^{(q)}, \tag{5.26}$$

where $k = 1, 2, 3, ..., N$ with i and q again denoting the in-phase and quadrature components of the respective elements. The combined input vector in (5.7) is therefore constructed as

$$\mathbf{I} = \begin{bmatrix} \mathbf{I}_i \\ \hline \mathbf{I}_q \end{bmatrix}. \tag{5.27}$$

Note that $\mathbf{\Lambda}$ and $\mathbf{\Omega}$ can easily be determined by

$$\mathbf{\Lambda} = \mathbf{H}^{(i)T}\mathbf{r}^{(i)} + \mathbf{H}^{(q)T}\mathbf{r}^{(q)}, \tag{5.28}$$

and

$$\mathbf{\Omega} = \mathbf{H}^{(i)T}\mathbf{r}^{(q)} - \mathbf{H}^{(q)T}\mathbf{r}^{(i)}, \tag{5.29}$$

where $\mathbf{r}^{(i)}$ and $\mathbf{r}^{(q)}$ are the respective in-phase and quadrature components of the received symbols $\mathbf{r} = \{r_1, r_2, \cdots, r_{N+L-1}\}^T$.

By deriving the cross-correlation matrix $\mathbf{X}$ and the input vector $\mathbf{I}$ in (5.7), the model in (5.6) is complete, and the iterative system in (5.8) can be used to equalize M-QAM modulated symbols transmitted through a channel with large CIR lengths. The HNN MLSE equalizer was evaluated in [19, 20] for BPSK and 16-QAM with performance reaching the matched-filter bound in extremely long channels.

5.2.2 HNN MLSE Decoder

The HNN has been shown to be able to decode balanced codes [21, 22]. A binary word of length m can be called balanced if it contains exactly $m/2$ ones and $m/2$ zeros [87]. In addition, balanced codes have the property that no codeword is contained in another word, simply meaning that the positions of ones in one codeword will never be a subset of the positions of the ones in another codeword [87].

The encoding process is described in [87] where the first k bits of the uncoded word is flipped in order to ensure the resulting codeword is "balanced," whereafter the position k is appended to the balanced codeword before transmission. This encoding process is not followed here, as the set of $m = 2^n$ balanced codewords are determined beforehand, after which encoding is performed by mapping a

set of n bits to 2^n balanced BPSK symbols, or by mapping a set of $2n$ bits to 2^n balanced 4-QAM symbols.

The HNN decoder developed here uses the set of predetermined codewords to determine the connection weights describing the level of connection between the neurons. It has previously been shown how an HNN can be used to decode one balanced code at a time, but the HNN MLSE decoder this author derives here is able to simultaneously decode any number of concatenated codewords in order to provide the ML transmitted sequence of codewords. After the HNN MLSE decoding, the ML BPSK or 4-QAM codewords of length 2^n are demapped to n bits (or $2n$ bits for 4-QAM), which completes the decoding process.

5.2.2.1 Codeword Selection:

The author has found that Walsh-Hadamard codes, widely used in CDMA systems [2], are desirable codes for this application, in view of their seeming balance and orthoganality characteristics. Walsh-Hadamard codes are linear codes that map n bits to 2^n codewords, where each set of codewords has a Hamming distance of 2^{n-1} and a Hamming weight of 2^{n-1}.

Walsh-Hadamard codes are not "balanced" as described above. The first codeword is always all-ones, while subsets of some codewords are contained in others, violating both restrictions for balance. Instead of using the complete set of Walsh-Hadamard codes to map n bits to 2^n codewords, a subset of codes in the Walsh-Hadamard matrix is selected, duplicated and modified so as to construct a new set of 2^n codewords of length 2^n. Consider the set of length $2^n = 8$ Walsh-Hadamard codes

$$\mathbf{H}_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}. \tag{5.30}$$

To construct a set of balanced codewords from $\mathbf{H}_8$, a subset of 2^{n-1} codewords is selected, which are used as the first 2^{n-1} codewords in the new set of codewords. The second set of 2^{n-1} codewords is constructed as follows:

1. Reverse the order in which the first 2^{n-1} codewords appear in the new set.

2. Flip the bits of the reversed set of 2^{n-1} codewords.

Assuming the subset selected from $\mathbf{H}_8$ above is the set $\mathbf{H}_{8,4:7}$ (implying that codewords in rows 4 through to 7 are selected), the resulting set of 2^n balanced codewords is

$$\mathbf{C}_8 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}. \tag{5.31}$$

It is clear that $\mathbf{C}_8$ is balanced in the sense that the rows (codewords) as well as the columns are balanced. It has been found that the HNN decoder performs better if the rows as well as the columns are balanced. The Hamming weight of $\mathbf{C}_8$ is still $2^{n-1} = 2^2$, while the Hamming distance increases to slightly larger than $2^{n-1} = 2^2$.

By following the steps described above, any set of Walsh-Hadamard codes of length 2^n can be used to create a new set of 2^n balanced codes of length $m = 2^n$.

5.2.2.2 Encoding

Encoding is performed by mapping a group of n bits to 2^n BPSK symbols, or a group of $2n$ bits to 2^n 4-QAM symbols. Before encoding, the set of codewords $\mathbf{C}_{2^n}$ derived from the set of Walsh-Hadamard codes $\mathbf{H}_{2^n}$ is made bipolar by converting the 0's to -1.

5.2.2.2.1 BPSK encoding When BPSK modulation is used, n bits are mapped to 2^n BPSK symbols. The n bits are used to determine an index k in the range 1 - 2^n, which is then used to select a codeword from the set of codewords in $\mathbf{C}_{2^n}$ such that the selected codeword $\mathbf{c} = \mathbf{C}_{2^n}(k)$. Table 5.1 shows the number of uncoded bits, codeword length, uncoded bit to coded symbol rate R_s and the uncoded bit to coded bit rate R_c (code rate) for different n.

Table 5.1: Input-output relationship for the BPSK encoder

n	2^n	R_s	R_c
1	2	1/2	1/2
2	4	1/2	1/2
3	8	3/8	3/8
4	16	1/4	1/4

5.2.2.2.2 4-QAM encoding When 4-QAM modulation is used, $2n$ bits are mapped to 2^n 4-QAM symbols. The first and second groups of n bits (out of $2n$ bits) are used to determine two indices, $k^{(i)}$ and $k^{(q)}$, in the range 1 - 2^n, one for the in-phase part, and the other for the quaternary part of the codeword. The first index $k^{(i)}$ selects a codeword from $\mathbf{C}_{2^n}^{(i)}$, where $\mathbf{C}_{2^n}^{(i)}$ is derived as before, and the second index $k^{(q)}$ selects a codeword from $\mathbf{C}_{2^n}^{(q)}$, which can be equal to $\mathbf{C}_{2^n}^{(i)}$ or can be uniquely determined as explained earlier. The 4-QAM "codeword" is then calculated as $\mathbf{c} = \mathbf{C}_{2^n}^{(i)}(k^{(i)}) + j\mathbf{C}_{2^n}^{(q)}(k^{(q)})$, which is much like the result of coded modulation where groups of coded bits (in this case uncoded bits) are mapped to signal constellation points to improve spectral efficiency [2]. Table 5.2 shows the number of uncoded bits, codeword length, the uncoded bit to coded symbol rate R_s and code rate R_c for different $2n$. Even though the code rate remains the same as with BPSK modulation, the throughput doubles as expected.

Table 5.2: Input-output relationship for the 4-QAM encoder

$2n$	2^n	R_s	R_c
2	2	1	1/2
4	4	1	1/2
6	8	3/4	3/8
8	16	1/2	1/4

5.2.2.3 Decoding

The HNN is known to be able to recognize input patterns from a set of stored patterns [88, 89]. In the context of the HNN decoder, the patterns are the balanced codewords, and the HNN is able to determine the ML codeword from a set of codewords. This has been demonstrated before [21], but only for one codeword at a time. Therefore, if a received data block contains P codewords, the HNN will have to be applied P times in order to determine P ML codewords. The HNN MLSE decoder developed here is able to determine the most likely sequence of codewords using a single HNN. The HNN MLSE decoder is therefore applied once to a received data block containing any number of codewords.

After the HNN MLSE decoder has determined the sequence of most likely transmitted codewords, the codewords are demapped by calculating the Euclidean distance between each ML codeword and each codeword in $\mathbf{C}_{2^n}$ for BPSK modulation, and each codeword in $\mathbf{C}_{2^n}^{(i)} + j\mathbf{C}_{2^n}^{(q)}$ for 4-QAM modulation. The indices(s) corresponding to the codeword(s) that have the lowest Euclidean distance/distances is/are converted to bits, which completes the decoding phase.

The derivation of the HNN MLSE decoder entails the calculation of the cross-correlation matrices $\mathbf{X}_i$ and $\mathbf{X}_q$, and the input vectors $\mathbf{I}_i$ and $\mathbf{I}_q$ in (5.7). The HNN MLSE decoder is first derived for the decoding of a single codeword, after which it will be extended to enable the decoding of any number of codewords simultaneously. Derivations are performed for 4-QAM only, since the BPSK HNN MLSE decoder is a simplification of its 4-QAM counterpart.

5.2.2.3.1 Single codeword decoding: To enable the HNN to store a set of codewords, the average correlation between all patterns must be stored in the weights between the neurons. According to Hebb's rule of auto-associative memory [90], the connection weight matrix, or correlation matrix, is calculated by taking the cross-correlation of the patterns to be stored. Since one is working with complex symbols, there are two weight matrices to be calculated. The cross-correlation matrices in (5.6) are calculated as

$$\begin{aligned}\mathbf{X}_i &= Re\{\mathbf{C}^T\mathbf{C}\} \\ &= \mathbf{C}_{2^n}^{(i)T}\mathbf{C}_{2^n}^{(i)} + \mathbf{C}_{2^n}^{(q)T}\mathbf{C}_{2^n}^{(q)}\end{aligned} \tag{5.32}$$

and

$$\begin{aligned}\mathbf{X}_q &= Im\{\mathbf{C}^T\mathbf{C}\} \\ &= \mathbf{C}_{2^n}^{(q)T}\mathbf{C}_{2^n}^{(i)} - \mathbf{C}_{2^n}^{(i)T}\mathbf{C}_{2^n}^{(q)},\end{aligned} \tag{5.33}$$

where $\mathbf{C} = \mathbf{C}_{2^n}^{(i)} + j\mathbf{C}_{2^n}^{(q)}$, and $\mathbf{C}_{2^n}^{(i)}$ and $\mathbf{C}_{2^n}^{(q)}$ are the matrices containing the generated codewords as before, respectively used for the in-phase and quadrature components of the codeword. Note the similarities between the correlation matrices in (5.32) and (5.33) and those in (5.18) and (5.20). Also, the two input vectors are simply the real and imaginary components of the noise-corrupted received codeword, such that

$$\mathbf{I}_i = Re\{\mathbf{c}\} + Re\{\mathbf{n}\} \tag{5.34}$$

and

$$\mathbf{I}_q = Im\{\mathbf{c}\} + Im\{\mathbf{n}\} \tag{5.35}$$

where $\mathbf{c}$ is of length 2^n and $\mathbf{n}$ is a vector containing complex samples from the distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu = 0$ and σ is the noise standard deviation. After the ML codeword has been detected, each detected codeword (of length 2^n) can be mapped back to n bits for BPSK modulation and $2n$ bits for 4-QAM modulation.

5.2.2.3.2 Multiple codeword decoding: It was shown how the HNN can be used to decode single codewords, but the HNN decoder can be extended in order to detect ML transmitted sequences of codewords. This step is crucial in the quest for merging the HNN decoder with the HNN MLSE equalizer, since the HNN MLSE equalizer detects ML sequences of transmitted symbols. If the transmitted information is encoded, these sequences contain multiple codewords, and hence the HNN decoder must be extended to detect not only single codewords, but codeword sequences.

This extension is easily achieved by using the HNN parameters already derived in (5.32) through (5.35). Consider a system transmitting a sequence of P balanced codewords of length 2^n, where n is the length of the uncoded bit-words. The new correlation matrix is constructed by copying $\mathbf{X}$ in (5.7) along the diagonal according to the number of transmitted codewords P, such that

$$\mathbf{X}^{(P)} = \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T & & & & & & \\ \mathbf{X}_q & \mathbf{X}_i & & & & & & \\ & & \mathbf{X}_i & \mathbf{X}_q^T & & & \emptyset & \\ & & \mathbf{X}_q & \mathbf{X}_i & & & & \\ & & & & \ddots & \ddots & & \\ & \emptyset & & & \ddots & \ddots & & \\ & & & & & & \mathbf{X}_i & \mathbf{X}_q^T \\ & & & & & & \mathbf{X}_q & \mathbf{X}_i \end{bmatrix}, \tag{5.36}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{X}_i & \mathbf{X}_q^T \\ \mathbf{X}_q & \mathbf{X}_i \end{bmatrix}$ is repeated on the diagonal P times and $\emptyset$ implies that the rest of $\mathbf{X}^{(P)}$ is empty, containing only 0's.

Also, the input vector $\mathbf{I}$ in (5.7), consisting of $\mathbf{I}_i$ and $\mathbf{I}_q$, is extended according to the number of transmitted codewords P such that

$$\mathbf{I}^{(P)} = \begin{bmatrix} \mathbf{I}_i^{(P)} \\ \hline \mathbf{I}_q^{(P)} \end{bmatrix}, \tag{5.37}$$

where

$$\mathbf{I}_i^{(P)} = [Re\{\mathbf{c}_1\}, Re\{\mathbf{c}_2\}, \ldots, Re\{\mathbf{c}_p\}]^T + Re\{\mathbf{n}\} \tag{5.38}$$

and

$$\mathbf{I}_q^{(P)} = [Im\{\mathbf{c}_1\}, Im\{\mathbf{c}_2\}, \ldots, Im\{\mathbf{c}_p\}]^T + Im\{\mathbf{n}\}, \tag{5.39}$$

where $\mathbf{c}_p$ is the pth codeword of length 2^n, where $p = 1, 2, \ldots, P$, and $\mathbf{n}$ is of length $2^n P$ and contains complex samples from the distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu = 0$ and σ is the noise standard deviation.

The extended cross-correlation matrix and input vector in (5.36) and (5.37) can now be used to estimate the ML sequence of transmitted codewords, after which each detected codeword (of length 2^n) can be mapped back to n bits for BPSK modulation and $2n$ bits for 4-QAM modulation.

5.2.3 Merging the HNN MLSE Equalizer and the HNN MLSE Decoder

The HNN-TE is an amalgamation of the HNN MLSE equalizer and the HNN MLSE decoder, which were discussed in the previous sections. In this section it is explained how the HNN MLSE equalizer and the HNN MLSE decoder are combined in order to perform iterative joint equalization and decoding (Turbo Equalization) using a single HNN structure. The HNN-TE is able to jointly equalize and decode BPSK and 4-QAM coded modulated signals in systems with highly dispersive multipath channels, with extremely low computational complexity compared to traditional Turbo Equalizers, which employ a MAP equalizer/decoder pair.

5.2.3.1 System Model

Since complete models for the HNN MLSE equalizer and decoder are now available, the combination of the two is fairly straightforward. In order to distinguish between equalizer and decoder parameters, a number of redefinitions are in order. For the HNN MLSE equalizer the correlation matrix and input vector relating to (5.7), as derived in (5.22) and (5.27), are now $\mathbf{X}_E$ and $\mathbf{I}_E$ repectively and will henceforth be called "equalizer correlation matrix" and "equalizer input vector". Similarly the HNN MLSE decoder correlation matrix and input vector relating to (5.7), as derived in (5.36) and (5.37), are now $\mathbf{X}_D$ and $\mathbf{I}_D$ respectively and will henceforth be called "decoder correlation matrix" and "decoder input vector".

When a coded data block of length N_c is transmitted through a multipath channel, $\mathbf{X}_E$ and $\mathbf{X}_D$ are determined according to (5.22) and (5.36), where both matrices are of size $N_c \times N_c$. Since the function of the equalizer and the decoder has to be merged, it makes sense to combine $\mathbf{X}_E$ and $\mathbf{X}_D$ somehow to enable the equalizer to perform decoding, or to enable the decoder to perform equalization. This combination is performed by first normalizing $\mathbf{X}_D$ with respect to $\mathbf{X}_E$, because of varying energy in a multipath fading channel between received data blocks. $\mathbf{X}_D$ is therefore normalized with respect to $\mathbf{X}_E$ such that

$$\mathbf{X}_D^{(norm)} = \left(\frac{||\mathbf{X}_E||}{||\mathbf{X}_D||} \right) \mathbf{X}_D. \tag{5.40}$$

Next the new correlation matrix is determined as

$$\mathbf{X}_{TE} = \mathbf{X}_E + \mathbf{X}_D^{(norm)}. \tag{5.41}$$

The rationale behind the addition of the equalizer correlation matrix and the normalized decoder correlation matrix is that the connection weights in the decoder correlation matrix should bias those

of the equalizer correlation matrix. Since $\mathbf{X}_{TE}$ contains $\mathbf{X}_E$ offset by $\mathbf{X}_D^{(norm)}$, joint equalization and decoding is made possible.

The new input vector also needs to be calculated. $\mathbf{I_D}$ contains the noise-corrupted coded symbols, while $\mathbf{I_E}$ contains not only received coded symbol information, but also the ISI information. Note that when there is no multipath or fading ($L = 1$ and $h_0 = 1$), $\mathbf{I}_E$ reduces to $\mathbf{I}_D$. The new input vector used in the HNN-TE is therefore only

$$\mathbf{I}_{TE} = \mathbf{I}_E. \tag{5.42}$$

With the new correlation matrix $\mathbf{X}_{TE}$ and input vector $\mathbf{I}_{TE}$, the HNN-TE model is complete, and the iterative system in (5.8) can be used to equalize and decode the received coded information jointly.

5.2.3.2 Interleaver Mitigation

Upon reception the received symbol vector has to be deinterleaved to restore the one-to-one relationship between each element in $\mathbf{r}$ and $\mathbf{c}$ with respect to the first coefficient h_0 of the CIR $\mathbf{h} = \{h_0, h_1, \cdots, h_{L-1}\}^T$. Deinterleaving $\mathbf{r}$ transforms the transmission model in (5.9).[3] Substituting (5.10) in (5.9) and applying the deinterleaver, which is simply the transpose of the interleaver matrix $\mathbf{J}$, gives

$$\mathbf{J}^T\mathbf{r} = \mathbf{J}^T\mathbf{H}\mathbf{J}\mathbf{G}^T\mathbf{s} + \mathbf{J}^T\mathbf{n}, \tag{5.43}$$

which is equivalent to transmitting the coded symbol sequence $\mathbf{c} = \mathbf{G}^T\mathbf{s}$ through a channel

$$\mathbf{Q} = \mathbf{J}^T\mathbf{H}\mathbf{J}. \tag{5.44}$$

Therefore (5.43) can be written as

$$\mathbf{J}^T\mathbf{r} = \mathbf{Q}\mathbf{G}^T\mathbf{s} + \mathbf{J}^T\mathbf{n}. \tag{5.45}$$

Consequently the new channel matrix $\mathbf{Q}$, rather than the conventional channel matrix $\mathbf{H}$ in (5.11), is used in the calculation of the equalizer correlation matrix $\mathbf{X}_E$ derived in (5.22). Owing to the above transformation, $\mathbf{Q}$ does not contain the CIR $\mathbf{h}$ on the diagonal as in $\mathbf{H}$. Rather, each column in $\mathbf{Q}$ (of length N_c) contains a different random combination of all CIR coefficients (where the rest of the $N_c - L$ elements in a column are equal to 0), dictated by the randomization effect exhibited in $\mathbf{Q}$ because of the random interleaver. This randomization effect results from first multiplying the channel $\mathbf{H}$ with the interleaving matrix $\mathbf{J}$ and then deinterleaving by multiplying the result with $\mathbf{J}^T$

[3] Unlike the DBN-TE in *Chapter 3* the HNN-TE does not require the channel to be in minimum phase form.

(see (5.44)). Deinterleaving places the first CIR coefficient (h_0) on the diagonal of $\mathbf{Q}$, restoring the one-to-one relationship between each element in $\mathbf{r}$ and each corresponding coded transmitted symbol in $\mathbf{c}$. The computational complexity of generating $\mathbf{Q}$ in (5.44) is $O(N_c^3)$, where N_c is the coded data block length, and the complexity of deinterleaving $\mathbf{r}$ is negligible.

To illustrate this concept, consider the three-dimensional representations of $|\mathbf{HJ}|$ and $|\mathbf{Q}|$ in Fig. 5.3 to Fig. 5.8, for a hypothetical system transmitting coded information through a multipath channel with CIR lengths of $L = 1$, $L = 5$ and $L = 20$ respectively, with a block length $N_c = 80$. Fig. 5.3 and Fig. 5.4 show $|\mathbf{HJ}|$ and $|\mathbf{Q}|$ for channels of length $L = 1$, where $|\mathbf{HJ}|$ in Fig. 5.3 is clearly interleaved. It is also clear that the new channel $\mathbf{Q}$ in Fig. 5.4 is deinterleaved, since the first coefficient h_0 of the CIR has been restored to the diagonal of $\mathbf{Q}$. Fig. 5.5 and Fig. 5.7 show the interleaved channels for $L = 5$ and $L = 20$, where Fig. 5.6 and Fig. 5.8 show the new channels $\mathbf{Q}$, again with the first CIR coefficient h_0 restored to the diagonal. Even though h_0 is restored to the diagonal of $\mathbf{Q}$, it is clear that the rest of the CIR coefficients $h_1, h_2, \ldots, h_{L-1}$ are scattered throughout $\mathbf{Q}$. As stated before, each column in $\mathbf{Q}$ contains a different random combination of all CIR coefficients (with h_0 on the diagonal for each column), dictated by the randomization effect exhibited in $\mathbf{Q}$, where the rest of the $N_c - L$ elements in each column are equal to 0.

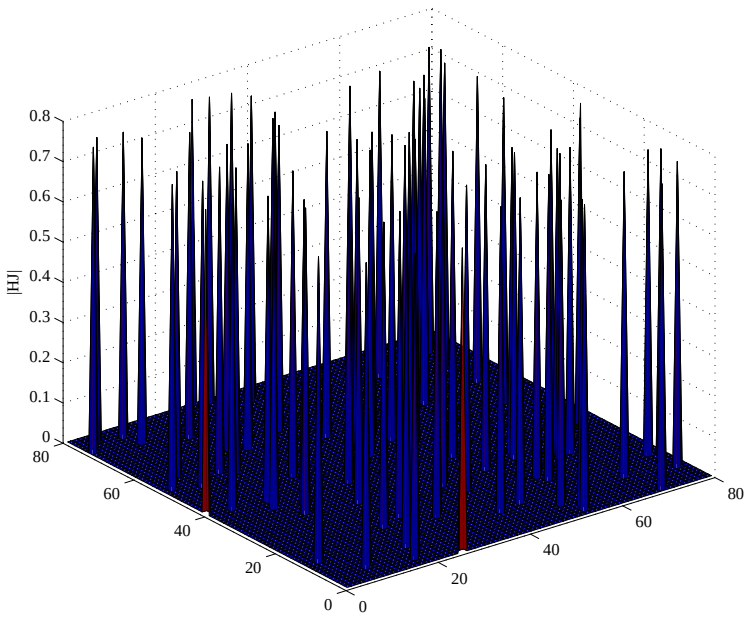


Figure 5.3: $|\mathbf{HJ}|$ for a system with $L = 1$ CIR coefficient.

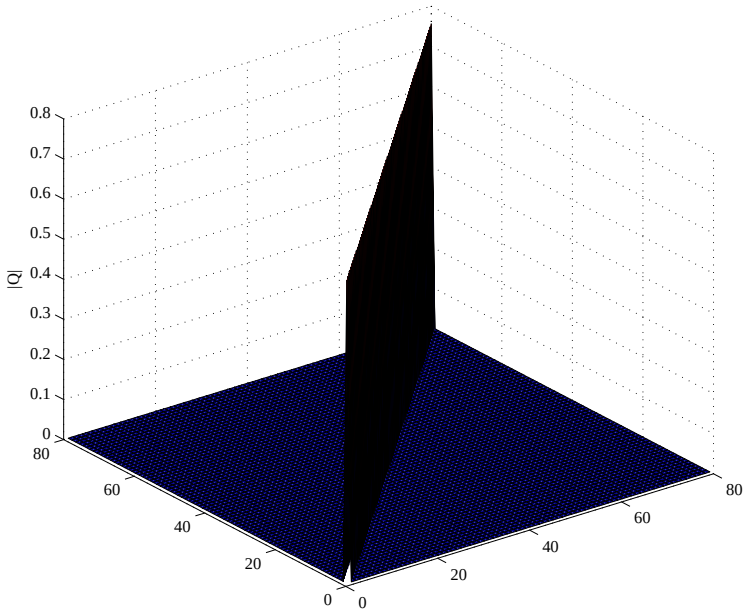


Figure 5.4: $|\mathbf{Q}|$ for a system with $L = 1$ CIR coefficient.

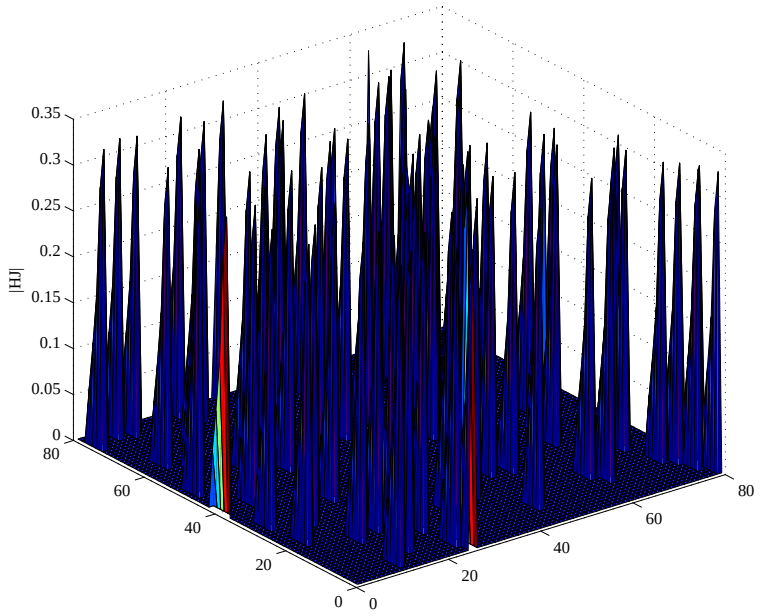
5.3 OPTIMIZATION

Optimization is necessary due to an inherent problem in neural networks.[4] When a solution space is large, the network sometimes struggles to converge to a "good," solution, leading to suboptimal performance. Because the HNN usually gets stuck in suboptimal local minima, it is necessary to employ optimization techniques [16, 91] to aid the HNN in escaping less optimal basins of attraction.

5.3.1 Simulated Annealing

Simulated annealing has its origin in metallurgy, where annealing is the process used to temper steel and glass by heating them to a high temperature and then gradually cooling them. This causes the material to coalesce into a low-energy crystalline state [16]. 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. Since 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,

[4]The optimization techniques implemented to improve the performance HNN-TE are identical to those used for the HHN MLSE equalizer in [19, 20]. The discussion in [20] is summarized here.



[h!]

Figure 5.5: $|\mathbf{HJ}|$ for a system with $L = 5$ CIR coefficients.

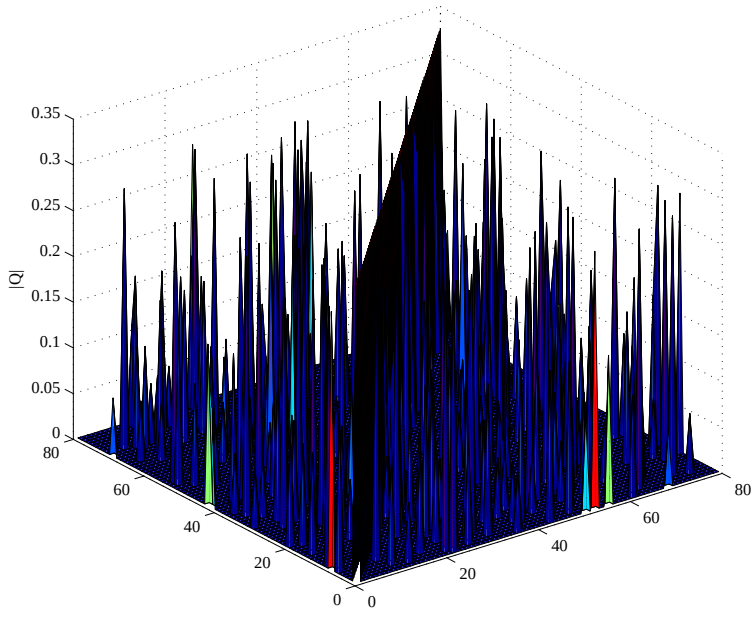


Figure 5.6: $|\mathbf{Q}|$ for a system with $L = 5$ CIR coefficients.

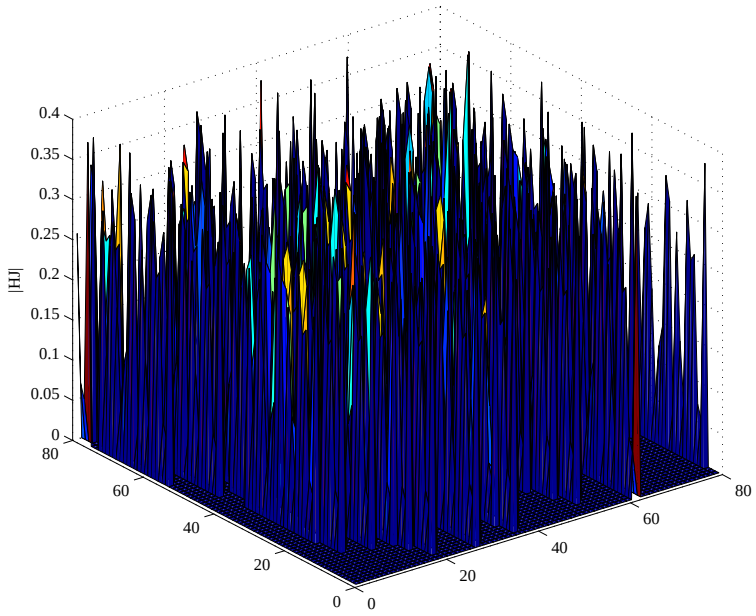


Figure 5.7: $|\mathbf{HJ}|$ for a system with $L = 20$ CIR coefficients.

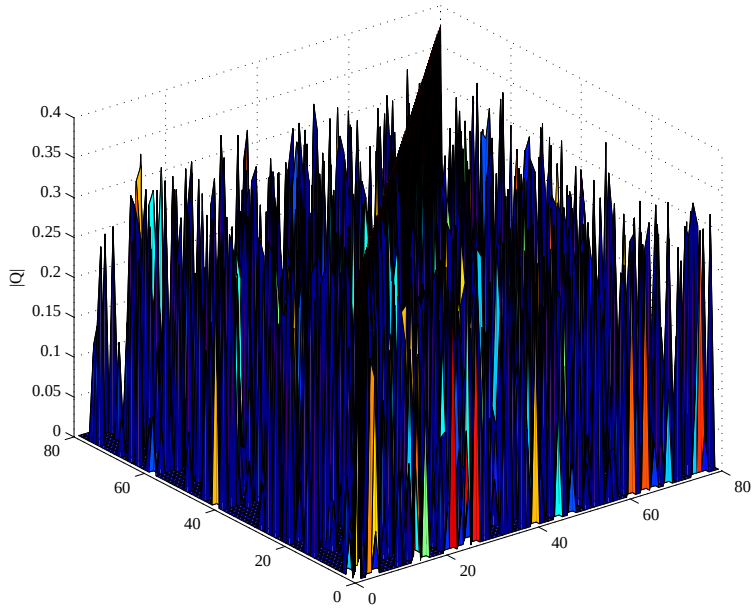


Figure 5.8: $|\mathbf{Q}|$ for a system with $L = 20$ CIR coefficients.

the network will gradually converge to the global, or near 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 (5.8) is used as a scaling factor in order to perform simulated annealing. As the system iterates, i is incremented, and $\beta(.)$ produces a value according to an exponential function. The output of this function starts at a near zero value for the first iterations, and converges to 1 for the last iteration Z. This function is given by

$$\beta(i-1) = 5^{\frac{2(i-Z)}{Z}}, \tag{5.46}$$

and shown in Fig. 5.9. This causes the output of $\beta(.)$ to start at a near-zero value and to converge exponentially to 1 with each iteration. The effect of annealing on the decision function during the iteration cycle is shown in Fig. 5.10, with the slope of the decision function increasing as $\beta(.)$ is updated with each iteration.

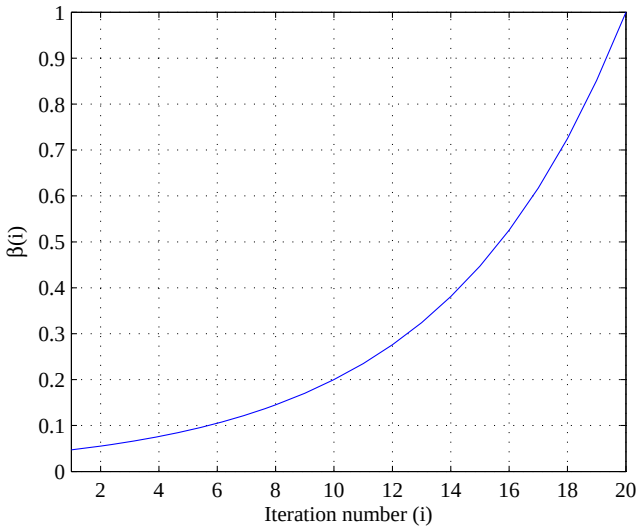


Figure 5.9: β-updates for $Z = 20$ iterations.

5.3.2 Asynchronous Updates

In artificial neural networks, the neurons in the network can be updated using either parallel or asynchronous updates. Consider the iterative solution of the HNN in (5.8). When *parallel* neuron updates are used, all the elements in $\mathbf{u}^{(n)}$ are calculated before the elements in $\mathbf{s}^{(n)}$ are determined. This implies that the output of the neurons will only be a function of the neuron outputs from the previous iteration. When using *asynchronous* neuron updates, however, one element in $\mathbf{s}^{(n)}$ is determined for

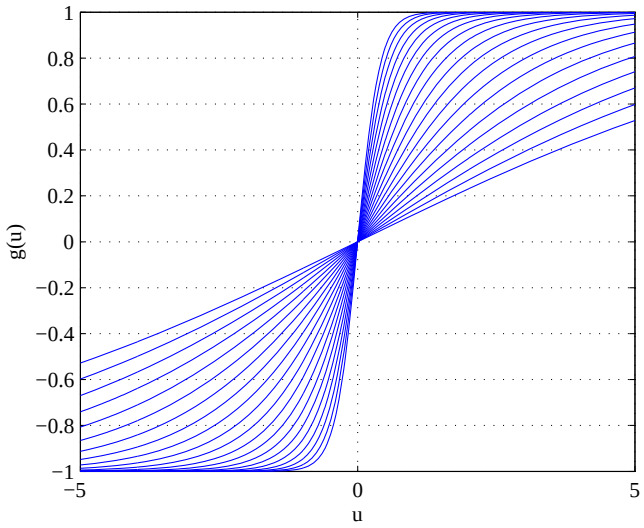


Figure 5.10: Simulated annealing on the bipolar decision function for $Z = 20$ iterations.

every corresponding element in $\mathbf{u}^{(n)}$. Asynchronous updates allow the changes of the neuron outputs to propagate to the other neurons immediately [91], while, with parallel updates, the output of all of the neurons will only be propagated to the other neurons after all of the neurons have been updated.

In the context of Turbo Equalization using superstructures the idea of asynchronous neuron updates in the HNN-TE is analogous to *dynamic LLR updates* used as an optimization technique for the DBN-TE in *Chapter 3* (see *4.3.6*). Dynamic LLR updates allow for coded symbol estimates to be used in the *current iteration* as soon as they become available. As soon as the forward and backward messages overlap, the corresponding LLR coded symbol estimates can be determined and used in the estimation of the remainder of the code symbol estimates. When dynamic LLR updates are not applied in the DBN-TE, new coded symbol estimates are only available after the forward-backward message has been determined completely, after which all the LLR updates are determined for use in the next iterations. Similarly, when parallel neuron updates are applied, new coded symbol estimates are only available at the end of each iteration, whereas coded symbol estimates can be used in the current iteration when asynchronous neuron updates are applied.

5.4 COMPUTATIONAL COMPLEXITY ANALYSIS

The computational complexity of the HNN-TE is compared to that of the CTE by calculating the number of computations performed for each received data block, for a fixed set of system parameters. The number of computations is normalized by the coded data block length so as to factor out the effect of the length of the transmitted data block, which allows the computational complexity to be expressed in terms of the number of computations required per received coded symbol. The complexity of the HNN-TE is cubically related to the coded data block length, so a change in N_c will still have an effect on the normalized computational complexity. The HNN-TE complexity is also compared to that of the LCTEs discussed in *Chapter 3*.

5.4.1 HNN-TE vs CTE

As discussed in *Chapter 3*, the complexity can be reduced when frequency hopping is employed, but for the HNN-TE that reduction is not significant. The complexity of the HNN-TE is therefore presented without frequency hopping. The computational complexity of the HNN-TE is

$$CC_{HNN-TE} \approx O(4(Nc+L-1)^{2.376} + Z_{HNN-TE}(N_cM/2)^2), \tag{5.47}$$

where N_c is the coded data block length, L is the CIR length, M is the modulation constellation alphabet size ($M = 2$ for BPSK and $M = 4$ for 4-QAM), Z_{HNN-TE} is the number of iterations and k is the codeword length, which was chosen as $k = 8$ for a code rate of $R_c = 3/8$. Since the cubic complexity of matrix inversion and multiplication can be reduced to $O(N^{2.376})$ [55], as mentioned in the previous chapter, (5.47) serves as a lower bound on the HNN-TE computational complexity.

The complexity of the CTE is

$$CC_{CTE} \approx O(Z_{CTE}(N_cM_eL + N_ck^2)), \tag{5.48}$$

where Z_{CTE} is the number of iterations and M_e is the number of equalizer states, determined by M^{L-1} ($M = 2$ for BPSK modulation and $M = 4$ for 4-QAM modulation).

Fig. 5.11 shows the normalized computational complexity of the HNN-TE and the CTE for coded data block lengths of $N_c = 80$, $N_c = 160$, $N_c = 320$, $N_c = 640$, $N_c = 1280$ and $N_c = 2560$, where $Z_{HNN-TE} = 25$ and $Z_{CTE} = 5$, for BPSK (red) and 4-QAM (green) modulation when $O(N_c^{2.376})$ matrix multiplication complexity is considered. For the CTE the BPSK (blue) and 4-QAM (black) complexities are also shown. Fig. 5.12 shows the same information as Fig. 5.11, but with $O(N_c^3)$ matrix

multiplication complexity. It is clear that the computational complexity of the HNN-TE increases with an increase in the coded data block length, but for realistic data block lengths the complexity of the HNN-TE is superior to that of the CTE for channels with long memory. The HNN-TE is computationally less complex for BSPK modulation than for 4-QAM. On the other hand, the complexity of the CTE grows exponentially with an increase in modulation order. From Fig. 5.11 it is clear that the complexity of the HNN-TE is almost quadratically related to the coded data block length and approximately independent of the channel memory length, which is more evident when L is increased. The normalized computational complexity of the HNN-TE and the CTE for $O(N_c^{2.376})$ and $O(N_c^3)$ matrix multiplication complexity) for $N_c = 1280$ using BPSK and 4-QAM for extremely long channels is shown in Fig. 5.13, where there is no comparison between the complexity of the HNN-TE and that of the CTE, for both BSPK and 4-QAM modulation.

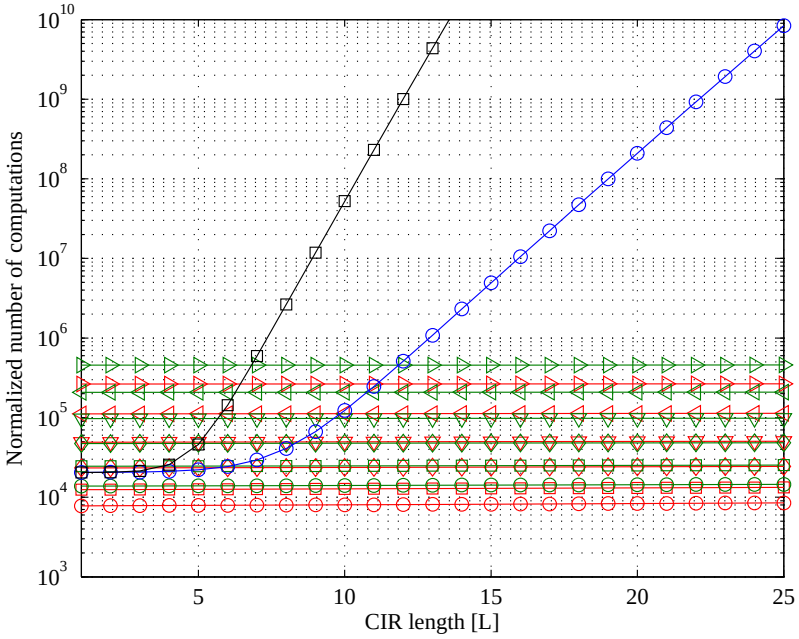


Figure 5.11: HNN-TE and CTE normalized computational complexity for short channels and varying coded block length with $O(N_c^{2.376})$ matrix multiplication complexity.

5.4.2 HNN-TE vs LCTEs

The computational complexity of the HNN-TE is compared to that of the LCTEs discussed in *Chapter 3* for BPSK modulation, using the same parameters as before, assuming the computational complexities of the MMSE-LE/DFE (in (3.62) and (3.64)) and the SFE/SDFE (in (3.66) and

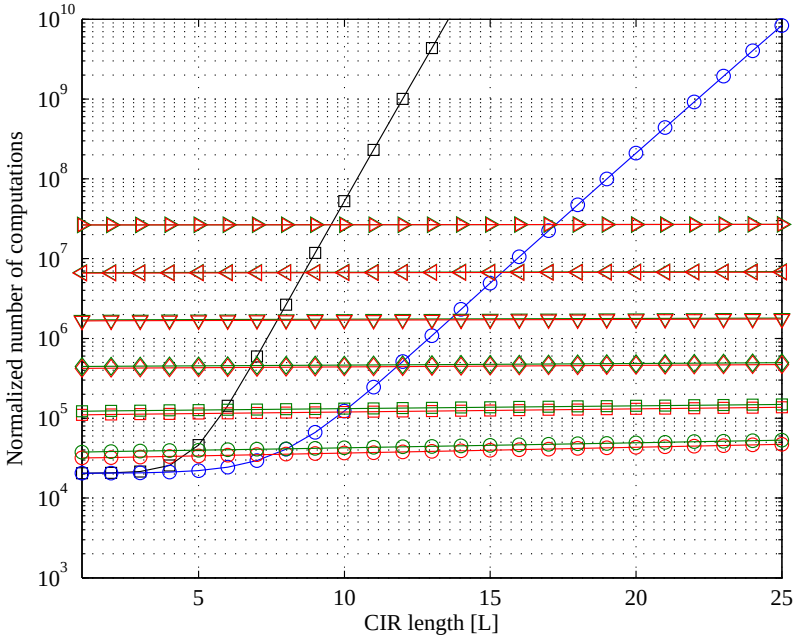


Figure 5.12: HNN-TE and CTE normalized computational complexity for short channels and varying coded block length $O(N_c^3)$ matrix multiplication complexity.

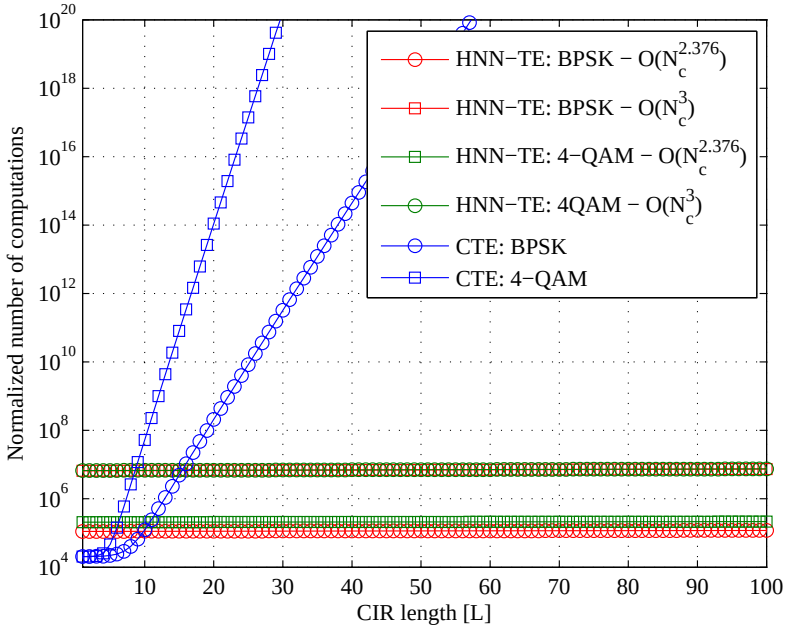


Figure 5.13: HNN-TE and CTE normalized computational complexity for long channels and $N_c = 1280$ with $O(N_c^{2.376})$ and $O(N_c^3)$ matrix multiplication complexity.

(3.68)). Assuming a MAP decoder computational complexity of $O(N_c k^2)$ (for the decoding of balanced codes), the resulting LCTE computational complexities are expressed as

$$CC_{MMSE-LE/DFE-TE} = O(Z(N_c(N^2+L^2)+N_c k^2)) \tag{5.49}$$

and

$$CC_{SFE/SDFE-TE} = O(Z(N_c(N+L)+N_c k^2)). \tag{5.50}$$

These computational complexities of the LCTEs (red) are shown together with those of the HNN-TE (for both $O(N_c^3)$ (purple) and $O(N_c^{2.376})$ (green) matrix multiplication complexity) in Fig. 5.14 for uncoded data block lengths of $N_u = 240$ (circle), $N_u = 480$ (square) and $N_u = 960$ (diamond). From Fig. 5.14 it can be seen that the HNN-TE complexity grows with an increase in the data block length, and it is clear that the HNN-TE is more computationally complex than both the MMSE-LE/DFE-TE and the SFE/SDFE-TE. However, when $O(N_c^{2.376})$ matrix multiplication complexity is considered, the complexity of the HNN-TE is comparable to that of the MMSE-LE/DFE-TE. Also, when compared to the CTE, the complexity of the HNN-TE is far superior when in highly dispersive channels.

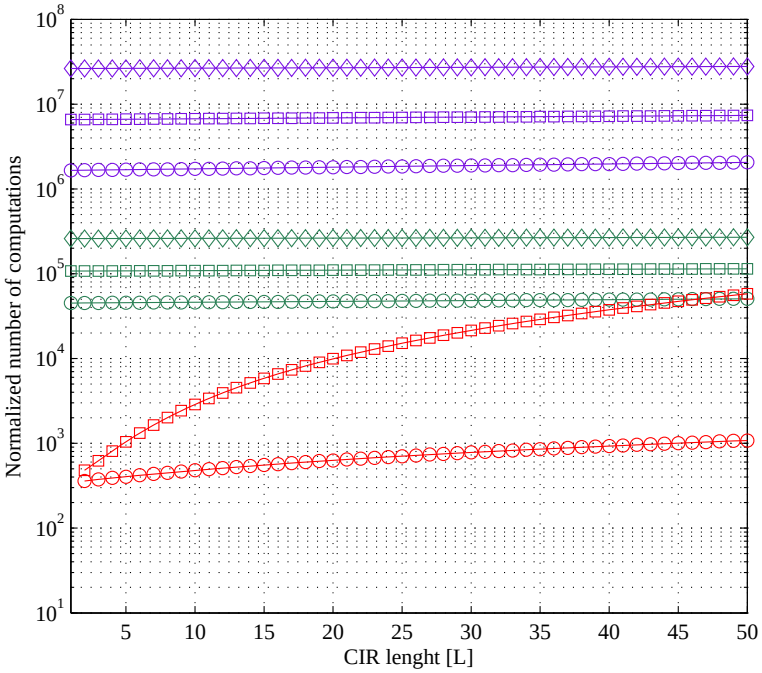


Figure 5.14: HNN-TE and LCTE normalized computational complexity for various uncoded data block lengths and BPSK modulation.

5.5 SIMULATION RESULTS

The HNN-TE is evaluated in a mobile fading environment for BPSK and 4-QAM modulation at a code rate of $R_c = n/k = 3/8$. The HNN-TE is simulated and compared for both BPSK and 4-QAM modulation, after which the performance of the HNN-TE is first evaluated alongside that of the CTE for BPSK modulation. The simulation environment is discussed at length in *Appendix A*. The following simulation parameter settings were used:

Parameter	**Setting**
Modulation	BPSK/4-QAM
Interleaver	Random
Uncoded block length (N_u)	480
Coded block length (N_c)	1280
Channel	Fading/Static
CIR length (L)	Varying
Channel Estimation	No
Channel State Information	Perfect CSI
Number of pilots	-
Frequency hopping	Yes
Number of freq. hops	4
Number of CTE iterations (Z_{CTE})	5
Number of HNN-TE iterations (Z_{HNN-TE})	$Z(E_b/N_0) = 2(5^{(E_b/N_0)/5})$
Mobile speed (v)	20 km/h
PDP	Uniform ($\mathbf{h}^T\mathbf{h} = 1$)

In order to evaluate the effect of each parameter on the performance of the HNN-TE, the following simulations were performed by fixing all the parameters and varying one parameter at a time.

5.5.1 HNN-TE: BPSK vs 4-QAM

5.5.1.1 Simulated Annealing

The effectiveness of the application of simulated annealing in the HNN-TE is evaluated by simulating the system with and without simulated annealing in static and fading channels using BPSK modula-

tion. Fig. 5.15 shows the performance of the HNN-TE in static channels of length $L = 4$ and $L = 6$, with and without simulated annealing. Although the performance of the HNN-TE is not very good in a static channel with equal tap weights, it is still much better when annealing is applied than without the use of annealing.

Fig. 5.16 shows the performance of the HNN-TE in fading channels of length $L = 5$, $L = 10$ and $L = 20$. Fig. 5.16 indicates that annealing vastly improves the performance of the HNN-TE. It can therefore be concluded that simulated annealing greatly improves the performance of the HNN-TE in both static and fading channels. Simulated annealing will henceforth be used in all simulations of the HNN-TE.

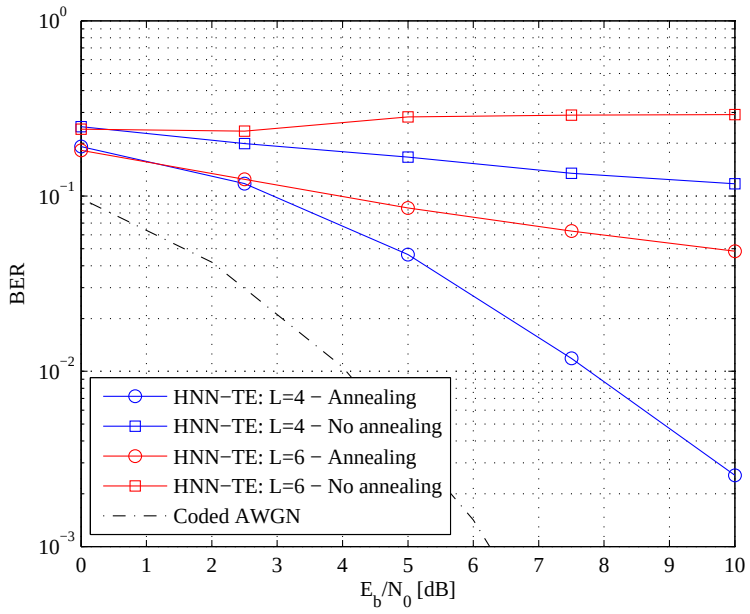


Figure 5.15: HNN-TE performance in short static channels with and without the use of simulated annealing.

5.5.1.2 Long Memory

Fig. 5.17 shows the performance of the HNN-TE for channels of length $L = 10$, $L = 20$, $L = 50$, $L = 100$ at a fixed mobile speed of 20 km/h for BPSK and 4-QAM modulation, assuming perfect CSI. It is clear that the performance for BPSK modulation is better than the performance for 4-QAM, which is due to the fact that Gray coding cannot be applied in the encoding process described in *Section 3.2.2*. The performance loss is therefore warranted.

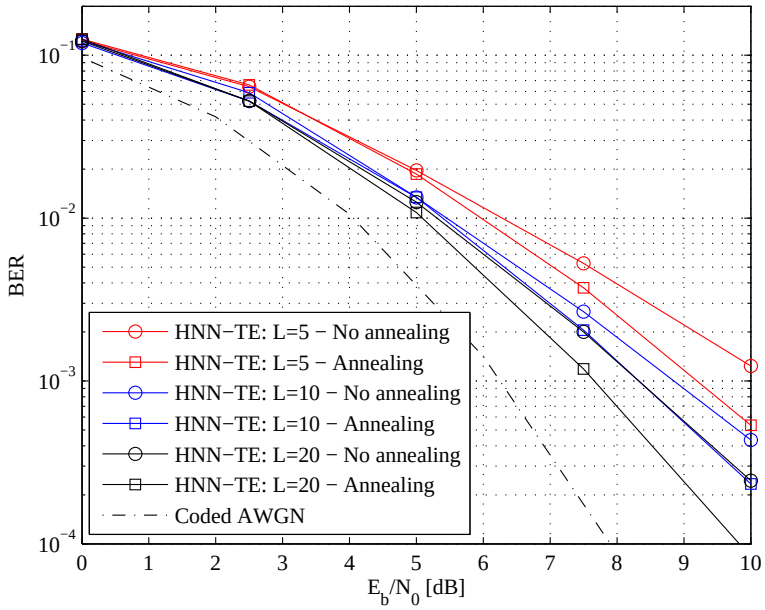


Figure 5.16: HNN-TE performance in long fading channels with and without the use of simulated annealing.

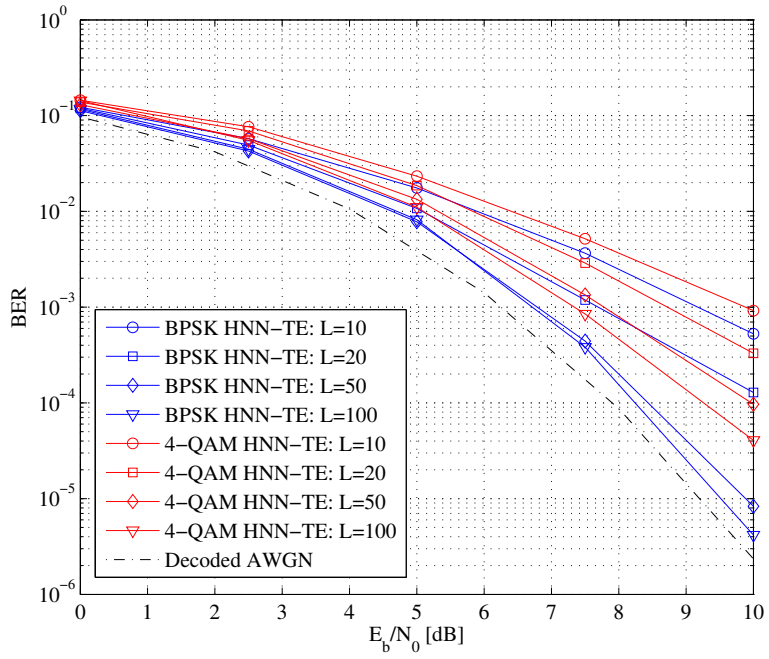


Figure 5.17: HNN-TE BPSK and 4-QAM performance in long channels.

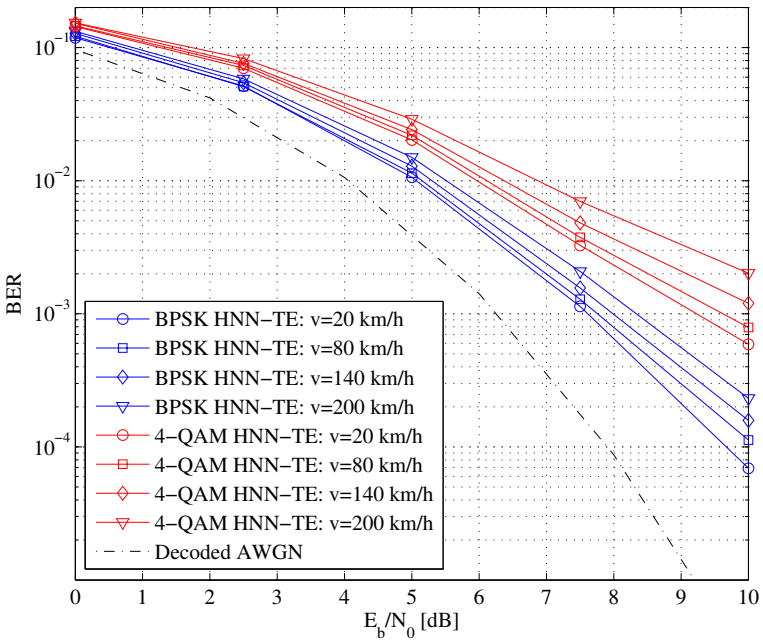


Figure 5.18: HNN-TE BPSK and 4-QAM performance in a long channel at various mobile speeds.

5.5.1.3 Mobile Speed

Fig. 5.18 shows the performance of the HNN-TE for a channel of length $L = 50$ at mobile speeds of 20 km/h, 80 km/h, 140 km/h, and 200 km/h for BPSK and 4-QAM modulation, assuming perfect CSI. It is clear that an increase in mobile speed leads to a performance degradation, although not as much as expected. Again BPSK modulation performs better than 4-QAM modulation.

5.5.1.4 Training Symbols

Fig. 5.19 shows the performance of the HNN-TE for a channel of length $L = 50$ at a mobile speed of 20 km/h for BPSK and 4-QAM modulation, assuming imperfect CSI. To estimate the channel, training sequences of length $4L$, $6L$, $8L$ and $10L$ were used. As expected, a performance loss is incurred with a decrease in the number of training symbols. Again BPSK modulation outperforms 4-QAM modulation.

5.5.1.5 Iterations

Fig. 5.20 shows the performance of the HNN-TE for a channel of length $L = 25$ at a mobile speed of 20 km/h for BPSK and 4-QAM modulation, assuming perfect CSI, for different numbers of iterations. The number of iterations was chosen to be $Z = 5$, $Z = 10$, $Z = 20$ and $Z = 50$. The BER performance increases with an increase in the number of iterations. Since the performance degradation due to a decrease in the number of iterations is low at low signal levels, this author adopts an iteration schedule that is dependent on the signal level. One uses the following function to determine the number of iterations: $Z(E_b/N_0) = 2(5^{(E_b/N_0)/5})$.

5.5.1.6 Code Rates

Fig. 5.21 shows the performance of the HNN-TE for a channel of length $L = 50$ at a mobile speed of 20 km/h for BPSK and 4-QAM modulation, assuming perfect CSI, for different code rates. The code rates were $R_c = 1/2$ (2/4), $R_c = 3/8$, $R_c = 1/4$ (4/16) and $R_c = 5/32$. From Fig. 5.21 it is clear that the performance of the HNN-TE increases with a decrease in the code rate, with 4-QAM modulation performing worse than BPSK modulation.

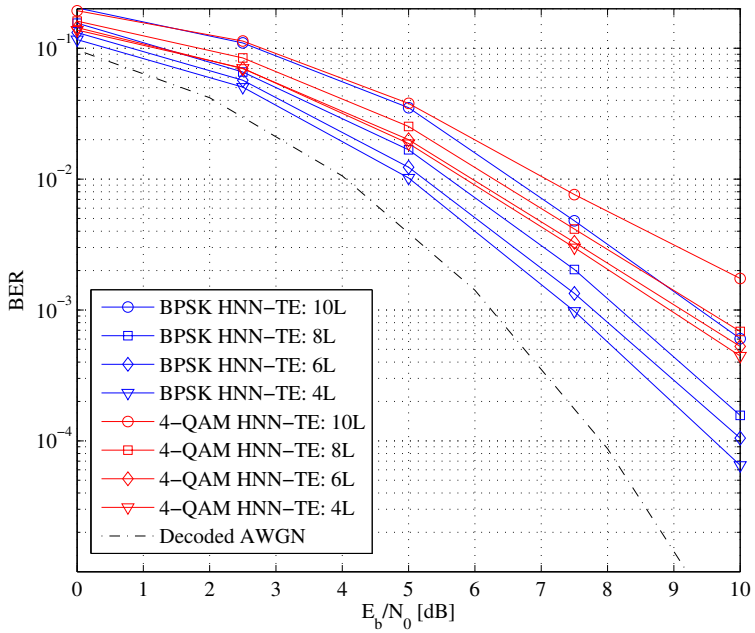


Figure 5.19: HNN-TE BPSK and 4-QAM performance in a long channel for various numbers of training symbols.

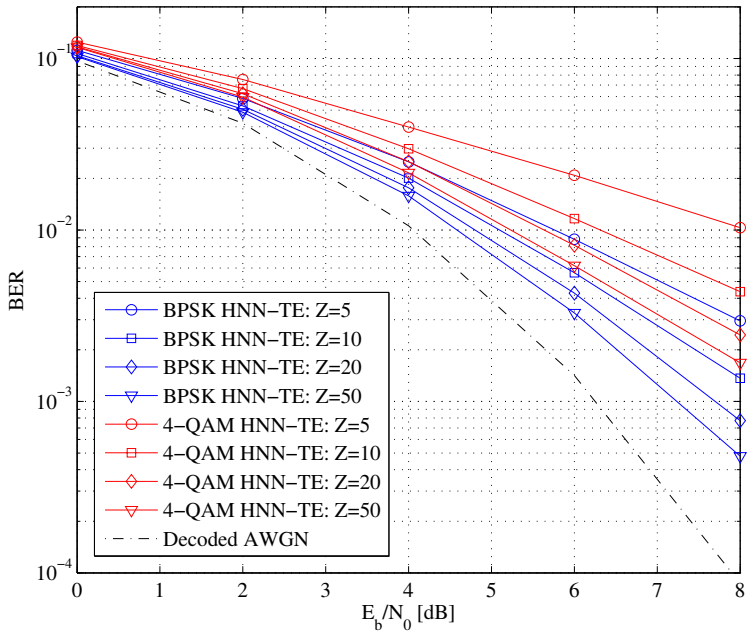


Figure 5.20: HNN-TE BPSK and 4-QAM performance in a long channel for various numbers of iterations.

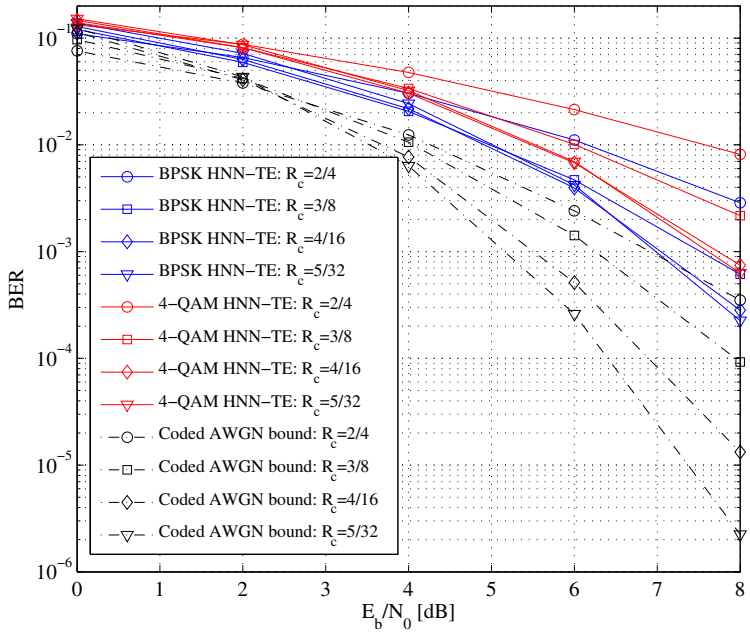


Figure 5.21: HNN-TE BPSK and 4-QAM performance in a long channel for different code rates.

5.5.2 HNN-TE vs CTE

5.5.2.1 Power Delay Profiles

The performance of the HNN-TE is compared to that of the CTE using the PDPs described in *Appendix A* for static and fading channels. The HNN-TE and the CTE are simulated for the uniform, linear and exponential PDPs for fading and static channels of length $L = 4$ and $L = 8$. The PDPs determine the tap weights, and as such influence the performance of the algorithms. Fig. 5.22 and Fig. 5.23 show the performance of the HNN-TE and the CTE in static channels of length $L = 4$ and $L = 8$ respectively, while the performance of the HNN-TE and the CTE in fading channels of length $L = 4$ and $L = 8$ is shown in Fig. 5.24 and Fig. 5.25.

From Fig. 5.22 and Fig. 5.23 it is clear that the CTE outperforms the HNN-TE for all but the exponential PDP in static channels. Fig. 5.22 shows that the CTE almost perfectly equalizes and decodes the received signal for all PDPs while the HNN-TE achieves acceptable performance for the uniform and linear PDP. The HNN-TE struggles to equalize and decode information when the uniform and linear PDPs are used. However, the HNN-TE performs well when the exponential PDP is used, since die bulk of the channel energy is concentrated in the leading CIR tap. From Fig. 5.24 and Fig. 5.25 it is clear that the HNN-TE consistently outperforms the CTE for all PDPs in fading channels. The HNN-TE is therefore not applicable to static channels where the channel energy is spread across all the taps. However, the HNN-TE is perfectly suited to fading channels regardless of the PDP.

5.5.2.2 Mobile Speed

The performance of the HNN-TE is evaluated and compared to that of the CTE at various mobile speeds in a fading channel of length $L = 6$. Fig. 5.26 shows the performance of the HNN-TE and the CTE for a channel of length $L = 6$ at mobile speeds of 3 km/h, 50 km/h, 80 km/h, 140 km/h, and 200 km/h, assuming perfect CSI. It is clear that the HNN-TE outperformans the CTE at mobile speeds greater than 20 km/h, with the advantage of performance increasing with an increase in mobile speeds. It seems that the HNN-TE is less affected by increasing mobile speeds, which suggests that the HNN-TE is able to handle channel estimation errors better that the CTE.

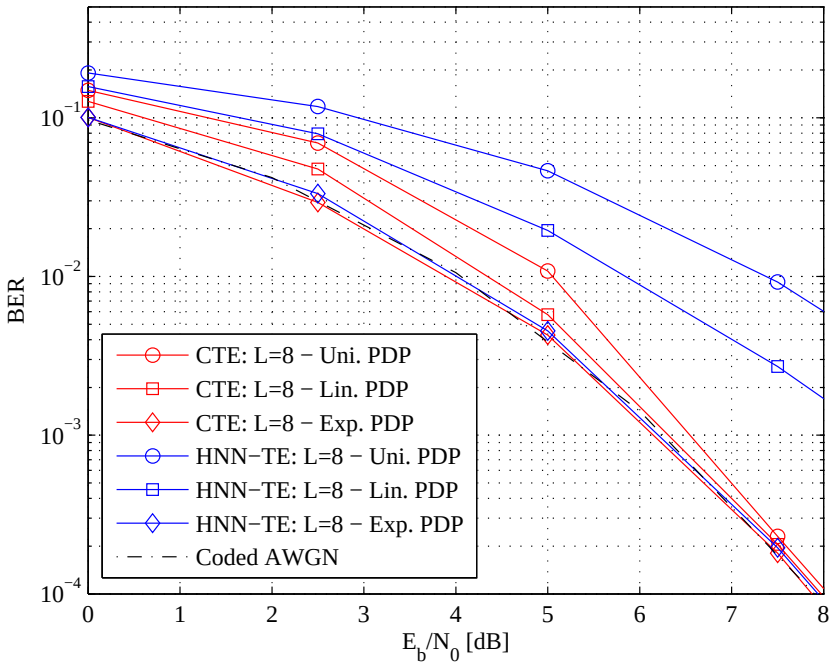


Figure 5.22: HNN-TE and CTE performance in static channels of length $L = 4$ for three PDPs.

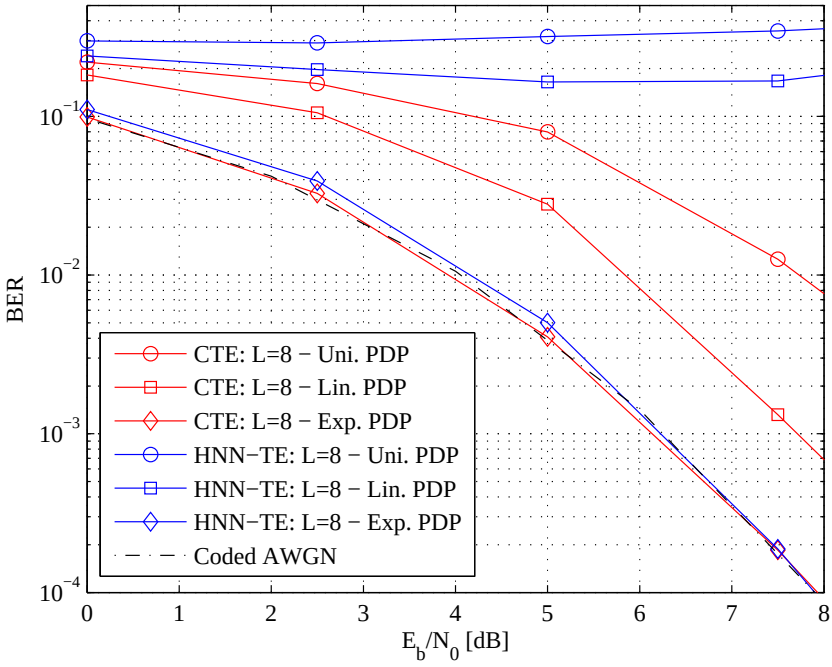


Figure 5.23: HNN-TE and CTE performance in static channels of length $L = 8$ for three PDPs.

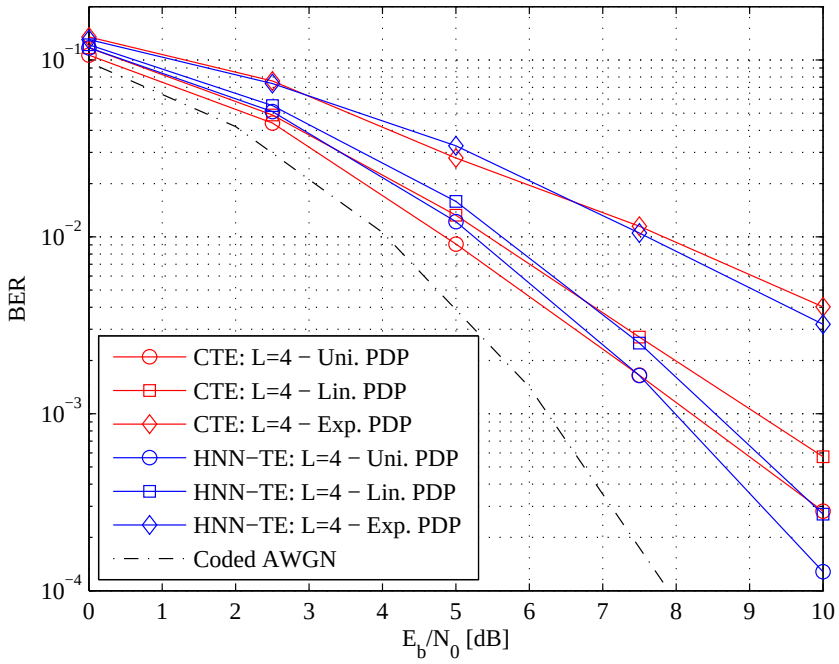


Figure 5.24: HNN-TE and CTE performance in fading channels of length $L = 4$ for three PDPs.

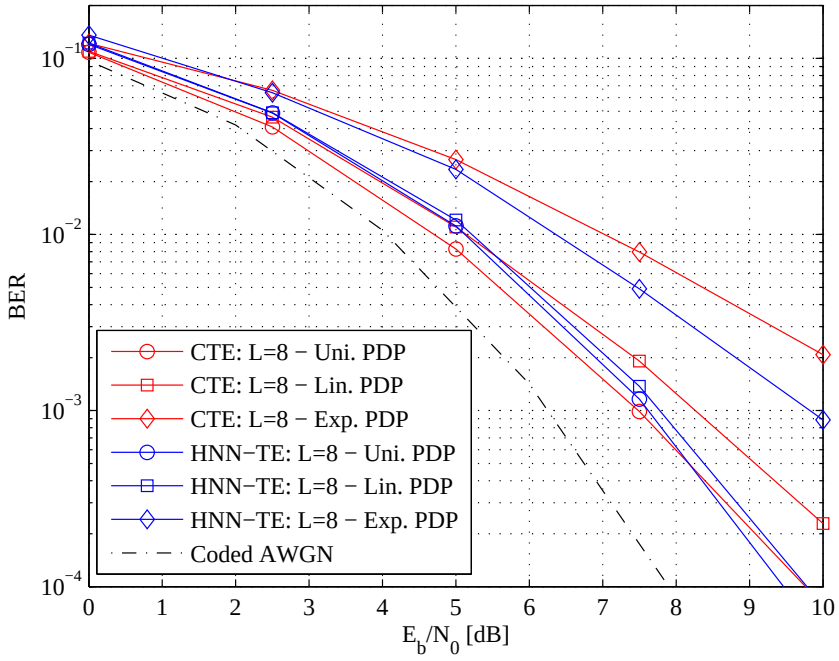


Figure 5.25: HNN-TE and CTE performance in faing channels of length $L = 8$ for three PDPs.

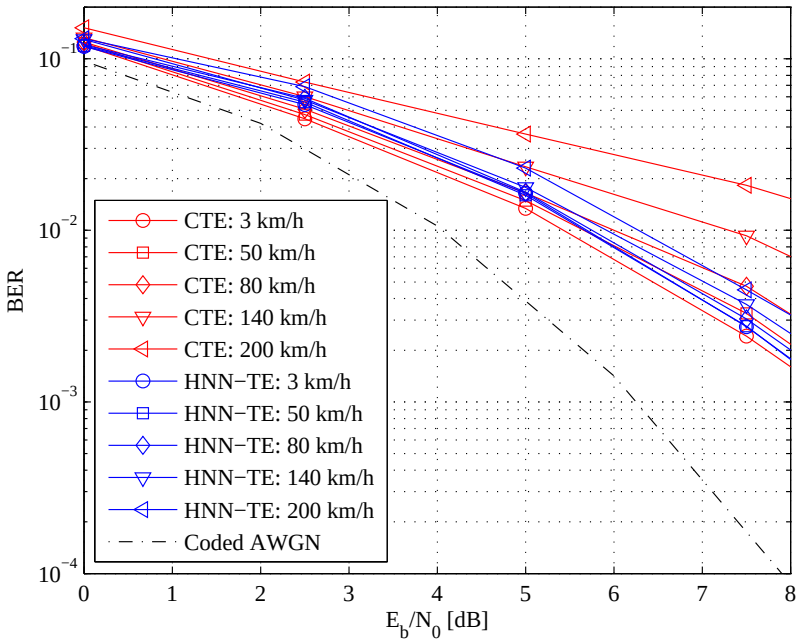


Figure 5.26: HNN-TE and CTE performance in a short channel at various mobile speeds.

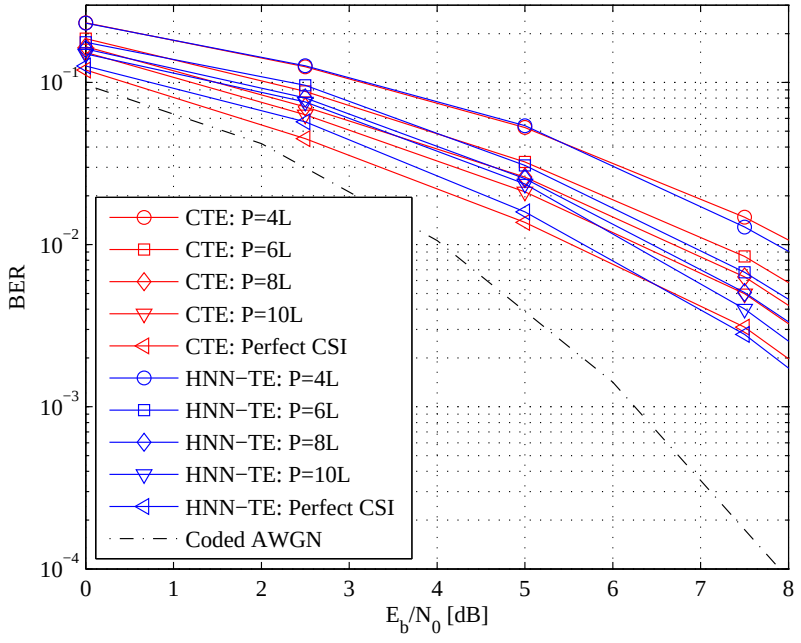


Figure 5.27: HNN-TE and CTE performance in a short channel for various numbers of training symbols used for channel estimation.

5.5.2.3 Training Symbols

To test the resilience of the HNN-TE against channel estimation errors, various numbers of training symbols are used to estimate the CIR using an LS estimator. To estimate the channel training sequences of length $P = 4L$, $P = 6L$, $P = 8L$ and $P = 10L$ are used. Fig. 5.27 shows the performance of the HNN-TE and the CTE for a channel of length $L = 6$ at a mobile speed of 20 km/h for various numbers of training symbols. From Fig. 5.27 it is clear that the performance of the HNN-TE and CTE are comparable and that the HNN-TE is able to achieve acceptable performance with fairly low numbers of training symbols.

5.6 CONCLUDING REMARKS

In this chapter an LCTE using the HNN as its underlying structure was presented, which is able to jointly equalize and decode BPSK and 4-QAM coded-modulated signals in systems transmitting interleaved information through a multipath fading channels. It was shown how sets of balanced codes can be generated from Walsh-Hadamard codes and the encoding process was discussed in detail. The HNN MLSE equalizer developed in [19, 20] was discussed, after which the HNN decoder in [21, 22] was discussed. It was shown how the HNN decoder can be modified to decode sequences of balanced codes in order to be merged with the HNN MLSE equalizer, resulting in an HNN MLSE decoder. It was further shown how the HNN MLSE equalizer can be merged with the HNN MLSE decoder in order to perform joint equalization and decoding using one HNN structure.

The resulting LCTE uses the HNN as framework and hence it was fittingly named the Hopfield Neural Network Turbo Equalizer, or HNN-TE. The HNN-TE is able to turbo equalize code modulated BPSK and 4-QAM signals in short as well as long multipath channels. It was shown that the HNN-TE outperforms the CTE in most simulation scenarios. It was clear that the HNN-TE is not suited to static channels. The HNN-TE's computational complexity in long channels is vastly superior to that of CTEs, but it is higher than that of the MMSE and DFE based LCTEs. The computational complexity of the HNN-TE is cubically related to the coded data block length, while being approximately independent of the CIR length. This enables it to turbo equalize signals in systems transmitting BPSK and 4-QAM modulated information using moderate coded data block lengths through multipath fading channels with multiple hundreds of multipath elements. The performance of the HNN-TE for BPSK modulation is better than for 4-QAM modulation, since Gray coding cannot be employed be-

cause of the coded modulation explained in this chapter, while the complexity for 4-QAM is slightly higher.

CHAPTER 6

CONCLUSION

6.1 SUMMARY

In this thesis two novel iterative joint equalizers and decoders, or turbo equalizers, were developed as low complexity alternatives to the well known, but computationally expensive, CTE, which employs an optimal MAP equalizer. This was done by treating the turbo equalization problem as a single optimization problem, and by designing two unique solutions using superstructures. The idea was to process all the available information (from the channel as well as the coding scheme used) as a whole, without exchanging extrinsic information between separate subunits. One concern was the fact that the interleaver destroys the causality relationship between the transmitted symbols and the received symbols, but in both cases a deinterleaving transformation was applied to restore the one-to-one relationship between each received symbol and its corresponding transmitted symbol via the fist CIR coefficient. ISI mitigation was then performed in an iterative fashion in both cases.

In *Chapter 2* equalization, decoding and joint equalization and decoding were discussed. Since the MLSE and MAP algorithms are used for both equalization and decoding, they were discussed for the general case, after which it was explained how each of them may be applied to the problems of equalization and decoding. Non-iterative joint equalization and decoding, dubbed the NI-JED, was then rigorously analyzed for the case where no interleaving is employed as well as for the case were block interleaving is employed. Analysis of the NI-JED was an important part of the quest to design low complexity joint equalization and decoding algorithms using superstructures, since the NI-JED perfectly models the joint equalization and decoding problem by using a super-trellis while achieving optimal performance without iterating. Although the NI-JED performs optimally, its complexity grows exponentially with an increase in channel memory length when no interleaver is

used, and exponentially in both the channel memory length, as well as the interleaver depth, when a block interleaver is used. The complexity of the NI-JED also increases exponentially with an increase in the encoder constraint length, but the constraint length is normally limited in length. In order to achieve the best performance in a wireless communication system transmitting information through a multipath channel, a random interleaver should be used. However, the NI-JED cannot be used when a random interleaver is employed, as there is no fixed interleaver structure from which the joint equalizer and decoder can be designed.

Chapter 3 was dedicated to turbo equalization. It was discussed how a MAP equalizer and a MAP decoder iteratively exchange information in the form of extrinsic information, which is then used by the other algorithm as prior information regarding the symbols to be estimated. With each iteration the output of each algorithm becomes more reliable, which leads to more reliable estimates after the next iteration etc. The computational complexity of CTEs are however problematic if the channel memory is large owing to the exponential relationship between the equalizer complexity and the channel memory. Attention was therefore paid to a number of low complexity equalizers - MMSE-LE, MMSE-DFE, SFE and SDFE - which can be used as a replacement for the MAP equalizer in a CTE. The derivations of these low complexity equalizers were presented based on the literature, and computational complexity analyses of the resulting LCTEs were performed. The computational complexities of the LCTEs were compared to that of the CTE, and simulation results from the literature were presented.

The DBN-TE was developed in *Chapter 4* by using a DBN to model the turbo equalization problem. For the DBN-TE to function correctly there had to be a dominant connection between the observed variables and their corresponding hidden variables, as well as weak connections to past and future hidden variables. This was achieved by performing a Cholesky transformation on the autocorrelation of the channel matrix and filtering the received symbols accordingly. It was however stated that Cholesky decomposition filtering is not adequate for some static channels, and therefore an MMSE-DF prefilter was also used to transform the channel to its minimum phase form. Another precondition was that the randomization effect of the interleaver had to be mitigated, which was achieved by filtering the received symbol sequence with the Hermitian transpose of the interleaved channel matrix. The computational complexity of the DBN-TE is approximately quadratic (cubic worst case) in the coded data block length, exponential in the encoder constraint length, as is standard in the CTE and all LCTEs, but approximately independent of the channel memory length. Its complexity is worse than that of the LCTEs considered in this thesis, but is far superior to that of the CTE for systems

with long channel memory. Its performance is comparable to that of a CTE in all fading channels and short static channels, and comparable to that of the LCTEs in the literature. Its performance is however lacking in some static channels.

The HNN-TE was derived in *Chapter 5* by combining the HNN MLSE equalizer and the HNN MLSE decoder. The HNN MSLE equalizer was developed by this author in [19, 20], and it has exceptionally low computational complexity due to the high parallelism of its underlying neural network structure, which makes it suitable for use in systems transmitting uncoded information through a highly dispersive multipath channel. The HNN MLSE decoder was proposed in [21, 22] for the decoding of balanced codewords, but it is only able to decode single balanced codewords. However, in order to merge the HNN MLSE decoder with the HNN MLSE equalizer, the decoder had to be able to decode sequences of balanced codewords, not single codewords only. This was achieved by scaling the correlation matrix with respect to the coded transmitted data block size and copying the correlation matrix of the single codeword HNN MLSE decoder along the diagonal, and instead of using single received codewords as input, received codeword sequences were used as input to the sequence based HNN MLSE decoder. The HNN MLSE equalizer and the HNN MLSE decoder were merged by simply normalizing the correlation matrix of the former and adding it to that of the latter, thus biasing the equalizer correlation matrix with that of the decoder, allowing for joint equalization and decoding to be performed in an iterative fashion. The computational complexity of the HNN-TE is approximately independent of the channel memory length and almost quadratically related to the coded data block length (cubic worst case), which allows it to jointly equalize and decode information transmitted through severely dispersive multipath channels. The HNN-TE outperforms the equivalent CTE in fading channels, but like the HNN MLSE equalizer in [19] and the DBN-TE developed in this thesis, the HNN-TE does not perform well in some static channels.

6.2 FURTHER RESEARCH

Although the DBN-TE and HNN-TE developed in this thesis are unique in the sense that they iteratively and jointly equalize and decode information using superstructures, they serve as a baseline for further investigation as proposed in the following:

6.2.1 DBN-TE

- It was developed for BPSK only. It should be possible to extend the DBN-TE to be able to jointly equalize and decode information transmitted using higher order modulation schemes.
- Only convolution codes were investigated. The DBN-TE could be adapted for convolution codes in higher order Galois fields as well as for LDPC codes.
- The DBN-TE does not perform well in some static channels. Better optimization techniques and prefitering might improve its performance in static channels.

6.2.2 HNN-TE

- To date the only HNN decodable codes are balanced check codes, which do not provide good coding gains. In order to make the HNN-TE attractive for practical use, better HNN decodable codes must be found.
- Like the DBN-TE, the HNN-TE does not perform well in static channels. Better optimization techniques might improve its performance in static channels.

6.3 CONCLUDING REMARKS

The DBN-TE and HNN-TE developed in this thesis have been thoroughly analyzed and simulated, and their performance and computational complexity characteristics have been compared to that of the CTE as well as other LCTEs. While offering very low computational complexity, the performance of both turbo equalizers are comparable to that of the CTE. The proposed turbo equalizers are unique in the sense that they make use of superstructures to solve the turbo equalization problem, doing so without any limitation regarding the structure of the interleaver.

REFERENCES

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

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

[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*, vol. IT-20, pp. 284–287, March 1974.

[4] G. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

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

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

[7] R. Koetter, M. Tüchler, and A. Singer, "Turbo Equalization," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 67–80, 2004.

[8] M. Tüchler, A. Singer, and R. Koetter, "Minimum Mean Squared Error (MMSE) Equalization Using A Priori Information," *IEEE Transactions on Signal Processing*, vol. 50, pp. 673–683,

2000.

[9] X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Transactions on Communications*, vol. 47, no. 7, pp. 1046–1061, 1999.

[10] R. Lopes and J. Barry, "The Soft Feedback Equalizer for Turbo Equalization of Highly Dispersive Channels," *IEEE Transactions on Communications*, vol. 54, no. 5, pp. 783–788, May 2006.

[11] H. Lou and C. Xiao, "Soft-Decision Feedback Turbo Equalization for Multilevel Modulations," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 186–195, January 2011.

[12] R. Koetter, M. Tüchler, and A. Singer, "Turbo Equalization: Principles and New Results," *IEEE Transactions on Communications*, vol. 50, no. 5, pp. 754–767, 2002.

[13] N. Wiberg, H. Loeliger, and R. Kotter, "Codes and iterative decoding on general graphs," *European Transactions on Telecommunications*, vol. 6, no. 5, pp. 513–525, 1995.

[14] A. Knickenberg, B. Yeap, J. Hamorsky, M. Breiling, and L. Hanzo, "Non-iterative joint channel equalization and channel decoding," *Globecom*, vol. 9, no. 3, pp. 442–446, November 1999.

[15] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Elsevier Artificial Intelligence*, vol. 29, no. 3, pp. 241–288, September 1986.

[16] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentic Hall, New Jersey, 2003.

[17] N. Wiberg, H. Loeliger, and R. Kotter, *Codes and Iterative Decoding on General Graphs*. Wiley, 2008.

[18] R. McEliece, D. MacKay, and J. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, 1998.

[19] H. Myburgh and J. Olivier, "Low Complexity MLSE Equalization in Highly Dispersive Rayleigh Fading Channels," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, no. ID 874874, 2010.

[20] H. Myburgh, "Low Complexity Iterative MLSE Equalization in Extremely Long Rayleigh Fading Channels," *Master's Dissertation, University of Pretoria*, 2010.

[21] N. Wiberg, "A class of Hopfield decodable codes," *Proceedings of the IEEE-SP Workshop on Neural Networks for Signal Processing*, pp. 88–97, 1993.

[22] V. Bhargava and Q. Wang, "An error correcting neural network," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 530–533, June 1989.

[23] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, no. 1, pp. 260–269, 1967.

[24] J. Hagenauer, "A Viterbi algorithm with soft-decision outputs and its applications," *Globecom*, vol. 3, pp. 1680–1686, November 1989.

[25] M. Breiling and L. Hanzo, "The super-trellis structure of turbo codes," *IEEE Transactions on Information Theory*, vol. 46, no. 6, pp. 2212–2228, 2000.

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

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

[28] S. ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, October 2001.

[29] J. Hagenhauer, "The EXIT chart - Introduction to extrinsic information transfer in iterative processing," *Proceedings of the 12th European Signal Processing Conference*, vol. 49, no. 10, pp. 1541–1548, October 2004.

[30] R. Lopes, "Iterative estimation, equalization and decoding," Ph.D. dissertation, Citeseer, 2003.

[31] F. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions onInformation Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[32] N. F. D. Koller, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[33] A. Worthen and W. Stark, "Unified design of iterative receivers using factor graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 843–849, 2001.

[34] H. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28–41, 2004.

[35] H. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. Kschischang, "The factor graph approach to model-based signal processing," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1295–1322, 2007.

[36] G. Colavolpe, "On LDPC codes over channels with memory," *IEEE Transactions on Wireless Communications*, vol. 5, no. 7, pp. 1757–1766, 2006.

[37] T. Mittelholzer, A. Dholakia, and E. Eleftheriou, "Reduced-complexity decoding of low density parity check codes for generalized partial response channels," *IEEE Transactions on Magnetics*, vol. 37, no. 2, pp. 721–728, 2001.

[38] B. Kurkoski, P. Siegel, and J. Wolf, "Joint message-passing decoding of LDPC codes and partial-response channels," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1410–1422, 2002.

[39] F. Kschischang and B. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 219–230, 1998.

[40] Q. Guo, L. Ping, and H. Loeliger, "Turbo equalization based on factor graphs," in *International Symposium on Information Theory*. IEEE, 2005, pp. 2021–2025.

[41] Q. Guo and L. Ping, "LMMSE turbo equalization based on factor graphs," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 2, pp. 311–319, 2008.

[42] R. Drost and A. Singer, "Factor-graph algorithms for equalization," *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 2052–2065, 2007.

[43] G. Colavolpe and G. Germi, "On the application of factor graphs and the sum-product algorithm to isi channels," *IEEE Transactions on Communications*, vol. 53, no. 5, pp. 818–825, 2005.

[44] H. Niu, M. Shen, J. Ritcey, and H. Liu, "A factor graph approach to iterative channel estimation and LDPC decoding over fading channels," *IEEE Transactions on Wireless Communications*, vol. 4, no. 4, pp. 1345–1350, 2005.

[45] ——, "Iterative channel estimation and LDPC decoding over flat-fading channels: A factor graph approach," in *Conference on Information Sciences and Systems (CISS), Baltimore*. Citeseer, 2003.

[46] A. Eckford *et al.*, "Channel estimation in block fading channels using the factor graph em algorithm," in *22nd Biennial Symposium on Communications*. Citeseer, 2004, pp. 1–3.

[47] Y. Weiss, "Belief Propagation and Revision in Networks with Loops," *MIT Press, Cambridge*, 1997.

[48] Y. Weiss and W. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Mitsubishi Elecric Research Laboratory*, vol. TR-99-38, 1999.

[49] M. J. K.P. Murphy, Y. Weiss, "Loopy Belief Propagation for Approximate Inference: An Empirical Study," *Proceedings of Conference on Uncertainty in Artificial Intelligence*, vol. 15, 1999.

[50] M. Krieg, "A tutorial on Bayesian belief networks," *Defence Science and Technology Organisation Salisbury (Australia) Electronics and Surveillance Research*, vol. DSTO-TN-0403, 2001.

[51] Y. Weiss, "Correctness of Local Probability Propagation in Graphical Models with Loops," *Neural Computation*, vol. 12, pp. 1–41, 2000.

[52] J. Wu and Y. Zheng, "Low Complexity Soft-Input Soft-Output Block Decision Feedback Equalization," *IEEE Journal on Selected Areas in Communication*, vol. 26, no. 2, pp. 281–289, February 2008.

[53] J. Wu, S. Leong, K. Lee, C. Xiao, and J. Olivier, "Improved BDFE Using A Priori Information for Turbo Equalization," *IEEE Transactions on Wireless Communication*, vol. 7, no. 1, pp. 233–240, January 2008.

[54] M. Hazewinkel, *Encyclopedia of Mathematics*. Springer, 2001.

[55] S. W. D. Coppersmith, "Matrix multiplication via arithmetic progressions," *Journal of Symbolic*

Computation, vol. 9, no. 3, pp. 251–280, June 1990.

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

[57] J. C. Olivier, W. Kleynhans, and S. Miteff, "Teaching the theory of estimation and detection via a GSM radio interface simulation," *IEEE Transactions on Education*, vol. 49, no. 1, pp. 61–66, September 2006. [Online]. Available: http://dx.doi.org/10.1109/TE.2005.856153

[58] M. Tüchler, "Iterative Equalization Using Priors," *Master's thesis, University of Illinois at Urbana Champaign, USA, 2000.*

[59] "Radio transmission and reception," *ETSI. GSM 05.05*, vol. ETSI EN 300 910 V8.51, no. 2, November 2000.

[60] Y. Yin, Y. Huang, and J. Zhang, "Turbo equalization using probabilistic data association," *Proceedings of Global Telecommunications Conference*, vol. 4, pp. 2535–2539, October 2004.

[61] S. Jeong and J. Moon, "Soft-In Soft-Out DFE and Bi-Directional DFE," *IEEE Transactions in Communications*, vol. 59, no. 10, pp. 2729–2741, October 2011.

[62] H. Lou and C. Xiao, "Soft-Decision Feedback Turbo Equalization for Multilevel Modulations," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 186–195, 2011.

[63] G. Kechriotis, E. Zervas, and E. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 267–278, 1994.

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

[65] C. Lee, J. Go, B. Baek, and H. Choi, "Neural network equalizer," *Intelligent Computing*, pp. 204–215, 2006.

[66] H. Zhao and J. Zhang, "Adaptively combined FIR and functional link artificial neural network equalizer for nonlinear communication channel," *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 665–674, 2009.

[67] W. Teich and M. Seidl, "Code division multiple access communications: Multiuser detection based on a recurrent neural network structure," in *International Symposium on Spread Spectrum Techniques and Applications*, vol. 3. IEEE, 1996, pp. 979–984.

[68] B. Aazhang, B. Paris, and G. Orsak, "Neural networks for multiuser detection in code-division multiple-access communications," *IEEE Transactions on Communications*, vol. 40, no. 7, pp. 1212–1222, 1992.

[69] T. Miyajima, T. Hasegawa, and M. Haneishi, "On the multiuser detection using a neural network in code-division multiple-access communications," *IEICE Transactions on Communications*, vol. 76, no. 8, pp. 961–968, 1993.

[70] S. Verdu, "Adaptive multiuser detection," in *IEEE Third International Symposium on Spread Spectrum Techniques and Applications*. IEEE, 1994, pp. 43–50.

[71] W. Teich, M. Seidl, and M. Nold, "Multiuser detection for DS-CDMA communication systems based on recurrent neural network structures," in *5th International Symposium on Spread Spectrum Techniques and Applications*, vol. 3. IEEE, 1998, pp. 863–867.

[72] W. Caid and R. Means, "Neural network error correcting decoders for block and convolutional codes," in *Global Telecommunications Conference, 1990, and Exhibition.'Communications: Connecting the Future', GLOBECOM'90., IEEE.* IEEE, 1990, pp. 1028–1031.

[73] H. Abdelbaki, E. Gelenbe, and S. El-Khamy, "Random neural network decoder for error correcting codes," in *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, vol. 5. IEEE, 1999, pp. 3241–3245.

[74] S. Bang and B. Sheu, "A neural network for detection of signals in communication," *Circuits and Systems I: IEEE Transactions on Fundamental Theory and Applications*, vol. 43, no. 8, pp. 644–655, January 1996.

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

[76] S. Bang, B. Shue, and J. Bing, "A Neural Network for Detection of Signals in Communication,"

IEEE Transactions on Circuits and Systems, vol. 42, no. 8, pp. 644–655, July 1996.

[77] J. Platt and J. Hopfield, "Analog decoding using neural networks," in *AIP Conference Proceedings*, vol. 151, 1986, p. 364.

[78] C. Berrou and V. Gripon, "Coded Hopfield networks," in *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*. IEEE, 2010, pp. 1–5.

[79] H. Lin, J. Wu, and L. Fu, "Solving problems of maximum likelihood decoding of graph theoretic codes via a Hopfield neural network," in *IEEE International Sympoisum on Circuits and Systems*. IEEE, 1991, pp. 1200–1203.

[80] G. Kechriotis and E. Manolakos, "Implementing the optimal CDMA multiuser detector with Hopfield neural networks," in *International Workshop on Applications of Neural Networks to Telecommunications*, 1993, pp. 60–66.

[81] ——, "Hopfield neural network implementation of the optimal CDMA multiuser detector," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 131–141, 1996.

[82] T. Miyajima and T. Hasegawa, "Multiuser detection useng a Hopfield network for asynchronous code-division multiple-access systems," *IEICE Transactions on fundamentals of Electronics, Communications and Computer Sciences*, vol. 79, no. 12, pp. 1963–1971, 1996.

[83] E. Soujeri and H. Bilgekul, "Hopfield multiuser detection of asynchronous MC-CDMA signals in multipath fading channels," *IEEE Communications Letters*, vol. 6, no. 4, pp. 147–149, 2002.

[84] X. Liu, X. Wang, Z. Wu, and X. Gu, "Modified Hopfield neural network for CDMA multiuser detection," *Advances in Neural Networks-ISNN 2006*, pp. 88–93, 2006.

[85] T. Koike and S. Yoshida, "Approximated ML detector using Hopfield neural network in MIMO spatial multiplexing systems," 2005.

[86] S. Mohammed, A. Chockalingam, and B. Sundar Rajan, "A low-complexity near-ML performance achieving algorithm for large MIMO detection," in *IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 2012–2016.

[87] D. Knuth, "Efficient Balanced Codes," *IEEE Transactions on Information Theory*, vol. IT-32,

no. 1, pp. 530–533, June 1986.

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

[89] J. Hopfield, "Artificial Neural Networks," *IEEE Circuits and Devices Magazine*, 1988.

[90] D. Hebb, *The Organization of Behavior*, 4th ed. Wiley, New York, 1949.

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

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

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

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

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

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

APPENDIX A

SIMULATION ENVIRONMENT

The simulation enviroment used to produce the simulation results in this thesis is described here. Since the author of this thesis used a similar simulation environment than the one in [20], the desciption is similar to that in *Chapter 3* of [20].

A.1 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 BPSK and 4-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 $log_2(M)$ [2].

Figures A.1 and A.2 show the normalized[1] Gray coded[2] constellation maps using BPSK and 4-QAM modulation, respectively, with their respective mappings shown in tables A.1 to A.2. The source bits are mapped to modulation symbols according to 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 have been estimated at the receiver, the constellation map is again used to map

[1] The constellation maps are normalized such that the Euclidean distance from the origin 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 differs by one bit. This will increase the probability of one bit error when a symbol error is made during detection [2].

the symbols back to bits in order to produce estimates of the transmitted source information.

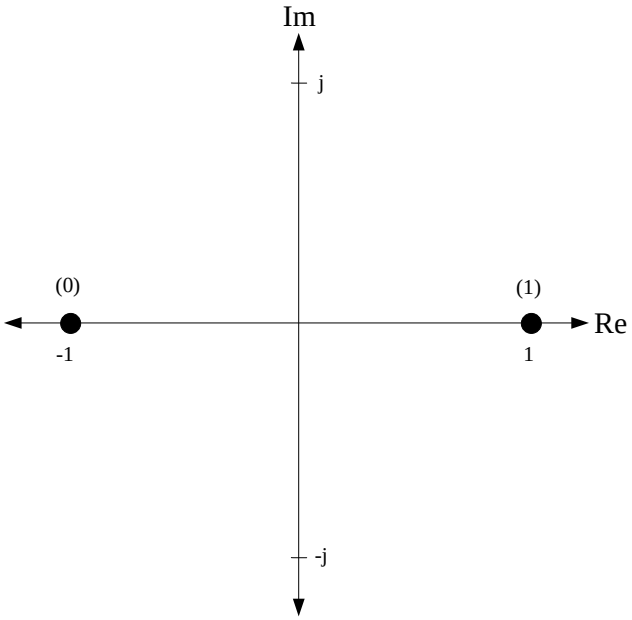


Figure A.1: BPSK constellation map.

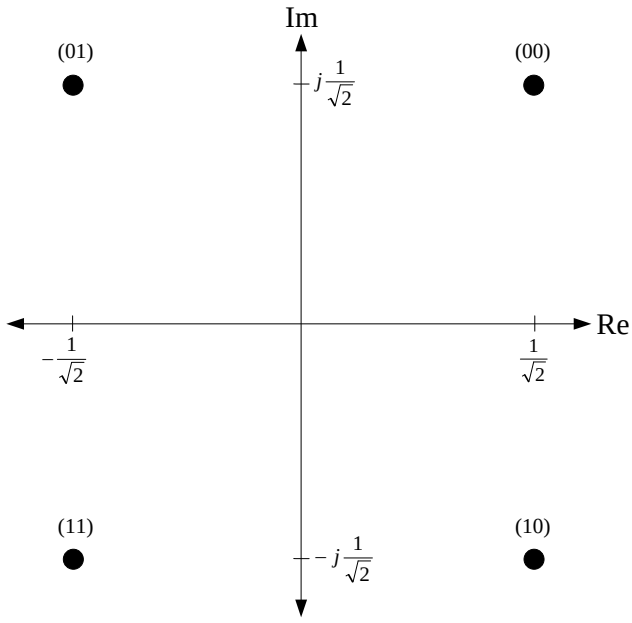


Figure A.2: 4-QAM constellation map.

Table A.1: BPSK symbol mapping.

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

Table A.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	$-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$	$\angle \frac{-\pi}{4}$

A.2 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 [2]. 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 [2]. The interleaver considered in this thesis is of the class of random interleavers, without the restriction imposed on S-random interleavers [92].[3] Fig. A.3 shows a random interleaver of length

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

$N = 20$. In this thesis block interleavers are also considered. Please refer to *Section 2.4.2*.

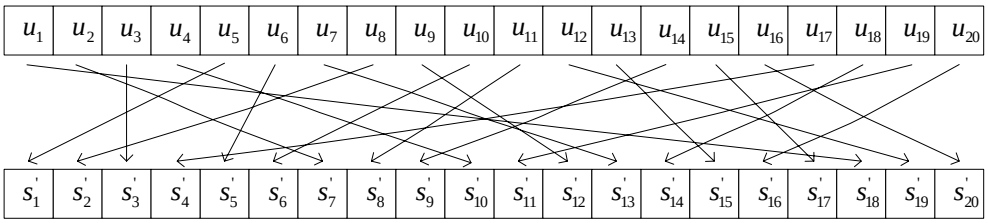


Figure A.3: Random interleaver of length $N = 20$.

A.3 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 over time [2]. During transmission, the transmitted symbols pass through the channel, which acts like a filter. The channel has a continuous time-varying impulse response, which is estimated at the receiver to aid in the estimation of the transmitted information.

A.3.1 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 tT_s.[4] As such, a tapped delay line is used to model the behavior of this channel, as shown in Fig. A.4.

[4]The section is a repetition of the description in *Section 2.2*. It is mentioned here again for completeness

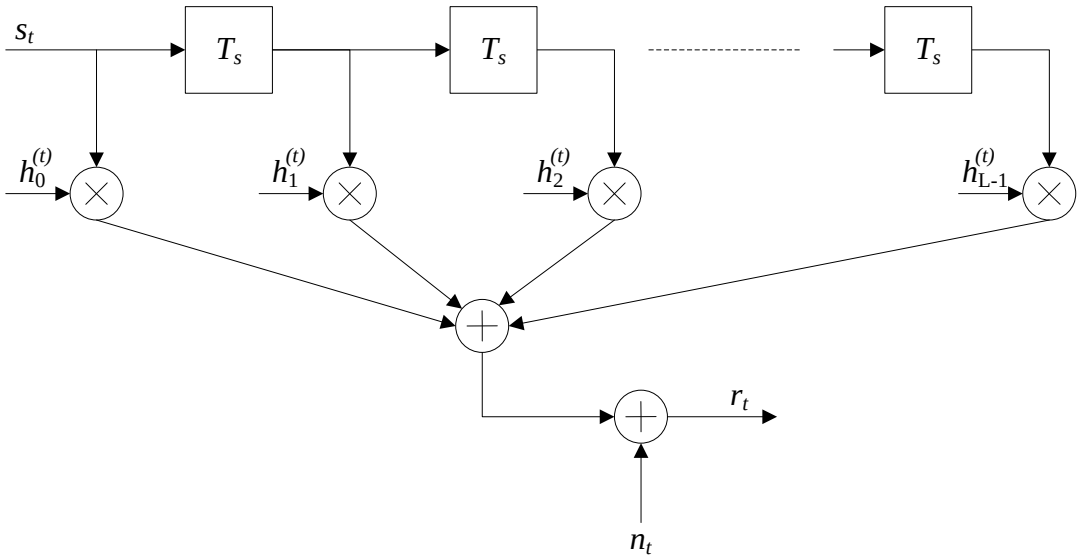


Figure A.4: Tapped delay line for a multipath fading channel.

Fig. A.4 indicates that the tth transmitted symbol s_t is delayed by T_s seconds $L-1$ times, where L is the channel impulse response (CIR) length and T_s is the symbol period. Each delayed copy of s_t is multiplied by $h_l^{(t)}$, $l = 1,2,...,L-1$, corresponding to the lth delay branch at time t. Therefore, the tth received symbol can be described by [1,2]

$$r_t = \sum_{l=0}^{L-1} h_l^{(t)} s_{t-l} + n_t, \tag{A.1}$$

where n_t is the tth noise sample from the distribution $\mathcal{N}(\mu = 0, \sigma^2 = 1)$. Each $h_l^{(t)}$ is a sample taken from one of L time-varying functions, where t corresponds with the tth transmitted symbol and $l = 1,2,...,L-1$ is the CIR tap number. Each CIR tap is modeled as an independent uncorrelated Rayleigh fading sequence, describing the fading behavior of that tap..

If it is assumed that the CIR is time-invariant for the duration of a data block, (A.1) can be rewritten as

$$r_t = \sum_{l=0}^{L-1} h_l s_{t-l} + n_t, \tag{A.2}$$

where s_t denotes the tth complex symbol in the transmitted sequence of N symbols chosen from an alphabet $\mathscr{D}$ containing M complex symbols, r_t is the tth received symbol, n_t is the tth noise sample from the distribution $\mathcal{N}(\mu = 0, \sigma^2 = 1)$, and h_l is the lth coefficient of the estimated CIR. Equalization is performed under the assumption that each CIR coefficient h_l is time-invariant for the duration of a data block. The CIR $\mathbf{h} = \{h_0, h_1, \ldots, h_{L-1}\}$ therefore completely describes the multipath channel

for a given received data block, assuming that the data block is sufficiently short so as to render the CIR time-invariant. The equalizer takes as input the received symbol sequence **r** as well as the CIR **h**.

A.3.2 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 [93]. 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 [93]. In flat fading channels there is no inter-symbol interference (ISI) and in frequency-selective channels there is ISI.

When channel fading is considered, a 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 [2]. The former is considered in this thesis.

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

$$R_k = \sqrt{X_k^2 + Y_k^2}, \tag{A.3}$$

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

$$p(x) = \frac{x}{\sigma_r^2} exp\left(\frac{-x^2}{2\sigma_r^2}\right), \qquad x \geq 0. \tag{A.4}$$

The model proposed in [94] 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 [95]. Clarke's model is given by

$$g(t) = \sqrt{\frac{2}{N}} exp\left(\omega_d t cos\alpha_n + \phi_n\right) \tag{A.5}$$

where α_n is the angle of the incoming wave, ϕ_n is the initial phase associated with the nth propagation, and ω_d is the maximum angular Doppler frequency when $\alpha_n = 0$. The model in (A.5) 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 $\alpha_n = (2\pi n - \pi + \theta)/N$, where θ is uniformly distributed in $[-\pi;\pi)$ [94]. The complex low-pass Rayleigh fading process in (A.5) is therefore described by

$$g(t) = \sqrt{\frac{2}{N}} \left(\sum_{n=1}^{M} 2cos(w_d t cos\alpha_n + \phi_n) + j \sum_{n=1}^{M} 2cos(w_d t sin\alpha_n + \varphi_n) \right), \tag{A.6}$$

of which the PDF of the magnitude will approximate a Rayleigh distribution [94]. 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 [94]

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

where

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

and ϕ_n, φ_n and θ are statistically independent and uniformly distributed in $[-\pi;\pi)$ for all n.

A.3.3 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 [2]. 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

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{|x-\mu|^2}{2\sigma_n^2} \right), \tag{A.7}$$

where μ is the mean and σ_n^2 is the variance. The assumption of AWGN is important for designing equalizers for digital communication systems. This will become clear in *Section 2.2*. The simulation of Gaussian noise, and how it is applied to the received symbols, is discussed next.

A.3.3.1 AWGN Simulation

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

$$SNR = \frac{\sigma_s^2}{\sigma_n^2}, \tag{A.8}$$

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 [96], 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 [96], the value by which the Gaussian random number generator output is scaled, is expressed as

$$k_\eta = \sqrt{\frac{\sigma_s^2 . f_{samp}}{2 . f_{bit} . 10^{\frac{E_b}{10 . N_0}}}}, \tag{A.9}$$

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 system 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.

A.3.4 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 [93]

$$f_D = \frac{v f_c}{c} \theta_D, \tag{A.10}$$

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 [93]:

- 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.

A.3.5 Power Delay Profiles

The power delay profile (PDP) represents the average power associated with a given multipath delay [93]. The PDP 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 [93]. In other words, since the symbol period

$$T_s = \frac{1+\alpha}{B}, \tag{A.11}$$

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 [93].

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

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

A.3.5.1 Exponential PDP

The exponential PDP coefficients are determined by

$$h_l = \exp\left(\frac{-l\tau_{max}}{L\tau_e}\right) \tag{A.12}$$

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

$$\tau_e = \frac{-\tau_{max}}{\ln\left(10^{\frac{P_{drop}}{10}}\right)}, \tag{A.13}$$

where P_{drop} is the relative power drop between $t = 0$ and $t = \tau_{max}$, with a value of -30 dB [96]. The normalized exponential PDP coefficients are shown in Fig. A.5 for $L = 20$.

A.3.5.2 Linear PDP

The linear PDP coefficients are determined by

$$h_l = \frac{L-l}{L}, \tag{A.14}$$

where $l = 0, 1, 2, ..., L-1$. The normalized linear PDP is shown in Fig. A.6 for $L = 20$.

A.3.5.3 Uniform PDP

The uniform PDP coefficients are determined by

$$h_l = 1, \tag{A.15}$$

where $l = 0, 1, 2, ..., L-1$. Fig. A.7 shows a normalized uniform PDP for $L = 20$.

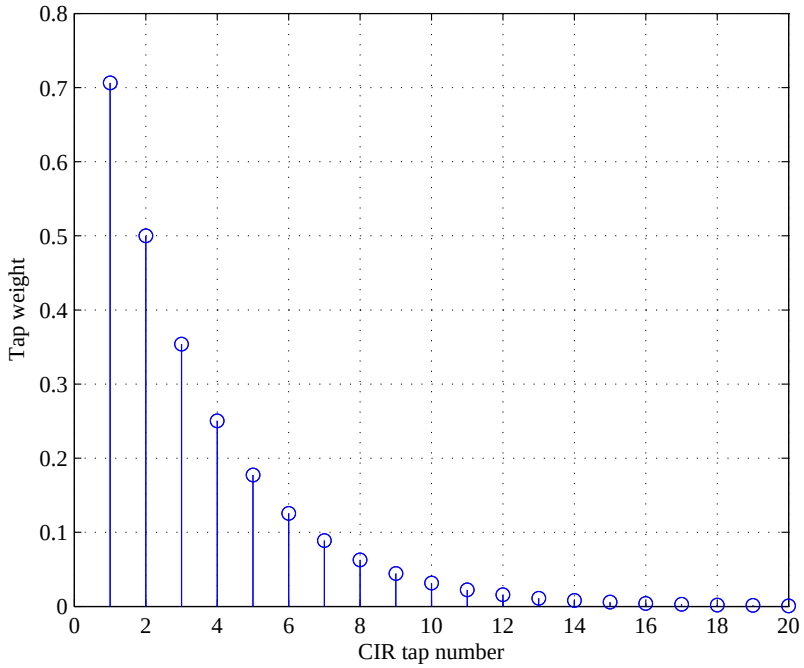


Figure A.5: Exponential power delay profile for $L = 20$.

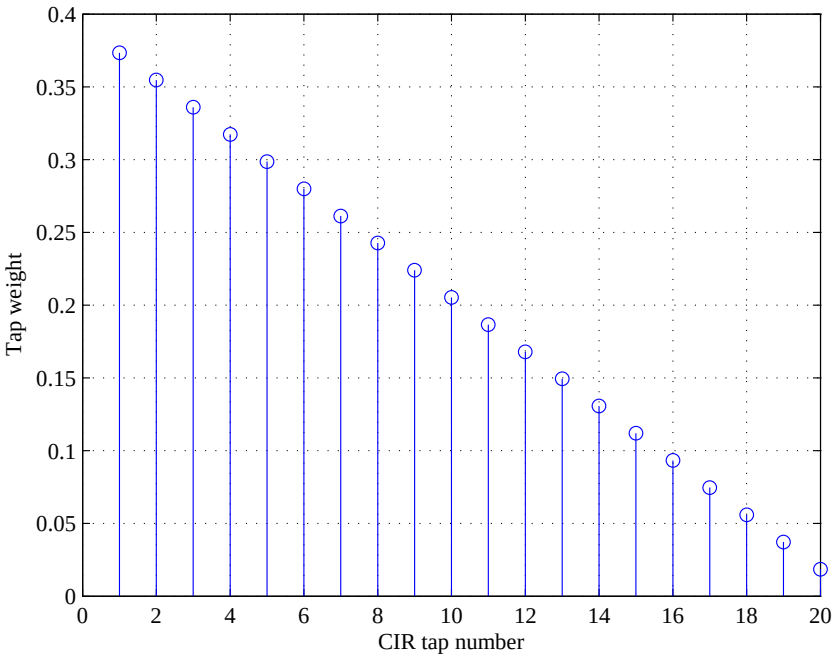


Figure A.6: Linear power delay profile for $L = 20$.

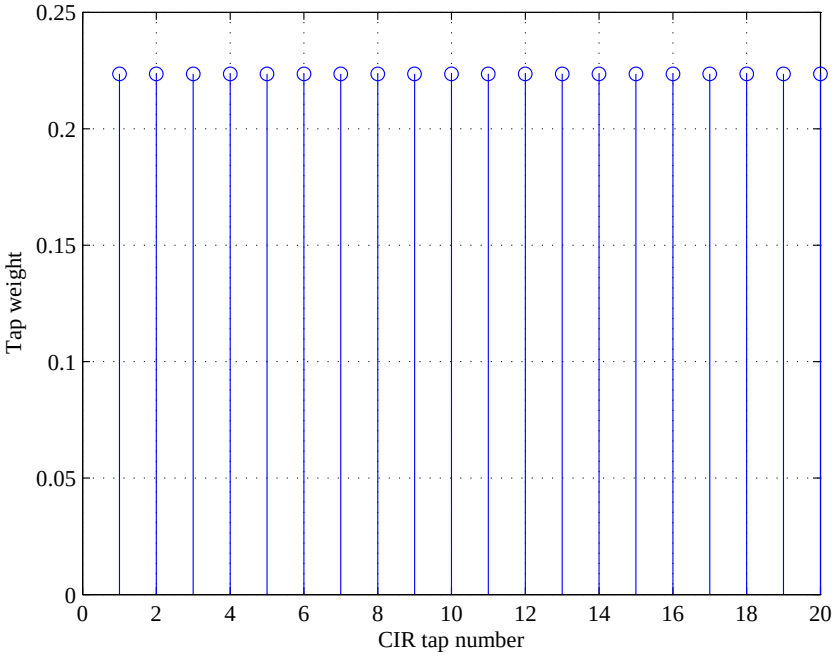


Figure A.7: Uniform power delay profile for $L = 20$.

A.3.5.4 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 $\mathbf{h} = \{h_0, h_1, \cdots, h_{L-1}\}^T$ are normalized such that

$$\mathbf{h}^T\mathbf{h} = 1. \tag{A.16}$$

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,

$$\mathbf{h} = \frac{h_0}{\|\mathbf{h}\|}, \frac{h_1}{\|\mathbf{h}\|}, \cdots, \frac{h_{L-1}}{\|\mathbf{h}\|} \tag{A.17}$$

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

$$\|\mathbf{h}\| = \sqrt{h_0^2 + h_1^2 + \ldots + h_{L-1}^2}. \tag{A.18}$$

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.

A.3.6 Frequency Hopping

Frequency hopping is applied in a wireless communication system in order to aid the receiver to better detect the received signal. Based on the coherence bandwidth (B_c) of the communication system, explained in *Section A.3.4*, the frequency at which a given data block is transmitted is changed, or hopped, to ensure that the detrimental effects of fading is combated. Frequency hopping provides a mean to artificially create temporal diversity in order to improve the performance. Figure A.8 through Figure A.11 shows three uncorrelated fading vectors, corresponding to a system with a CIR length of $L = 3$, for a duration of $N = 1200$ data symbols transmitted at a mobile speed of 50 km/h, where $F = 1$, $F = 2$, $F = 4$ and $F = 8$ frequencies were respectively utilized for frequency hopping. With each frequency hop, a unique uncorrelated fading vector is obtained, which minimizes the chance of the entire transmitted data block being in a deep fade, where the signal might be unrecoverable. Thus, by applying frequency hopping, signal fidelity is improved, which leads to performance gains. One detrimental aspect of frequency hopping, however, is that channel estimation will have to be performed for each frequency that was hopped to in the data block. This will add computational complexity, and it will also require the addition of more training symbols in each frequency hopped section of the data block for channel estimation.

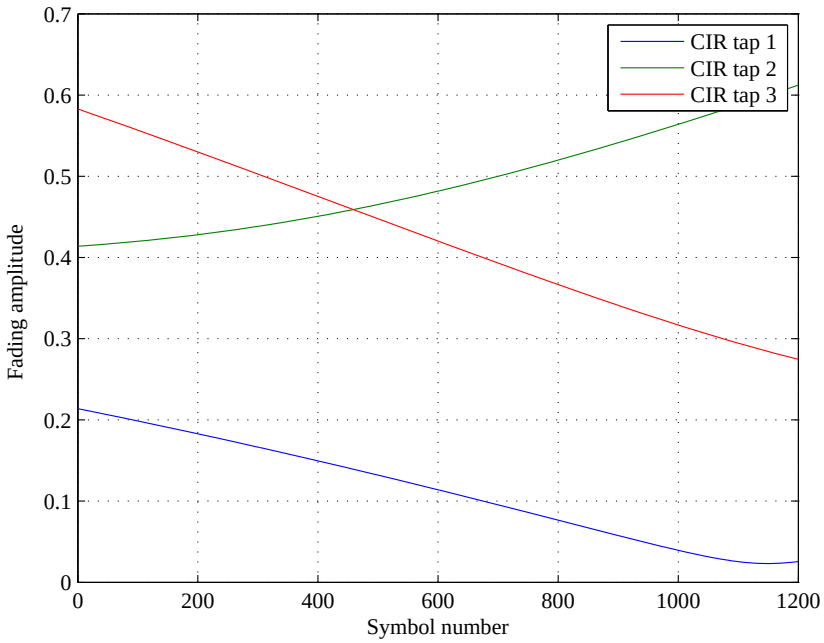


Figure A.8: Uncorrelated fading vectors for no frequency hopping.

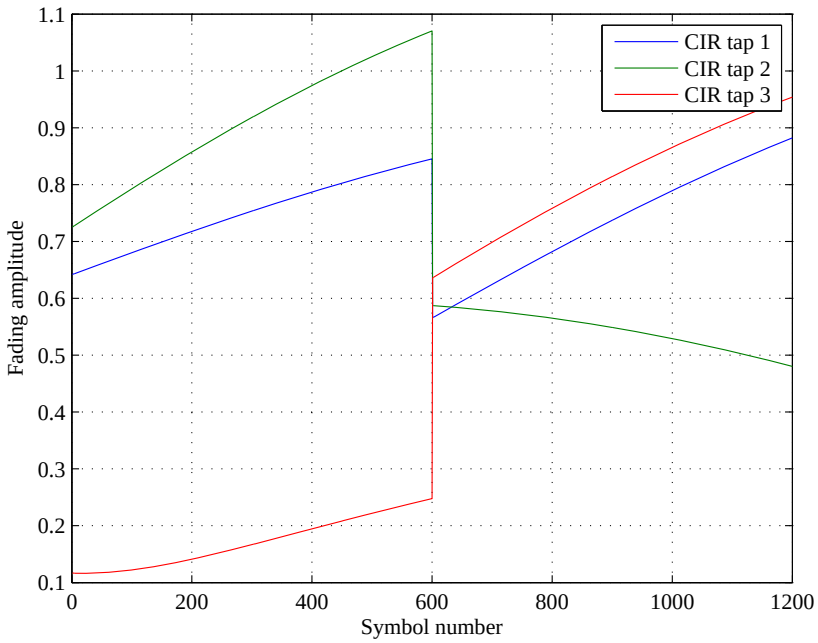


Figure A.9: Uncorrelated fading vectors for frequency hopping ($F = 2$).

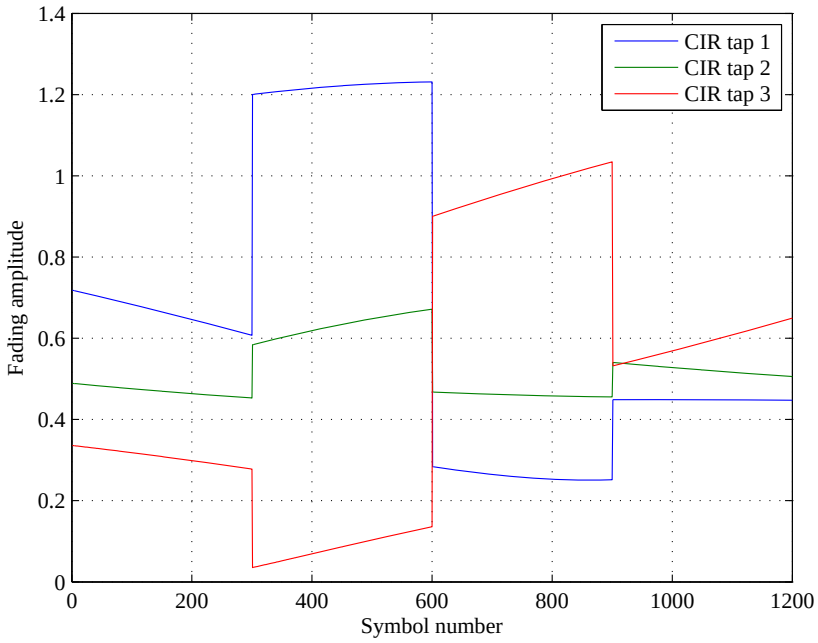


Figure A.10: Uncorrelated fading vectors for frequency hopping ($F = 4$).

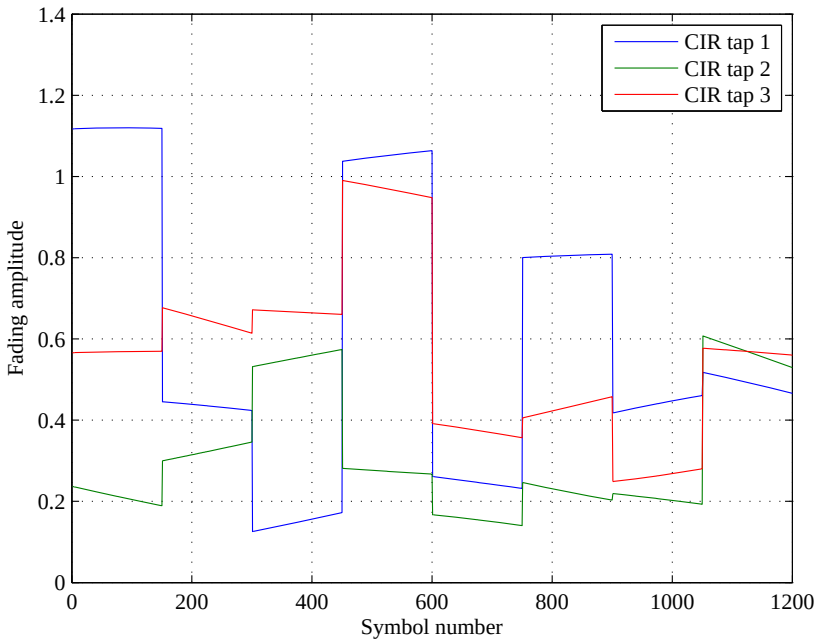


Figure A.11: Uncorrelated fading vectors for frequency hopping ($F = 8$).

A.4 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 invert the effect of the channel on the transmitted symbols effectively , 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.

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 [56, 57].

The channel estimator receives K ISI-corrupted pilot symbol observations $\mathbf{r} = \{r_1, r_2, ..., r_{K+1}\}^T$, where K 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

$$\mathbf{r} = \mathbf{Qh} + \mathbf{n}. \tag{A.19}$$

The matrix $\mathbf{Q}$ is fully populated by K 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

$$\mathbf{Q} = \begin{bmatrix} t_n & t_{n-1} & \cdots & t_{n-L+1} & t_{n-L} \\ t_{n+1} & t_n & \ddots & t_{n-L} & t_{n-L+1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ t_{n+(K-1)} & t_{n+(K-2)} & \ddots & t_{n-L+K} & t_{n-L+(K-1)} \\ t_{n+K} & t_{n+(K-1)} & \cdots & t_{n-L+K+1} & t_{n-L+K} \end{bmatrix} \tag{A.20}$$

To estimate the CIR $\mathbf{c}$, the squared error between $\mathbf{r}$ and $\mathbf{Qh}$ is

$$\varepsilon^2 = tr[(\mathbf{r} - \mathbf{Qh})^H (\mathbf{r} - \mathbf{Qh})] = \mathbf{n^H n}. \tag{A.21}$$

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

$$\frac{\partial \varepsilon^2}{\partial \mathbf{c}} = 2\mathbf{Q}^H(\mathbf{r} - \mathbf{Qh}). \tag{A.22}$$

Equating the gradient to zero produces the LS estimate

$$\tilde{\mathbf{c}} = (\mathbf{Q}^H \mathbf{Q})^{-1} \mathbf{Q}^H \mathbf{r}. \tag{A.23}$$

The pilot sequence must be chosen such that its autocorrelation approaches the *Delta Dirac* function. This will ensure that the Grammian Matrix $(\mathbf{Q}^H\mathbf{Q})$ is diagonally dominant and therefore invertible, which will simplify calculation of $\tilde{\mathbf{c}}$ in (A.23).

A.5 CONCLUDING REMARKS

In this appendix the most important aspects of the simulation environment used in this thesis were discussed, and it was also explained how the communication systems considered in this thesis will be simulated. Although the discussion is not exhaustive, the concepts relevant to this thesis were discussed at length.