ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Communication Technology Laboratory
Wireless Communications Group
Prof. Dr. A. Wittneben
Sternwartstrasse 7 ETH Zentrum Zurich
Tel +41 1 632 2788 Fax +41 1 632 1209

Student Project at the Communication Technology Laboratory
SS 2003

Combination of Space-Time Codes with Forward Error Correction (FEC)

Tobias Koch and Till Quack
4 July 2003

Advisors: Ingmar Hammerström and Marc Kuhn
Professor: Armin Wittneben

Acknowledgements

This project was possible thanks to the assistance of many people: We thank Ingmar Hammerström and Marc Kuhn for their kind help and support during the last fourteen weeks. We also want to thank Pof. Dr. A. Wittneben to give us the opportunity to carry out this project in the Wireless Communications Group.

Zurich, July 4, 2003

Till Quack

Tobias Koch

Abstract

Future wireless communication systems will demand high data rates for delivery of data like multimedia content. The use of MIMO systems with multiple transmit and receive antennas enables the introduction of space-time coding to improve link reliablity and spatial multiplexing to increase spectral efficiency. Recently a class of codes was proposed in [14] which makes it possible to combine space-time coding and spatial multiplexing. The proposed design for the LSD codes includes an optional forward error correction (FEC) element. The goal this work is to investigate the combination of LSD codes with FEC.

The results of our simulations show that the combination of LSD codes with FEC yields a significant improvement in terms of a lower bit error rate especially if a weak FEC with a high data rate is performed. In addition, if follows that in order to increase spectral efficiency spatial multiplexing outperforms the choice of a higher symbol-alphabet and diversity. However, the combination of diversity and a higher FEC code-rate even outperforms spatial multiplexing in terms of link reliability.

LSD codes and FEC involve a large number of parameters and, consequently, a wide range of optimization. Therefore, this work only covers a small part of the problem. However, the results raise suspicion that the combination of LSD codes with FEC is a reasonable tool in order to improve link reliability and spectral efficiency.

Contents

Chapter 1

Introduction

Future wireless communication systems will demand high data rates for delivery of data like multimedia content. The use of MIMO systems (multiple input multiple output) with multiple transmit and/or receive antennas is one technology to support these high data rates in a rich-scattering wireless channel [4].

The use of multiple transmit and receive antennas enables the introduction of space-time coding to improve link reliability and spatial multiplexing to increase spectral efficiency. Space-time codes introduce spatial and temporal correlations into the signal to benefit from diversity. Gain from diversity can be achieved either in the sender (*transmit diversity*), the receiver (*receive diversity*) or in both, sender and receiver. Spatial multiplexing increases the data rate at constant bandwidth by opening several data streams over the different antennas.

Recently, a class of codes was proposed in [14] which makes it possible to combine space-time coding and spatial multiplexing. These *linear scalable dispersion (LSD) codes* are designed in a way that allows to flexibly trade off between improved link reliability (i.e. diversity) and increased spectral efficiency (i.e. spatial multiplexing). The proposed design for the LSD codes includes an optional forward error correction (FEC) element as shown in Figure 1.1. So far, no research has been done on the effects of this FEC block on the performance of the whole system. Thus, in this work we present the results we obtained by combining LSD codes with several well-known FEC codes. The combination of space-time codes is motivated by higher performance. Questions that arise are for example at what cost in terms of complexity or if in general the combination of space-time codes and FEC is useful.

In Chapter 2 we discuss the theory of space-time processing in general and LSD codes in particular. Chapter 3 describes the FEC codes we used during our project. Interference

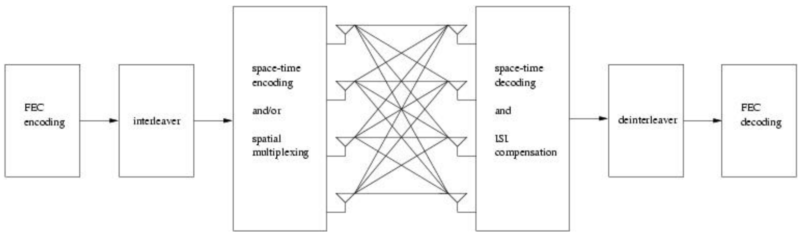
FEC encoding
interleaver
space-time encoding and/or spatial multiplexing
space-time decoding and ISI compensation
deinterleaver
FEC decoding

Figure 1.1: *Considered MIMO-system*

compensation and decoding are discussed in Chapter 4. The computational complexity of the different FEC techniques and other parts of the system is presented in Chapter 5. Chapter 6 contains the results from our simulations and finally, Chapter 7 summarizes our work with conclusions and outlook.

Chapter 2

Space-Time Processing

When data is transmitted in wireless environments it can be corrupted by effects like fading or interference. Fading is a result of the different paths the signal travels between sender and receiver. When sender or receiver move, each transmitted signal on the different paths gets a different phase-shift which leads to (constructive or destructive) superposition at the receiver. In the following we give a short introduction to techniques that combat fading in order to improve link reliability and/or exploit fading in order to improve spectral efficiency.

In Section 2.1 space-time codes are discussed. Section 2.2 gives a short introduction to spatial multiplexing. In Section 2.3 a new class of codes which makes it possible to combine space-time coding and spatial multiplexing, the so called *linear scalable dispersion (LSD) codes* are presented.

2.1 Space-Time Coding

To combat fading and thus improve link reliability diversity can be used. The basic principle of diversity can be described as follows: Several “copies” of the information signal are transmitted on channels with independent fading. Thus, the probability that we can obtain at least one signal with decent quality.

Some examples for diversity are:

Temporal diversity: Combination of channel coding and time interleaving. The signal is spread over time, i.e. the copies are transmitted in different time-slots. For example a repetition code.

Frequency diversity: Waves transmitted on different frequencies induce different multipath characteristics in the propagation media. Thus, redundancy in the frequency domain is obtained. The signal is spread over frequency, examples can be found in spread-spectrum technologies like CDMA.

Spatial diversity: Spatial diversity is obtained by using several antennas, if the antennas are separated far enough from each other to provide independent fading channels.

Multiple antennas can be used for diversity at the receiver and/or the sender. In the first case we speak of *receive diversity* in the latter of *transmit diversity.* Today, most systems are built with one antenna at the mobile station and multiple antennas at the basestation since implementing multiple antennas in the mobile station is technically rather complex. This means that in the uplink of the system receive diversity can be used, in the downlink transmit diversity.

While receive diversity is fairly well known, we concentrate on transmit diversity in the following. For transmit diversity two scenarios are possible: Either channel state information is available or not.

If the channel is perfectly known at the sender, its influence can be compensated in the signal *before* transmitting it such that the signal obtained at the receiving end of the channel is demodulated perfectly.

More realistic but also more challenging is the case of the unknown channel. Here the spatial and temporal diversity comes into play and we can pre-process the data for the different transmit antennas with so called space-time codes.

The code is defined by its transmission matrix. The generic transmission matrix for transmitting k input symbols $\alpha_1, \alpha_2, \ldots, \alpha_k$ using N_{TX} transmit antennas and N_{dim} time-slots is defined as

$$\mathbf{G} = \begin{pmatrix} g_{11} & g_{21} & \cdots & g_{N_{TX}1} \\ g_{12} & g_{22} & \cdots & g_{N_{TX}2} \\ \vdots & \vdots & \vdots & \vdots \\ g_{1N_{dim}} & g_{2N_{dim}} & \cdots & g_{N_{TX}N_{dim}} \end{pmatrix} \tag{2.1}$$

The entries g_{ij} of the transmission matrix $\mathbf{G}$ are linear combinations of the input symbols $\alpha_1, \alpha_2, \ldots, \alpha_k$ and their conjugates. For example in time-slot $j = 2$ signals $g_{12}, g_{22}, \ldots, g_{N_{TX}2}$ are transmitted simultaneously from transmit antennas $Tx1, Tx2, \ldots, TxN_{TX}$. The transmission matrix defines encoding in time and space, hence the term space-time coding.

A very elegant example for a space-time code is the two-transmitter scheme proposed by Alamouti [1]. To present this code we follow closely [10]. In this particular case the transmission matrix is defined as

$$\mathbf{G_2} = \begin{pmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{pmatrix} \tag{2.2}$$

The code uses $N_{TX} = 2$ antennas over $n = 2$ time-slots to transmit the $k = 2$ input symbols x_1 and x_2. This means that the code-rate $r = k/n$ is one.

In any time-slot j two signals are simultaneously transmitted over the antennas $Tx1$ and $Tx2$: In one time-slot signal x_1 is transmitted from antenna $Tx1$ and x_2 is transmitted from $Tx2$, in the next time-slot the signals $-x_2^*$ and x_1^* are transmitted from $Tx1$ and $Tx2$, respectively.

To explain the decoding process we assumme one receive antenna, further we also assume that the complex fading envelope is constant across the two consecutive timeslots:

$$h_1 = h_1(T = 1) = h_1(T = 2) \tag{2.3}$$

$$h_2 = h_2(T = 1) = h_2(T = 2) \tag{2.4}$$

Note that the complex fading envelopes h_1 and h_2 are known in the receiver - they can be obtained from a training sequence for example.

In the receiver we obtain

$$y_1 = h_1 x_1 + h_2 x_2 + n_1 \tag{2.5}$$

$$y_2 = -h_1 x_1^* + h_2 x_2^* + n_2 \tag{2.6}$$

where y_1 is the first received signal, y_2 the second and n_1 and n_2 are independent noise samples added by the receiver.

Both signals y_1 and y_2 are passed to the combiner depicted in Figure 2.1 to extract x_1 and x_2. In order to extract the signal x_1, the received signals y_1 and y_2 are combined according to

$$\begin{aligned} \tilde{x}_1 &= h_1^* y_1 + h_2 y_2^* \\ &= h_1^* h_1 x_1 + h_1^* h_2 x_2 + h_1^* n_1 - h_2 h_1^* x_2 + h_2 h_2^* x_1 + h_2 n_2^* \\ &= (|h_1|^2 + |h_2|^2) x_1 + h_1^* n_1 + h_2 n_2^* \end{aligned} \tag{2.7}$$

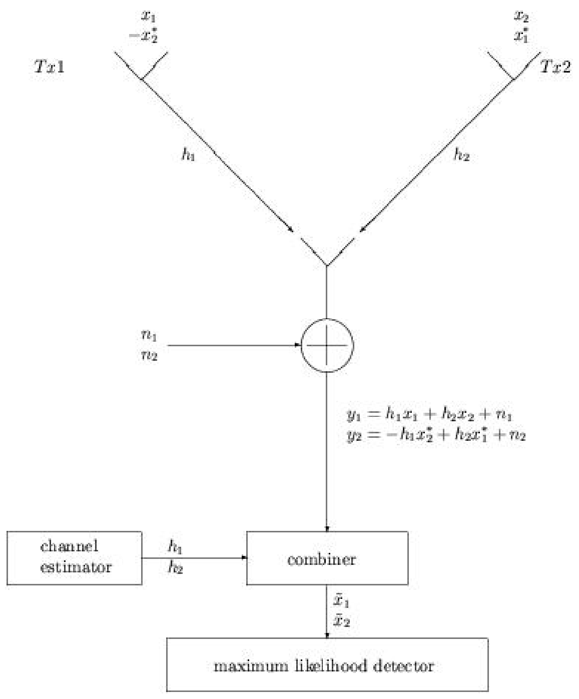

Figure 2.1: *Twin transmitter space-time code by Alamouti*

and

$$\begin{aligned}
\tilde{x}_2 &= h_2^* y_1 - h_2 y_2^* \\
&= h_2^* h_1 x_1 + h_2^* h_2 x_2 + h_2^* n_1 + h_1 h_1^* x_2 - h_1 h_2^* x_1 - h_1 n_2^* \\
&= (|h_1|^2 + |h_2|^2) x_2 + h_2^* n_1 + h_1 n_2^* \qquad (2.8)
\end{aligned}$$

It can be seen from (2.7) and (2.8) that x_1 and x_2 have been separated by simple multiplications and additions - because of the orthogonality introduced in $\mathbf{G_2}$ x_2 and x_1 are cancelled out in (2.7) and (2.8), respectively. Thus, signals x_1 and x_2 are separated, $\tilde{x}_1$ and $\tilde{x}_2$ depend only on constructive interference and an additive noise term.

2.2 Spatial Multiplexing

When using multiple antennas at the sender *and* the receiver as depicted in Figure 1.1, they can not only be used to obtain diversity gain, but also to enable spatial multiplexing. In a rich scattering environment (i.e. the fading coefficients are independent), the use of multiple transmit and receive antennas makes it possible to operate several parallel

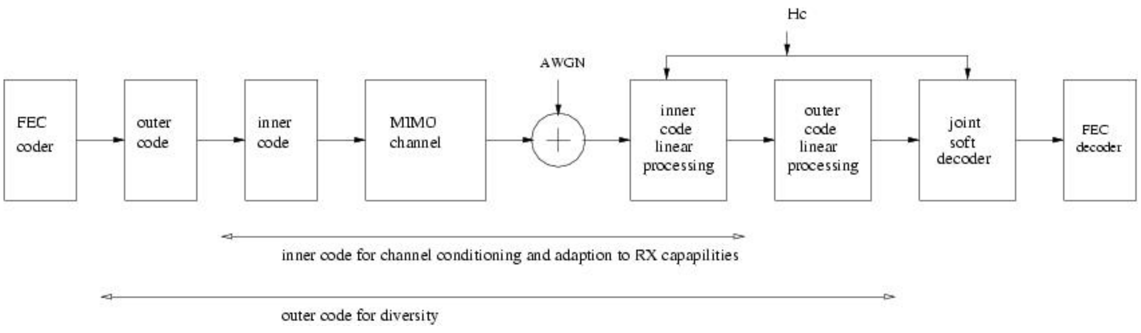
Hc
AWGN
FEC coder
outer code
inner code
MIMO channel
inner code linear processing
outer code linear processing
joint soft decoder
FEC decoder
inner code for channel conditioning and adaption to RX capapilities
outer code for diversity

Figure 2.2: *LSD code overview*

data streams in the same frequency band - the only cost for this increase in capacity is the cost of the additional antennas, in particular, no extra bandwidth is needed. The symbol stream at the sender is broken up into several parallel symbol streams which are transmitted simultaneously and in the same frequency band from the antennas. Each receive antenna observes a superposition of these symbol streams, i.e. spatial multiplexing leads to intersymbol interference (ISI). Since each transmit antenna induces a different spatial pattern at the receiver due to multipath propagation, the individual data streams can be separated again.

For a detailed description of the receivers we refer to Section 4.

2.3 Linear Scalable Dispersion (LSD) Codes

When using a MIMO system with multiple antennas, it is possible to use them either for spatial multiplexing (i.e. subchanneling) to improve the data rate or for diversity to improve link quality. As it is mentioned in the introduction, the new class of codes introduced in [14] makes it possible to jointly utilize spatial multiplexing and diversity. Given a number of antennas at the transmitter and the receiver, the LSD code is designed to choose the number of antennas for spatial multiplexing and the number of antennas for diversity in a flexible way.

The code consists of a concatenation of a time variant inner linear code and a time invariant outer linear code. As an additional element an FEC code can be added. The linearity of the inner and outer codes preserves the mean capacity of the MIMO channel H_ν and makes the use of a variety of low complexity decoders possible. As it can be seen in Figure 2.2 all three code designs are decoupled which allows the adaption to the heterogeneous conditions in the environment. To be precise, the outer code consists of

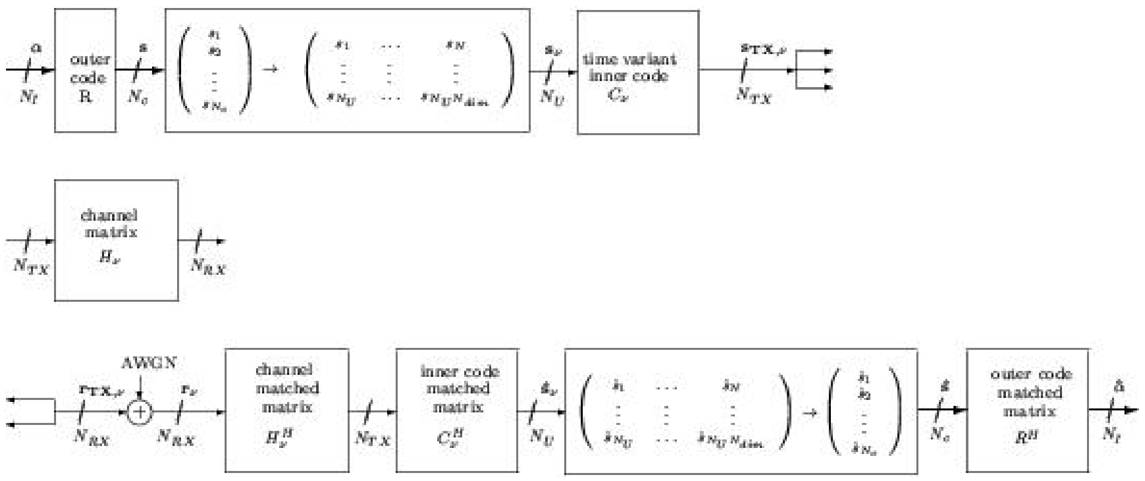

Figure 2.3: *LSD code overview*

a unitary matrix and is optimized for diversity in flat fading. In terms of diversity the optimal choice for the outer code would be the repetition code. However, the rate of repetition code is very low. Thus, the outer code in the LSD code is designed to be close to the performance of the repetition code in terms of diversity, but with substantially higher code rate. Also, unlike for example the Alamouti code introduced in Subsection 2.1 the outer code is not orthogonal.

The inner code is used for channel conditioning and adaption to RX capabilities.

Figure 2.3 gives a more detailed overview over the signals, parameters and dimensions used. The input symbol vector $\boldsymbol{\alpha}$ is multiplied with the outer code matrix $\mathbf{R}$ to form the transmit symbol vector $\mathbf{s}$. The dimensions of $\boldsymbol{\alpha}$ and $\mathbf{s}$ define the code rate N_I/N_c.

The transmit symbol vector $\boldsymbol{s}$ of dimension $(N_C \times 1)$ is reshaped into a $(N_u \times N_{dim})$-matrix, where N_u is the number of symbols transmitted per time-slot and N_{dim} is the number of time-slots used. The N_{dim} columns of this matrix are the consecutive input vectors $\mathbf{s}_\nu$ for the linear code C_ν, where $\nu \leq N_{dim}$ is the time index.

The inner code produces the codewords $\mathbf{s}_{\mathbf{TX},\nu}$, which are transmitted over the Multiple Input Multiple Output (MIMO) channel H_ν. Usually H_ν contains the channel coefficients between each pair of TX and RX antennas. Further, we assume Rayleigh fading, i.e i.i.d. complex normal coefficients of H_ν (flat fading). The received signal is disturbed by additive white Gaussian noise (AWGN). Multiplication with H_v^H and the inner code matched matrix C_v^H yields a sufficient statistics for the estimation $\hat{\mathbf{s}}_\nu$ of the inner code input vector $\mathbf{s}_\nu$. Since the system transmits N_u symbols in one temporal dimension, N_u corresponds to the number of spatial subchannels to be used.

A detailed description of the inner and outer code can be found in [14]. We will continue with the integration of the FEC codes.

Chapter 3

Forward Error Correction

Forward error correction (FEC) coding is a type of digital signal processing that improves data reliability by introducing a known structure into a data sequence prior to transmission. This structure allows the decoder to detect and possibly correct errors caused by corruption from the channel and the receiver. As the name implies, this coding technique enables the decoder to correct errors without requesting retransmission of the original information.

In our project, we investigated Bose-Chaudhuri-Hocquenghem (BCH) as well as convolutional codes. This chapter can be outlined as follows: In Section 3.1 a review of BCH codes including a short introduction to linear block codes is given. Then, in Section 3.2 convolutional codes are discussed.

3.1 Bose-Chaudhuri-Hocquenghem Codes

The class of Bose-Chaudhuri-Hocquenghem (BCH) codes is a large class of multiple-error-correcting codes that occupies a prominent place in the theory and practice of error correction. One reason for this prominency is that given not too large block-lengths, there are good codes in this class. Another reason is the knowledge of relatively simple and instrumentable encoding and decoding techniques [3].

BCH codes belong to the class of linear block codes or, to be more specific, to the subclass of cyclic codes. In the following an introduction to linear block codes and cyclic codes is given. Then a more precise definition of the class of BCH codes is presented.

3.1.1 A Short Introduction to Linear Block Codes

[1] A block code consists of a set of vectors with N elements, where the vectors are called *codewords* and N is called the length of the codeword. The elements of a codeword are selected from an alphabet of q symbols. If the alphabet consists of the two symbols 1 and 0, the code is a binary code.

A block code maps k information bits into a codeword of length N and the ratio $r = k/N$ is defined to be the *rate* of the code. In general, there exist q^k different codewords. (Remember that q is the number of symbols from which the elements of the codeword are selected.)

The encoding and decoding functions involve the arithmetic operations of addition and multiplication performed on codewords. These operations are performed according to the conventions of the algebraic field whose elements are the symbols contained in the alphabet. A field consists of a set of elements which has the two arithmetic operations addition and multiplication satisfying certain properties, named in Appendix A.

As stated before, the elements of a codeword are selected from an alphabet of q symbols. Consequently, codes are constructed from fields with q elements. In coding, q is usually a finite number, so the field is a finite field or a so called *Galois field.* A Galois field with q elements is denoted by $\mathrm{GF}(q)$.

When $q = p^m$, where p is a prime and m is any positive integer, it is possible to extend the field $\mathrm{GF}(p)$ to the field $\mathrm{GF}(p^m)$. This is called the extension field of $\mathrm{GF}(p)$. Note that $\mathrm{GF}(p)$ is contained in $\mathrm{GF}(p^m)$. The theory of finite fields is not discussed in this work. For further informations we refer to literature (e.g. [3], [12]).

Let $\boldsymbol{x} = (x_{k-1}, x_{k-2}, \ldots, x_1, x_0)$ be the vector of the k information bits. Then the encoding operation of a linear block code $\boldsymbol{c} = (c_{N-1}, c_{N-2}, \ldots, c_1, c_0)$ can be represented by multiplying the message $\boldsymbol{x} = (x_{k-1}, x_{k-2}, \ldots, x_1, x_0)$ with the so called *generator matrix* $\mathbf{G}$, i.e.

$$\boldsymbol{c} = \boldsymbol{x}\mathbf{G}. \tag{3.1}$$

Similarly, for every codeword $\boldsymbol{c}$ exists a *parity check matrix* $\mathbf{H}$, such that

$$\boldsymbol{c}\mathbf{H}^T = 0. \tag{3.2}$$

It can be easily verified that

$$\mathbf{G}\mathbf{H}^T = 0. \tag{3.3}$$

Remember that the additions and multiplications in (3.1), (3.2) and (3.3) are performed in the Galois field $\mathrm{GF}(q)$.

[1]This introduction is based on [12].

As an example, we consider a binary $(7,4)$ hamming code that maps 4 information bits into a codeword of length 7. Note that all operations are performed mod 2, since the code is binary. For this specific code, the generator matrix $\mathbf{G}$ and the parity check matrix $\mathbf{H}$ are defined as:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Assume that the 4 information bits are $\boldsymbol{x} = (1101)$. Then the codeword $\boldsymbol{c}$ can be computed according to (3.1). It follows $\boldsymbol{c} = (1101001)$. It can easily be verified that $\boldsymbol{c}\mathbf{H} = 0$.

A subclass of the class of linear block codes are cyclic codes. Cyclic codes satisfy the cyclic shift property: If $\boldsymbol{c} = (c_{N-1}, c_{N-2}, \ldots, c_1, c_0)$ is a codeword, then $\tilde{\boldsymbol{c}} = (c_{N-2}, c_{N-3}, \ldots, c_0, c_{N-1})$ is also a codeword. This structure can be used in order to design encoding and decoding operations.

For convenience, cyclic codes are often represented by a polynomial description. In this case, $\boldsymbol{c}$ can be written as

$$c(p) = c_{N-1}p^{N-1} + c_{N-2}p^{N-2} + \cdots + c_1 p + c_0, \tag{3.4}$$

where the coefficients are elements of the Galois field. Instead of the matrix multiplications in (3.1), (3.2) and (3.3), polynomial multiplications mod $\left(p^N - 1\right)$ in the galois field are performed. A cyclic shift becomes

$$\tilde{c}(p) = p\,c(p) \bmod \left(p^N - 1\right) = c_{N-2}p^{N-1} + c_{N-3}p^{N-2} + \cdots + c_0 p + c_{N-1}. \tag{3.5}$$

According to (3.1) the polynomial of a codeword $c(p)$ can be determined by the *message polynomial* $x(p) = x_{k-1}p^{k-1} + x_{k-2}p^{k-2} + \ldots + x_1 p + x_0$ and the *generator polynomial* $g(p) = g_{N-k}p^{N-k} + g_{N-k-1}p^{N-k-1} + \ldots + g_1 p + g_0$ in a way that

$$c(x) = x(p)g(p). \tag{3.6}$$

Furthermore, there exists a *parity polynomial* $h(p)$. The parity polynomial and the generator polynomial fulfil

$$g(p)h(p) = p^N - 1 \triangleq 0, \tag{3.7}$$

where $\triangleq$ means that $P^N - 1 \bmod \left(p^N - 1\right)$ is equivalent to 0.

It is possible to determine the zeros $\gamma_1, \ldots, \gamma_r \in \mathrm{GF}(q^m)$ of a codeword polynomial $c(p)$, such that

$$c(\gamma_i) = 0, \text{ for } i = 1, \ldots, r. \tag{3.8}$$

This will be important for the definition of the BCH codes.

3.1.2 BCH Codes

Assume that $p^N - 1$ can be written as

$$p^N - 1 = f_1(p) f_2(p) \cdots f_s(p). \tag{3.9}$$

Because of (3.7), the generator polynomial $g(p)$ must be the product of a subset of these polynomials:

$$g(p) = f_1(p) f_2(p) \cdots f_r(p). \tag{3.10}$$

This leads us to the definition of BCH codes:

Definition 3.1 *A BCH code is the cyclic code of block-length*[2] $N = q^m - 1$ *with the generator polynomial*

$$g(p) = \mathrm{LCM}\left[f_1(p), f_2(p), \ldots, f_{2t}(p)\right]^{3} \tag{3.11}$$

where $f_j(p)$ *is the polynomial with zero* $\beta^j \in \mathrm{GF}(q^m)$ *(i.e.* $f_j(\beta^j) = 0$*) and* t *is the number of correctable errors.*

As an example we consider a binary BCH code that maps 4 information bits to a codeword of block-length 7. The number t of correctable errors is 1. Note that all operations are performed mod 2, since the code is binary. This specific code has a generator polynomial $g(p) = p^3 + p^2 + 1$. For a message $\boldsymbol{x} = (1001)$ the message polynomial $x(p)$ becomes $x(p) = p^3 + 1$. According to (3.6) the codeword polynomial $c(p)$ for this message is:

$$c(p) = (p^3 + 1)(p^3 + p^2 + 1) = p^6 + p^5 + p^2 + 1$$

and the corresponding codeword can be written as $\boldsymbol{c} = (1100101)$.

At this point we are done with the introduction to BCH codes. In Subsection 4.2.1 a decoding procedure of BCH codes is presented.

[2]In this case, the block-length is called to be primitive.

[3]LCM means *Least Common Multiple.*

3.2 Convolutional Codes

A (n,k) convolutional code is generated using a linear finite state shift register as depicted in Figure 3.1. In general, the shift register consist of K k-bit stages and n linear algebraic function generators. The binary input data is shifted into and then along the register - k bits at a time. This way, for each k-bit input sequence a n-bit output sequence is generated by the encoder so the rate is $r = k/n$. The parameter K is called the constraint length of the convolutional code. Two methods for describing convolutional codes will be

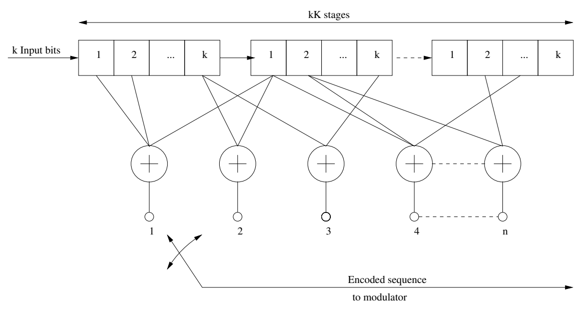

Figure 3.1: *Convolutional encoder [12]*

presented: Generator vectors and the trellis representation.
Block codes can be represented with so called generator matrices, as it is described in section 3.1. Since the input to a convolutional code is semi-infinite, these matrizes would be of semi-infinite size, too. As an alternative a convolutional code can be represented by a functionally equivalent representation as a set of n vectors, one for each function generator. Each vector has Kk dimensions and specifies the connections between the function generators and the stages of the encoder, i.e. 1 stands for an existing connection, 0 means there is no connection between the specific stage and function generator. As

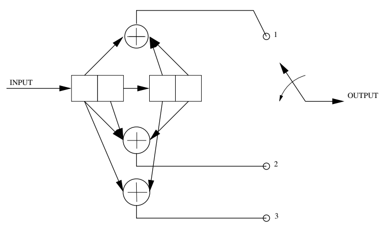

Figure 3.2: $n = 3, k = 2, K = 2$ *convolutional encoder*

an example consider the rate 2/3 convolutional encoder ($n = 3, k = 2, K = 2$) in Figure 3.2. In this encoder, $k = 2$ bits at a time are shifted into it, and $n = 3$ output bits are generated with the $K = 2$ stages. We suppose the registers are both in the all zero state

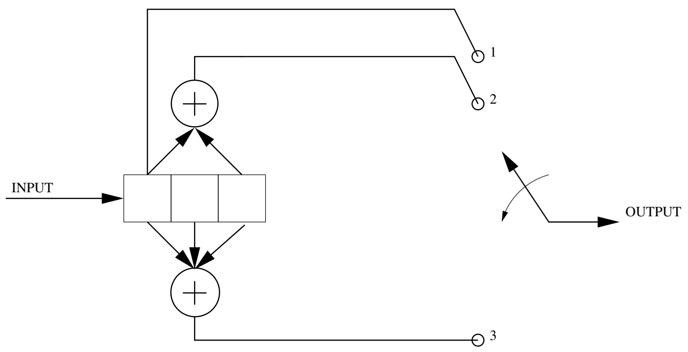

Figure 3.3: *n=3,k=1,K=3 convolutional encoder*

at the beginning. The 2-bit input sequences can be 00, 01, 10 or 11. The resulting output bits are 000, 010, 111, 101.

Further, suppose we number the outputs from top to bottom. Then we can represent each output as one of the n vectors described above, i.e. the so called generators for the code are

$$\boldsymbol{g1} = [1011], \boldsymbol{g2} = [1101], \boldsymbol{g3} = [1010]$$

Often, the generators are written in octal form for convenience. For the above example this results in (13,15,12).

In a trellis diagram, the states the shift register can take, are denoted by nodes, the transitions between the states are represented by edges between these nodes. In Figure 3.4 a $K = 3$, rate $r = 1/3$ encoder and the corresponding trellis diagram are shown. A label on the edges shows the input and output bit(s) for each transition in the form input/output. In the beginning there is a transient phase (of K-1 steps) which represents the initial “filling” of the register from the all zero state. After that the steady state is reached and every input can lead only to one of the existing states again. (Since there is a limited number of bits in the register, there is a limited number of possible states, namely $2^{k(K-1)}$). For example, if the first bit is 1, 111 is generated as output and the register contains the values 100 which defines the current state. I we proceed from there and again shift a 1 into the register it contains 110 now and the ouput is 110, too. Now we shift a zero into the register, the new state is 011 and the output is 010. We have reached the steady state now, from here the structure starts repeating itself.

In Section 4.2.2 we will show that the trellis representation is very useful for decoding convolutional codes.

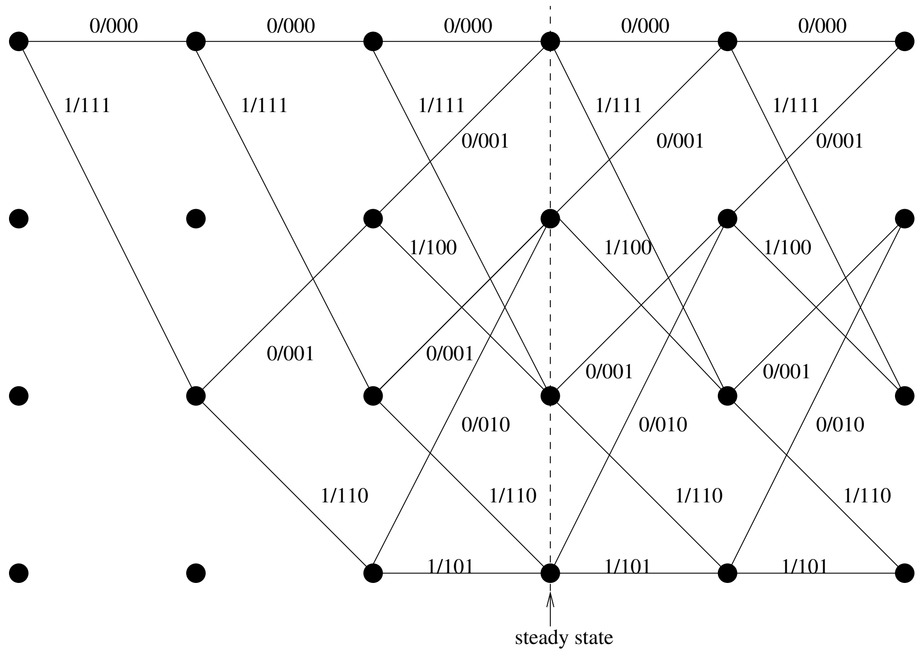

Figure 3.4: *Trellis for the encoder above*

Chapter 4

Interference Compensation and Decoding

The main focus of the last two chapters was on the transmitter side. Techniques like space-time coding, spatial multiplexing and FEC improving link reliability and spectral efficiency were presented.

In this chapter we focus on the receiver side. We view the receiver as a cascade of *equalizer* and *channel decoder*. First the equalizer compensates inter symbol interference (ISI) introduced by the channel on the one hand and space-time coding respectively spatial multiplexing on the other hand. Then the channel decoder decides on what signal has been transmitted using the knowledge of the redundancy introduced by the FEC.

This chapter will be outlined as follows: First some equalization techniques including the so called MAP-DFE according to [8] are presented. Then the decoding of BCH codes is investigated, including a short introduction to decoding of linear block codes. The chapter terminates with the description of a decoder for convolutional codes.

4.1 Equalization and the MAP-DFE

In order to explain the components of the decoder we consider a communication system consisting of transmitter, channel and receiver. To be specific, the transmitted symbol passes through the following stages: First FEC is performed. Then, the symbols are interleaved, encoded by the LSD encoder and transmitted over the fading channel. The receiver multiplies the sequence with the so called channel matched matrix, performs LSD decoding and feds the resulting sequence into the equalizer. In the following, we combine

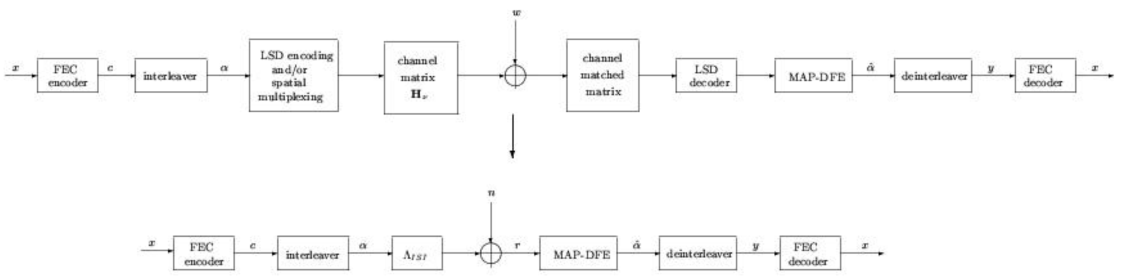

Figure 4.1: *System consisting of FEC and LSD coding*

the influences of LSD encoder, the channel, the multiplication with the channel matched matrix and the LSD decoder in one matrix Λ_{ISI}. Figure 4.1 shows the communication system described above.

The input sequence at the MAP-DFE can be written as

$$\boldsymbol{r} = \Lambda_{ISI}\boldsymbol{\alpha} + \boldsymbol{n}. \tag{4.1}$$

$\boldsymbol{n}$ is the noise after multiplication with the channel matched matrix and LSD decoding. Consequently, $\boldsymbol{n}$ is in general colored. $\boldsymbol{\alpha}$ denotes the signal vector at the output of the interleaver. $\boldsymbol{r}$ names the N_c-dimensional vector at the input of the MAP-DFE. Λ_{ISI} concludes the influence of the LSD codes and the channel.

The LSD codes can lead to ISI due to an optimized diversity performance and spatial multiplexing (i.e. Λ_{ISI} is not diagonal). Since this could heavily affect the performance, ISI compensation is an important task of the decoding process. The component in the receiver that performs ISI compensation is called *equalizer*.

In the following we give an overview over some equalization techniques. Then the so called *MAP-DFE* according to [8] is considered.

4.1.1 Overview over some Equalization Techniques

The receiver which is optimal in the sense that it minimizes the probability of error is the *maximum likelihood* (ML) receiver. However, for large block-lengths this receiver has a very high decoding complexity and is, therefore, only useful for small block-lengths. Thus, only suboptimal equalization methods are possible. These can roughly be classified in linear methods and successive interference cancellation methods, which again can be divided in serial and parallel methods.

Linear methods like *zero forcing (ZF)* or *minimum mean-square error (MMSE)* detectors perform ISI compensation applying a linear transformation (i.e. multiplication with a matrix) to the received vector. Successive interference cancellation methods like V-BLAST [5] or other *decision feedback equalizers (DFE)* first estimate a certain number of symbols. After that the influence of the estimated symbols on the remaining symbols is compensated using channel knowledge at the receiver. Then the next symbols are estimated and so on.

In the following we discuss some of the methods mentioned above.

4.1.1.1 Zero Forcing Equalizer

The zero forcing (ZF) equalizer inverts the channel transfer matrix. Given (4.1), the estimation of $\boldsymbol{\alpha}$ is:

$$\hat{\boldsymbol{\alpha}} = \Lambda_{ISI}^{-1}\boldsymbol{r}. \tag{4.2}$$

Note that it is possible to compute Λ_{ISI}^{-1}, since Λ_{ISI} is quadratic. This equalizer perfectly compensates the ISI. However, if the channel has a small gain, then there will be significant noise enhancement. Therefore, the ZF receiver has only an adequate performance in the high SNR regime.

4.1.1.2 Minimum Mean-Square Error Equalizer

The minimum mean-square error (MMSE) equalizer minimizes the error due to interference and noise. The estimate of $\boldsymbol{\alpha}$ is obtained according to

$$\hat{\boldsymbol{\alpha}} = \mathbf{G}_{MMSE}\boldsymbol{r}, \tag{4.3}$$

where $\mathbf{G}_{MMSE}$ is

$$\mathbf{G}_{MMSE} = \left(\Lambda_{ISI}\Lambda_{\boldsymbol{\alpha}\boldsymbol{\alpha}}\Lambda_{ISI}^{H} + \Lambda_{\boldsymbol{nn}}\right)^{-1}\Lambda_{ISI}\Lambda_{\boldsymbol{\alpha}\boldsymbol{\alpha}}. \tag{4.4}$$

$\Lambda_{\boldsymbol{\alpha}\boldsymbol{\alpha}}$ is the correlation matrix of the transmitted symbol vector $\boldsymbol{\alpha}$ and $\Lambda_{\boldsymbol{nn}}$ is the correlation of the added noise $\boldsymbol{n}$. In our case $\Lambda_{\boldsymbol{\alpha}\boldsymbol{\alpha}} = \mathbf{I}$, which means that the input symbols are uncorrelated. Then (4.4) becomes

$$\mathbf{G}_{MMSE} = \left(\Lambda_{ISI}\Lambda_{ISI}^{H} + \Lambda_{\boldsymbol{nn}}\right)^{-1}\Lambda_{ISI}. \tag{4.5}$$

Compared to the ZF equalizer the MMSE receiver is less sensitive to noise. In the high SNR regime the MMSE equalizer converges to the ZF equalizer.

4.1.1.3 V-BLAST

The V-BLAST is a serial successive interference cancellation method. In contrast to the ZF and the MMSE equalizer, the V-BLAST first decodes the "strongest" signal, then subtracts it from the received signal vector $\boldsymbol{r}$, proceeds to decode the strongest signal of the remaining signals and so on. The strongest signal is determined based on the knowledge of the channel gains. (We do not discuss the equalization process of the V-BLAST more in detail and refer to [5] for more specific informations). In the following subsection we present an equalizer whose decoding procedure is similar to the V-BLAST but determines the "strongest" signal based on the a posteriori probabilities.

4.1.2 The MAP-DFE

Like the V-BLAST presented above, the MAP-DFE is also a decision feedback equalizer. However, in contrast to the V-BLAST the MAP-DFE is able to decode more than one symbol in one iteration. The equalization procedure is similar to that of the V-BLAST, i.e. some symbols are decoded, then the influence of those symbols on the remaining symbols is cancelled, then the next symbols are decoded and so on. In contrast to the V-BLAST, the order of the cancellation process (i.e. the order of the symbols that are decoded) is determined based on the a posteriori error probabilities. The cancellation procedure goes as follows:

1. MMSE estimation of all symbols in consideration of Λ_{ISI}.
2. Determining the a posteriori error probabilities of all still not decoded symbols on the basis of the MMSE estimation.
3. Determining the segment of the symbols that will be jointly decoded on the basis of the a posteriori error probabilies.
4. Deciding the symbols of the segment of step 3.
5. Compensation of the interference of all symbols of step 4 on the remaining not decoded symbols.
6. Proceed to step 1 unless all symbols are decoded.

Figure 4.2 shows the compensation process described above. First an MMSE estimation is performed according to (4.5). Then, based on this estimation, the a posteriori error probabilities of the symbols are determined. Note that the error probabilities depends on

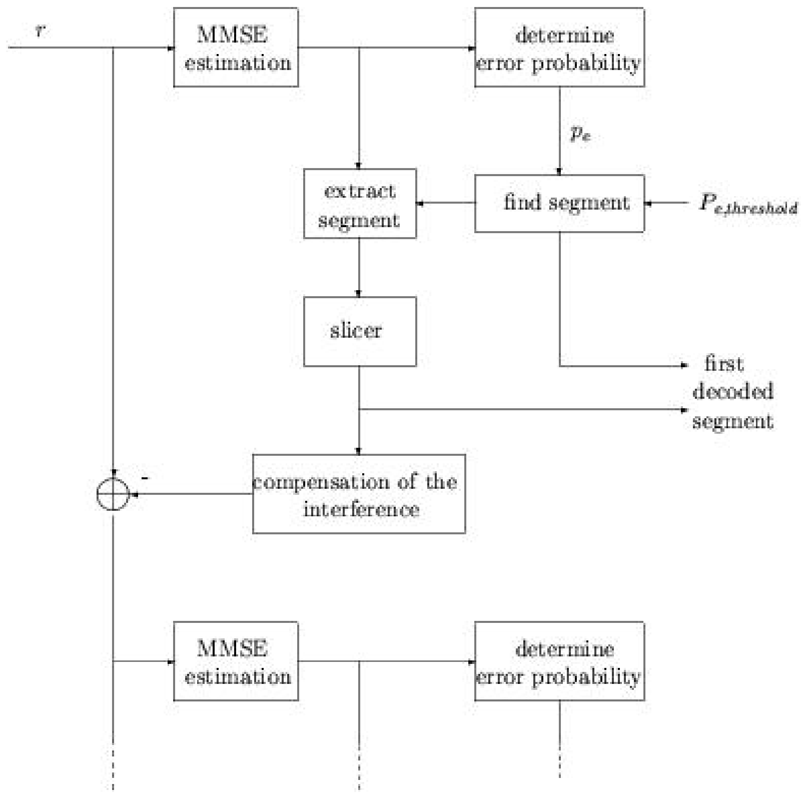

Figure 4.2: *Interference compensation process of the MAP-DFE*

the symbol alphabet used. Assume that $\boldsymbol{\alpha}$ is the vector of the estimated symbols. Then the a posteriori error probability of a received 2-PSK symbol is

$$P_{e,2-PSK} = \left(1 + e^{\frac{2|\alpha|\sqrt{E_b}}{\sigma_n^2}}\right)^{-1}, \tag{4.6}$$

where σ_n^2 is the variance of the noise after MMSE equalization, $|\alpha|$ is the absolute value of the estimated amplitude of the symbol and E_b is the energy per bit. In order to obtain (4.6) we approximated the noise with AWGN, although the noise is colored.

The probability of error for 4-QAM can be approximated as:

$$\begin{aligned} P_{e,4-QAM} &= (1 - P^{(I)}_{e,2-PSK} P^{(Q)}_{e,2-PSK}) \\ &= P^{(I)}_{e,2-PSK} + P^{(Q)}_{e,2-PSK} - P^{(I)}_{e,2-PSK} P^{(Q)}_{e,2-PSK}. \end{aligned} \tag{4.7}$$

For (4.7) we view 4-QAM as two 2-PSK signals I and Q (inphase and quadrature component) with statistically independent noise and independent error probabilities. (Of course, it is also possible to compute the error probabilities for higher symbol alphabets like 16-QAM or 64-QAM. We leave it to the reader to calculate those probabilities.)

After calculating these probabilities, a segment of estimated symbols is decoded parallelly. This segment consists of a variable number of symbols with an a posteriori error probability below a certain threshold $P_{e,threshold}$. If all symbols have a an error probability higher than this threshold, only the symbol with the lowest error probability is decoded. After that, the influence of these decoded symbols on the remaining symbols is canceled using the knowledge of Λ_{ISI}. Then again an MMSE estimation of the symbols is performed, followed by the computation of the a posteriori error probabilities of all remaining symbols. The process stops when all received symbols are decoded. As sketched in Figure 4.2 the error probabilities of the decoded symbols are available for further use at the receiver.

A characteristic of the MAP-DFE is that a variable number of symbols can be decoded parallelly. The idea is that if the probability of an error is small, then there is no need to decode only one symbol per iteration. Consequently, interference compensation can be done with less iterations which yields a lower complexity. (Note that in every iteration an MMSE estimation is performed which requires a matrix inversion.) The threshold $P_{e,threshold}$ determines the complexity of the decoder. If $P_{e,threshold}$ is large, many symbols are decoded parallelly and, hence, the the complexity is low. However, it is more likely to have an erroneous symbol. On the other hand, if $P_{e,threshold}$ is small, only a few symbols are decoded together which results in a higher complexity but leads to a lower error probability.

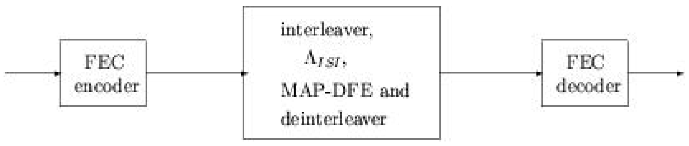

Figure 4.3: *System with FEC encoder and decoder*

It should be noted that for $P_{e,threshold} = 1$ the MAP-DFE only performs one iteration. In this case, the MAP-DFE becomes an MMSE equalizer. In contrast, if $P_{e,threshold} = 0$, only the symbol with the lowest error probability is decoded which yields the best performance in terms of error rate.

As mentioned above, there exists a tradeoff between performance (i.e. small error probability) and complexity. In Chapter 6 this tradeoff is investigated.

4.2 Channel Decoding

In the last section several equalization techniques and in particular the MAP-DFE are discussed. Now we consider the channel decoder. To be specific, we investigate decoders for both BCH and convolutional codes. For this purpose, the components between FEC encoder and FEC decoder are viewed as one channel that transmits symbols from the encoder to the decoder and introduces some errors. The resulting system is shown in Figure 4.3.

In order to keep track of our project, we use FEC decoders performing hard-decision. A hard-decision decoder maps a real-valued signal into a sequence of bits, where every bit is either 1 or 0. Consequently, information is lost. It is intuitively clear that one could reach a lower probability of error by considering all the information available. Therefore, hard-decision decoders are suboptimal.

The best way of performing hard-decision decoding is *minimum distance decoding.* In mininum distance decoding, the hamming distance between the received bit sequence and all possible codewords is computed. The decoder decides on that codeword that leads to the minimum distance. This decoder is optimal in the sense that it minimizes the probability of a codeword error for the *binary symmetric channel.*

Figure 4.4 shows a binary symmetric channel. In a binary symmetric channel, both the transmitted and the received bit can only have the values 1 or 0. The values on the transitions denote the probability $p(y|x)$ of the received bit y given the transmitted bit x. The value ϵ can be interpreted as the probability of a bit error. For example if a 1 was transmitted, the probability of receiving a 0 is ϵ, the probability of receiving a 1 is $1 - \epsilon$.

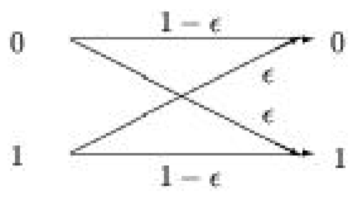

Figure 4.4: *Binary symmetric channel*

The probability $p(\boldsymbol{y}|\boldsymbol{c})$ can be written as

$$p(\boldsymbol{y}|\boldsymbol{c}) = (1-\epsilon)^{N-i}\,\epsilon^{i}, \tag{4.8}$$

where i is the number of bits with $y \neq x$ and N is the block-length of the code, i.e. the vector $\boldsymbol{y}$ has N components. If we assume $\epsilon < \frac{1}{2}$, (4.8) is maximized by minimizing i which leads us to minimum distance decoding.

In the following an overview over decoders for both BCH codes and convolutional codes is given.

4.2.1 BCH Codes

As mentioned above, minimum distance decoding minimizes the probability of a codeword error for the binary symmetric channel. In order to implement a minimum distance decoder, one could compute the hamming distance of the received bit sequence to all possible codewords. However, there exist 2^k different codewords, so for a large codebook size (i.e. the number of different codewords in a particular code), this is not feasible anymore.

In the next part, we show a way of decoding that is easier to realize, the so called *syndrome decoding.* Then, we present a decoder that does syndrome decoding for BCH codes.

4.2.1.1 Syndrome Decoding

As stated before, in order to perform minimium distance decoding, one has to compute 2^k distances and choose the codeword that leads to the minimum distance. However, it is possible to implement a decoder managing with a look-up-table with 2^{N-k} entries.

Let $\boldsymbol{y}$ denote the received bit sequence. $\boldsymbol{y}$ can be written as the sum of the transmitted codeword and an error introduced by the channel:

$$\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}. \tag{4.9}$$

Note that all operations are performed in the galois field GF(q). By multiplying the received sequence $\boldsymbol{y}$ with the parity check matrix $\mathbf{H}$, the N − dimensional vector $\boldsymbol{y}$ is projected onto an $(N-k)$ − dimensional vector $\boldsymbol{s}$:

$$\begin{aligned} \boldsymbol{s} &= \boldsymbol{y}\mathbf{H}^T \\ &= (\boldsymbol{c}+\boldsymbol{e})\,\mathbf{H}^T \\ &= \boldsymbol{c}\mathbf{H}^T + \boldsymbol{e}\mathbf{H}^T \\ &= \boldsymbol{e}\mathbf{H}^T. \end{aligned} \tag{4.10}$$

The last equality follows by (3.2). The vector $\boldsymbol{s}$ is called a *syndrome*. Note that the syndrome $\boldsymbol{s}$ is only a function of the error $\boldsymbol{e}$ and does not depend on the codeword $\boldsymbol{c}$.

In order to build a decoder one can create a look-up-table containing all possible syndromes $\boldsymbol{s}$ and the corresponding error patterns $\boldsymbol{e}$. The decoding procedure goes as follows:

1. Compute the syndrome $\boldsymbol{s} = \boldsymbol{y}\mathbf{H}^T$.
2. Determine the estimated error $\hat{\boldsymbol{e}}$ using the look-up-table.
3. Compute the estimated codeword $\hat{\boldsymbol{c}} = \boldsymbol{y} + \hat{\boldsymbol{e}}$.

As an example, we again consider the binary $(7, 4)$ hamming code with generator matrix $\mathbf{G}$ and parity check matrix $\mathbf{H}$ described in Subsection 3.1.1. Again, the 4 information bits are $\boldsymbol{x} = (1101)$. Assuming we receive $\boldsymbol{y} = (1111001)$, the syndrome $\boldsymbol{s}$ can be computed according to (4.10) which yields $\boldsymbol{s} = (110)$. Making use of the look-up-table, we obtain for the estimated error $\hat{\boldsymbol{e}} = (0010000)$. From the addition $\boldsymbol{y} + \hat{\boldsymbol{e}}$ follows $\hat{\boldsymbol{c}} = (1101001)$ which is equal the codeword $\boldsymbol{c}$ in Subsection 3.1.1.

Because a syndrome is an $(N-k)$ − dimensional vector, there exist 2^{N-k} different syndromes. Consequently, the decoder is able to manage with a look-up-table with 2^{N-k} entries as mentioned before.

4.2.1.2 Decoders for BCH Codes

Syndrome decoding is able to deal with a look-up-table with $(N-k)$ entries. However, one needs a look-up-table for any particular code. Such a decoder is not very flexible. For BCH codes, there exist syndrome decoders that deal without a look-up-table. In the following we discuss the *Peterson-Gorenstein-Zierler decoder* that is not optimal in respect of complexity but gives us a good understanding of how a decoding procedure for BCH codes could be implemented. In addition, decoders with less complexity can be deduced from this decoder. Note that BCH codes are cyclic and we use, for convenience, the polynomial description.

Assume, that $y(p)$ is the received bit sequence. According to (4.9), $y(p)$ can be written as

$$y(p) = c(p) + e(p), \tag{4.11}$$

where $e(p) = e_{N-1}p^{N-1} + e_{N-2}p^{N-2} + \ldots + e_1 p + e_0$ is the error polynomial.

From the definition of the BCH codes follows that the generator polynomial $g(p)$ has the zeros $\beta, \beta^2, \ldots, \beta^{2t}$. It can be shown that

$$c(\beta) = c(\beta^2) = \ldots = c(\beta^{2t}) = 0 \tag{4.12}$$

if and only if $c(p)$ is a codeword polynomial [3]. Therefore, the syndrome S_j can be computed as follows:

$$S_j = y(\beta^j) = c(\beta^j) + e(\beta^j) = e(\beta^j), \text{ for } j = 1, \ldots, 2t. \tag{4.13}$$

If we assume that ν errors occur at error locations $i_1, i_2, \ldots, i_\nu$, the error polynomial can be written as

$$e(p) = e_{i_1}p^{i_1} + e_{i_2}p^{i_2} + \ldots + e_{i_{\nu-1}}p^{i_{\nu-1}} + e_{i_\nu}p^{i_\nu}. \tag{4.14}$$

With (4.13), (4.14) and the substitutions $X_l = \beta^{i_l}$ and $Y_l = e_{i_l}$[1], we get the following system of nonlinear equations:

$$\begin{array}{ccccccccc} S_1 & = & Y_1X_1 & + & Y_2X_2 & + & \ldots & + & Y_\nu X_\nu \\ S_2 & = & Y_1X_1^2 & + & Y_2X_2^2 & + & \ldots & + & Y_\nu X_\nu^2 \\ \vdots & & \vdots & & \vdots & & & & \vdots \\ S_{2t} & = & Y_1X_1^{2t} & + & Y_2X_2^{2t} & + & \ldots & + & Y_\nu X_\nu^{2t}. \end{array} \tag{4.15}$$

Note that X_l is unknown since one does not know the error location i_l.

In order to solve (4.15), we use the *error-locator polynomial* $\Lambda(p)$ that is zero for $p = X_1^{-1}, X_2^{-1}, \ldots, X_\nu^{-1}$:

$$\begin{aligned} \Lambda(p) &= \Lambda_\nu p^\nu + \Lambda_{\nu-1}p^{\nu-1} + \ldots + \Lambda_1 p + 1 \\ &= (1 - pX_1)(1 - pX_2)\cdots(1 - pX_\nu) \end{aligned} \tag{4.16}$$

[1] Y_l is called the error magnitude.

Using (4.15) and (4.16), one obtains a system of linear equations:

$$\underbrace{\begin{pmatrix} S_1 & S_2 & \cdots & S_{\nu-1} & S_\nu \\ S_2 & S_3 & \cdots & S_\nu & S_{\nu+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-2} & S_{2\nu-1} \end{pmatrix}}_{\mathbf{M}} \begin{pmatrix} \Lambda_\nu \\ \Lambda_{\nu-1} \\ \vdots \\ \Lambda_1 \end{pmatrix} = \begin{pmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ \vdots \\ -S_{2\nu}. \end{pmatrix} \tag{4.17}$$

It can be shown that for a proper choice of the number of errors ν, the matrix $\mathbf{M}$ is nonsingular and, therefore, this system of equations can be solved.

The procedure of the Peterson-Gorenstein-Zierler decoder, shown in Figure 4.5, goes as follows. First the syndromes $S_1, \ldots, S_{2t}$ are computed. Then the error-locator polynomial $\Lambda(p)$ is determined solving (4.17). By finding the zeros of $\Lambda(p)$ one obtains the error locations $X_1, \ldots, X_\nu$. Solving (4.15) yields the error magnitudes $Y_1, \ldots, Y_\nu$ that are needed to decode the codeword.

As mentioned above this decoder is not optimal in regard of complexity. The reason is that in this decoding procedure two matrix inversions need to be performed. Since matrix inversions have a high complexity, better decoders can be deduced from the Peterson-Gorenstein-Zierler decoder by finding a way to get by without those inversions.

One approach is to obtain the error-locator polynomial $\Lambda(p)$ using the *Berlekamp-Massey algorithm*, shown in Figure 4.6. The Berlekamp-Massey algorithm interprets (4.17) as the equation of the output of a shift register circuit and tries to find the shift register with smallest length. (For a more detailed explanation see [3].)

Another appproach uses the *Forney algorithm* in order to find the error magnitudes. However, if the BCH code is binary the error magnitudes are always 1 and it is sufficient to know the error locations $i_1, \ldots, i_\nu$. In that case, the decoding procedure is finished after finding the zeros of $\Lambda(p)$.

Figure 4.7 shows a decoding procedure for BCH codes that gets by without any matrix inversions. It is based on the Peterson-Gorenstein-Zierler decoder but uses the Berlekamp-Massey algorithm and the Forney algorithm in order to avoid matrix inversions.

With the decoder shown in Figure 4.7 an efficient decoder for BCH codes is presented. In Section 5.2 we investigate the complexity of this decoders.

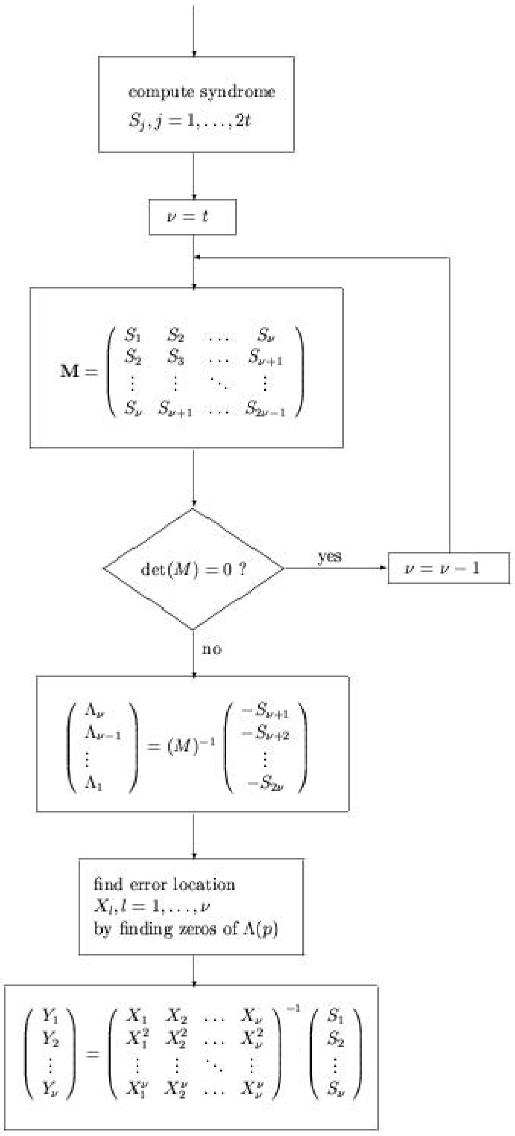
compute syndrome
$S_j, j = 1, \ldots, 2t$
$\nu = t$
$\mathbf{M} = \begin{pmatrix} S_1 & S_2 & \ldots & S_\nu \\ S_2 & S_3 & \ldots & S_{\nu+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_\nu & S_{\nu+1} & \ldots & S_{2\nu-1} \end{pmatrix}$
$\det(M) = 0$?
yes
$\nu = \nu - 1$
no
$\begin{pmatrix} \Lambda_\nu \\ \Lambda_{\nu-1} \\ \vdots \\ \Lambda_1 \end{pmatrix} = (M)^{-1} \begin{pmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ \vdots \\ -S_{2\nu} \end{pmatrix}$
find error location
$X_l, l = 1, \ldots, \nu$
by finding zeros of $\Lambda(p)$
$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_\nu \end{pmatrix} = \begin{pmatrix} X_1 & X_2 & \ldots & X_\nu \\ X_1^2 & X_2^2 & \ldots & X_\nu^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^\nu & X_2^\nu & \ldots & X_\nu^\nu \end{pmatrix}^{-1} \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_\nu \end{pmatrix}$

Figure 4.5: *Peterson-Gorenstein-Zierler decoder*

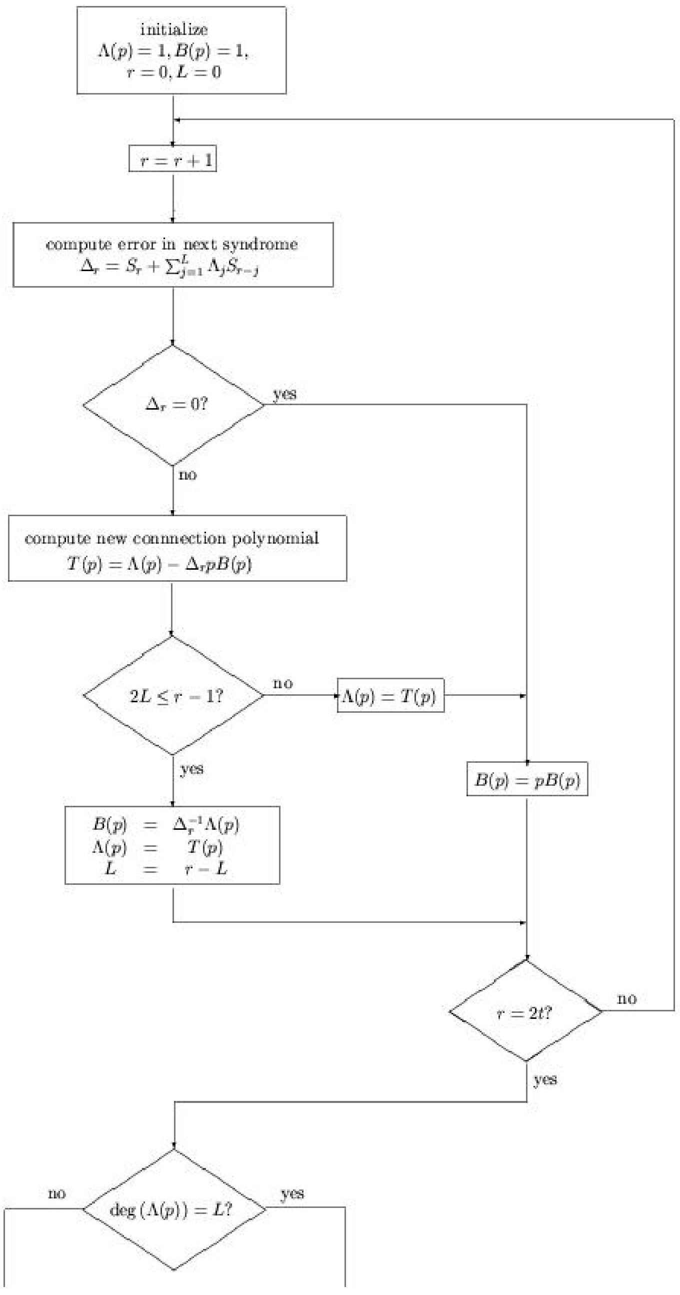
initialize
$\Lambda(p) = 1, B(p) = 1,$
$r = 0, L = 0$
$r = r + 1$
compute error in next syndrome
$\Delta_r = S_r + \sum_{j=1}^{L} \Lambda_j S_{r-j}$
$\Delta_r = 0?$
yes
no
compute new connnection polynomial
$T(p) = \Lambda(p) - \Delta_r p B(p)$
$2L \leq r - 1?$
no
$\Lambda(p) = T(p)$
yes
$B(p) = pB(p)$
$B(p) = \Delta_r^{-1}\Lambda(p)$
$\Lambda(p) = T(p)$
$L = r - L$
$r = 2t?$
no
yes
no
$\deg(\Lambda(p)) = L?$
yes

Figure 4.6: *Berlekamp-Massey algorithm*

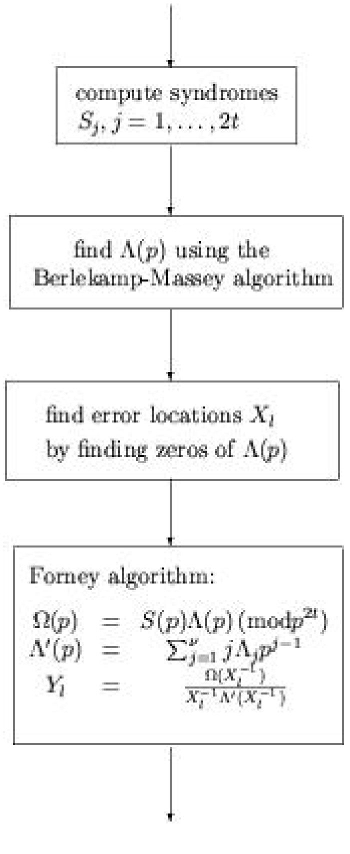
compute syndromes
$S_j, j = 1, \ldots, 2t$
find $\Lambda(p)$ using the
Berlekamp-Massey algorithm
find error locations X_l
by finding zeros of $\Lambda(p)$
Forney algorithm:
$\Omega(p) = S(p)\Lambda(p)\,(\mathrm{mod} p^{2t})$
$\Lambda'(p) = \sum_{j=1}^{\nu} j\Lambda_j p^{j-1}$
$Y_l = \frac{\Omega(X_l^{-1})}{X_l^{-1}\Lambda'(X_l^{-1})}$

Figure 4.7: *Fast decoding of BCH codes*

4.2.2 Convolutional Codes

For decoding data encoded with convolutional codes we use the Viterbi algorithm on a trellis. In Figure 3.4 a trellis is given as an example. For every edge e in a trellis we write $lst(e)$ for the left state, $rst(e)$ for the right state of e and $\mu(s)$ stands for the metrics at the the state s. The Viterbi algorithm is defined as follows:

Initialisation: $\mu(s) = 0$ **for every starting state** s

Calculation of metrics in every state:

$$\mu(s) = \min_{Edges\, e\, with\, rst(e)=s} \mu(lst(e)) + \mu(e) \tag{4.18}$$

A detailed discussion of the Viterbi algorithm can be found in [12] for example.

Chapter 5

Complexity

In the last chapter a new equalization algorithm according to [8] (i.e. the MAP-DFE) and decoders for both BCH and convolutional codes are discussed. Two main criteria for a good equalizer respectively decoder are the performance in terms of error probability and the decoding complexity. In this chapter we focus on the complexity. We derive the decoding complexity in terms of multiplication and additions. In order to compare multiplications and additions we refer to [13] where it is stated that multiplication needs a chip-area 26-times larger than addition at same speed. Therefore, we assume that one addition equals 26 multiplications.

In the following, first the complexity of the MAP-DFE is derived. Then we discuss the complexity of BCH codes followed by the discussion of the complexity of convolutional codes.

5.1 MAP-DFE

In Subsection 4.1.2 the MAP-DFE is presented. The equalization procedure can be outlined as follows:

1. MMSE estimation of all still not decoded symbols.
2. Determining the a posteriori error probabilities of all still not decoded symbols.
3. Determining the segment of the symbols that will be jointly decoded.
4. Deciding the symbols of the segment of step 3.

5. Compensation of the interference of all symbols of step 4 on the remaining not decoded signals.

6. Proceed to step 1 unless all symbols are decoded.

According to [15] a multiplication of two $(N \times N)$-matrices can be done at the cost of

$$\frac{N^3}{2} + N^2 \text{ multiplications}$$

and

$$\frac{3N^3}{2} + 2N(N-1) \text{ additions}$$

using the algorithm of Winograd [15].

The inverse of an $(N \times N)$-matrix can be computed by Gaussian elimination. In general, Gaussian elimination can be used in order to solve the equation $\mathbf{AX} = \mathbf{B}$. The key idea is to divide $\mathbf{A}$ into a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$ and to split up the problem into two equations with $\mathbf{L}$ and $\mathbf{U}$. Here, this procedure is not discussed in more detail. For further information we refer to [6]. In order to compute the inverse of an $(N \times N)$-matrix $\mathbf{A}$, one can solve the equation $\mathbf{AX} = \mathbf{B}$ for $\mathbf{B} = \mathbf{I}_N$ which leads to $\mathbf{X} = \mathbf{A}^{-1}$. This procedure requires

$$\frac{N}{3}\left(2N^2 + 3N - 2\right)$$

multiplications and the same amount of additions [15]. The multiplication of an K-dimensional vector with an $(N \times K)$-matrix needs NK multiplications and $(NK - N)$ additions. In the following we derive the decoding complexity of the MAP-DFE in terms of multiplications and additions. Note that all additions and multiplications in this section are performed on complex numbers.

In a first step, the MAP-DFE performs an MMSE estimation on the received symbols. The MMSE matrix $\mathbf{G}_{MMSE}$ according to (4.5) is

$$\left(\Lambda_{ISI}\Lambda_{ISI}^H + \Lambda_{\boldsymbol{nn}}\right)^{-1}\Lambda_{ISI}.$$

Λ_{ISI} and Λ_{ISI} are $(N_c \times N_c)$-matrices, where N_c is the dimension of the received vector $\boldsymbol{r}$. The computation of the MMSE matrix requires, therefore,

$$\begin{aligned} 2\left(\frac{N_c^3}{2} + N_c^2\right) + \frac{N_c}{3}\left(2N_c^2 + 3N_c - 2\right) &= \\ \frac{5}{3}N_c^3 + 3N_c^2 - \frac{2}{3}N_c & \end{aligned} \tag{5.1}$$

multiplications and

$$\begin{aligned} 2\left(\frac{3}{2}N_c^3 + 2N_c(N_c-1)\right) + \frac{N_c}{3}\left(2N_c^2+3N_c-2\right) + N_c^2 &= \\ \frac{11}{3}N_c^3 + 5N_c^2 - \frac{14}{3}N_c & \end{aligned} \tag{5.2}$$

additions, where N_c^2 in the first line in (5.2) is the number of required additions to compute the sum $\Lambda_{ISI}\Lambda_{ISI}^H + \Lambda_{\boldsymbol{nn}}$. In order to perform an MMSE estimation, one has to multiply the received vector $\boldsymbol{r}$ with the $(N_c \times N_c)$-MMSE matrix $\mathbf{G}_{MMSE}$. This requires N_c^2 multiplications and $(N_c^2 - N_c)$ additions.

Note that it is possible to reduce the dimension of the MMSE matrix $\mathbf{G}_{MMSE}$ with every decoded symbol. Hence, the computed decoding complexity of the MMSE estimation is only exact if just one iteration is required (i.e. $P_{e,threshold} = 0$). Otherwise the computed complexity is an upper bound.

In a next step, the a posteriori error probabilities need to be computed. For a received 4-QAM symbol, this could be done using (4.6) and (4.7). However, (4.6) requires the evaluation of an exponential function, which is rather complex. Therefore, we assume that the error probabilities can be determined using a look-up-table and, consequently, the required operations are neglectable.

The required operations in step 3 and 4 are neglected, too. In step 5 the interference of the decoded symbols on the remaing symbols needs to be cancelled. In order to do so, first one has to compute $\tilde{\boldsymbol{r}} = \Lambda_{ISI}(:, \boldsymbol{k})\boldsymbol{\alpha}(\boldsymbol{k})$, where $\boldsymbol{k}$ is the index vector containing all indices of the decoded symbols. Then, the interference is compensated by subtracting $\tilde{\boldsymbol{r}}$ from $\boldsymbol{r}$. Assume that $|\boldsymbol{k}|$ denotes the number of elements in $\boldsymbol{k}$. Then, the multiplication of vector $\boldsymbol{\alpha}(\boldsymbol{k})$ with matrix $\Lambda_{ISI}(:, \boldsymbol{k})$ requires $N_c|\boldsymbol{k}|$ multiplications and $(N_c|\boldsymbol{k}| - N_c)$ additions. The substraction of $\tilde{\boldsymbol{r}}$ from $\boldsymbol{r}$ needs $|\boldsymbol{k}|$ more additions.

The decoding complexity of one iteration of the MAP-DFE then results in:

$$\frac{5}{3}N_c^3 + 4N_c^2 + \left(|\boldsymbol{k}| - \frac{2}{3}\right)N_c \text{ multiplications} \tag{5.3}$$

and

$$\frac{11}{3}N_c^3 + 6N_c^2 + \left(|\boldsymbol{k}| - \frac{20}{3}\right)N_c \text{ additions.} \tag{5.4}$$

Assume that ϑ denotes the number of required iterations and $|\bar{\boldsymbol{k}}|$ denotes the average number of elements in $\boldsymbol{k}$. ϑ can be expressed as:

$$\vartheta = \frac{N_c}{|\bar{\boldsymbol{k}}|}. \tag{5.5}$$

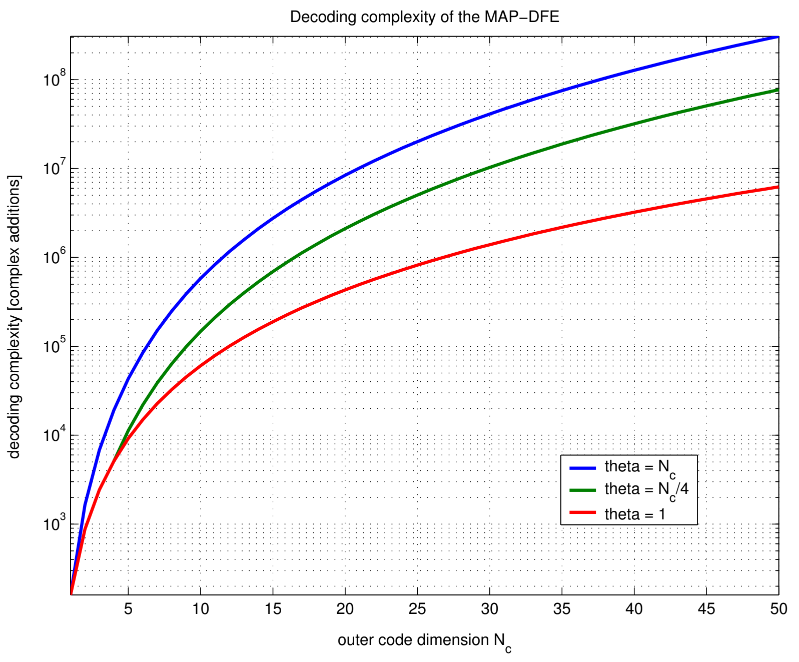

Figure 5.1: *Decoding complexity of the MAP-DFE in terms of additions*

Using (5.5) leads to the average decoding complexity of the MAP-DFE:

$$\vartheta\left(\frac{5}{3}N_c^3+4N_c^2+\left(|\bar{\boldsymbol{k}}|-\frac{2}{3}\right)N_c\right)\quad\text{multiplications}\tag{5.6}$$

and

$$\vartheta\left(\frac{11}{3}N_c^3+6N_c^2+\left(|\bar{\boldsymbol{k}}|-\frac{20}{3}\right)N_c\right)\quad\text{additions.}\tag{5.7}$$

As stated before, one addition equals 26 multiplications. Consequently, the decoding complexity in terms of additions becomes

$$\vartheta\left(47N_c^3+110N_c^2+\left(27|\bar{\boldsymbol{k}}|-24\right)N_c\right)\tag{5.8}$$

(complex) additions.

Figure 5.1 shows the decoding complexity of the MAP-DFE in terms of (complex) additions. The lowermost curve corresponds to the MMSE estimation where only one iteration is required. This case leads to the minimum complexity. The uppermost curve shows the decoding complexity if $\vartheta = N_c$. The curve in between corresponds to $\vartheta = N_c/4$. We will show in Chapter 6 that for $P_{e,threshold} \geq 10^{-8}$ and $N_c = 8$, the MAP-DFE requires less than 2 iterations on average in the high SNR regime. Hence, the consideration of the decoding complexity corresponding to $\vartheta = N_c/4$ is reasonable.

It should be noted that only the lowermost curve shows the exact decoding complexity. The other curves are upper bounds.

5.2 BCH Codes

In Subsection 4.2.1 the decoder for BCH codes is discussed. Here we want to derive the decoding complexity of those codes. The decoding complexity is expressed in terms of additions and multiplications. Note that all additions and multiplications in this section are performed on real numbers.

In order to decode data encoded with BCH codes, we use the decoder shown in Figure 4.7. The decoding procedure can be outlined as follows:

1. Compute the syndromes $S_j = y(\beta^j)$.
2. Determine the error-locator polynomial $\Lambda(p)$ using the Berlekamp-Massey algorithm.
3. Find the zeros $p_1, p_2, \ldots, p_\nu$ of $\Lambda(p)$.
4. Identify the error locations $i_1, i_2, \ldots, i_\nu$ by using the fact that $p_l = X_l^{-1} = \beta^{-i_l}$.
5. Determine $\hat{c(p)}$ by flipping the bits at the positions $i_1, i_2, \ldots, i_\nu$.

Note that we assume the received sequence $y(p)$ to be binary. Therefore, all error magnitudes $Y_1, Y_2, \ldots, Y_\nu$ are one and, consequently, it is sufficient to know the error locations. (The decoder shown in Figure 4.7 uses the Forney algorithm in order to compute the error magnitudes; in the binary case we can omit this step.)

In the following we want to analyze every step in this decoding procedure. For those who are not interested in the whole derivations we refer to Subsection 5.2.4.

5.2.1 The Syndromes

In order to compute the syndromes, one has to evaluate the polynomial of the received sequence $y(p)$ at positions $\beta^1, \beta^2, \ldots, \beta^{2t}$. According to Horner's scheme [2] every computation $y(\beta^j)$ can be done in κ multiplications and the same amount of additions, where κ is the degree of the polynomial $y(p)$. Therefore, in order to compute $y(\beta^j)$ for a code of length N, there are $(N-1)$ multiplications and additions required.

Considering the fact that $y(p)$ needs to be evaluated at $2t$ different positions, the computation of the syndromes requires

$$2t(N-1) \tag{5.9}$$

multiplications and additions, where t is the number of correctable errors.

5.2.2 The Berlekamp-Massey Algorithm

The next step of the decoding procecure is the computation of the error-locator polynomial using the Berlekamp-Massey algorithm. The complexity of this algorithm in terms of additions and multiplications is derived in [7]. In the following we present the results:

Let t be the number of correctable errors and assume the syndromes $S_1, S_2, \ldots, S_\nu$ to take values in $\{0, 1, \ldots, p-1\}$. Moreover, define $\tilde{n} = 2t$. Then the Berlekamp-Massey algorithm needs on average

$$\frac{1}{2}\tilde{n}(\tilde{n}+1) - \frac{1}{4}\tilde{n}p^{-1} \tag{5.10}$$

multiplications and the same amount of additions.

In the case of BCH codes of length N, the syndromes take values in $\{0, 1, \ldots, N\}$, so $p = N + 1$. In this case, (5.10) becomes

$$\begin{aligned} \frac{1}{2}\tilde{n}(\tilde{n}+1) - \frac{1}{4}\tilde{n}^2\frac{1}{N+1} &= \frac{1}{2}\tilde{n}^2 + \frac{1}{2}\tilde{n} - \frac{1}{4}\tilde{n}^2\frac{1}{N+1} \\ &= \frac{1}{2}\tilde{n}^2\left(1 - \frac{1}{2(N+1)}\right) + \frac{1}{2}\tilde{n}. \end{aligned} \tag{5.11}$$

Using $\tilde{n} = 2t$ the average decoding complexity becomes

$$\frac{1}{2}(2t)^2\left(1 - \frac{1}{2(N+1)}\right) + \frac{1}{2}(2t) = 2t^2\left(1 - \frac{1}{2(N+1)}\right) + t \tag{5.12}$$

multiplications and additions.

5.2.3 The Zeros of the Error-Locator Polynomial

By finding the zeros of the error-locator polynomial $\Lambda(p)$ one is able to identify the error locations $i_1, i_2, \ldots, i_\nu$. We determine the zeros by computing

$$\Lambda(\beta^j), j = 1, \ldots, 2t \tag{5.13}$$

and checking if (5.13) is zero. This procedure is called *Chien search* [3]. The Chien search can be performed using Horner's scheme. The error-locator polynomial $\Lambda(p)$ has degree ν, where ν is number of errors occured. If we assume that all errors can be corrected, then $\nu \leq t$. Therefore, the evaluation of $\Lambda(\beta^j)$ requires less than t multiplications and additions. Hence, the zeros of $\Lambda(p)$ can be found at the cost of

$$t(2t) = 2t^2 \tag{5.14}$$

multiplications and the same amount of additions.

Note that the error-locator polynomial $\Lambda(p)$ has the zeros

$$\begin{array}{rclcl} p_1 & = & X_1^{-1} & = & \beta^{-i_1} \\ p_2 & = & X_2^{-1} & = & \beta^{-i_2} \\ & \vdots & & \vdots & \\ p_\nu & = & X_\nu^{-1} & = & \beta^{-i_\nu}. \end{array} \tag{5.15}$$

By knowing $p_1, \ldots, p_\nu$, the error locations $i_1, \ldots, i_\nu$ can easily be identified using a look-up-table. Therefore, the required operations to evaluate the error locations can be neglected.

5.2.4 Decoding Complexity of BCH Codes

As stated above the decoding procedure of BCH codes goes as follows:

1. Compute the syndromes $S_j = y(\beta^j)$.
2. Determine the error-locator polynomial $\Lambda(p)$ using the Berlekamp-Massey algorithm.
3. Find the zeros $p_1, p_2, \ldots, p_\nu$ of $\Lambda(p)$.
4. Identify the error locations $i_1, i_2, \ldots, i_\nu$ by using the fact that $p_l = X_l^{-1} = \beta^{-i_l}$.
5. Determine $\hat{c}(p)$ by flipping the bits at the positions $i_1, i_2, \ldots, i_\nu$.

We have shown that the computation of the syndromes needs

$$2t(N-1)$$

multiplications and additions. Additionally, the Berlekamp-Massey algorithm needs

$$2t^2\left(1 - \frac{1}{2(N+1)}\right) + t$$

operations. The zeros of the error-locator polynomial $\Lambda(p)$ can be determined at the cost of

$$2t^2$$

operations. The flipping of the bits can be done with ν bit additions, whereas $\nu \leq t$. Consequently, the decoding complexity of BCH codes follows as:

$$\begin{aligned} 2t(N-1) + 2t^2 + t^2\left(1 - \frac{1}{2(N+1)}\right) + t + t &= \\ 2\left(3 - \frac{1}{2(N+1)}\right)t^2 + 2tN & \end{aligned} \tag{5.16}$$

multiplications and the same amount of additions.

As stated before, we assume that one addition equals 26 multiplications. Therefore, the decoding complexity of BCH codes in terms of additions becomes

$$54\left(3-\frac{1}{2(N+1)}\right)t^2+54tN \tag{5.17}$$

real additions.

5.3 Convolutional Codes

In the following section we want to express the decoding complexity for convolutional codes in terms of the number of arithmetic operations.

For decoding data encoded with convolutional codes we use the Viterbi algorithm on a trellis as described in Section 4.2.2. For a (n,k) convolutional code with constraint length K the number of possible states is $2^{k(K-1)}$. The number of incoming edges at each state is 2^k as soon as the steady state has been reached. (After the steady state the structure of the trellis keeps repeating as can be seen in Figure 3.4).

The number of nodes in the trellis follows as

$$N2^{k(K-1)}-2(K-1)2^{k(K-1)}+2\sum_{i=0}^{K-2}2^{ik}=(N-2(K-1))2^{k(K-1)}+\sum_{i=0}^{K-2}2^{ik+1} \tag{5.18}$$

We use k=1 why (5.18) simplifies to

$$(N-2(K-1))2^{(K-1)}+\sum_{i=1}^{K-1}2^{i} \tag{5.19}$$

Before decoding using the Viterbi algorithm we need to label the edges. Since we use hard decision decoding this means calculating a Hamming distance at each edge. After the steady state this corresponds to 2^k additions per node, for the nodes before the steady state (i.e. the nodes with $depth \leq K-1$) there is only one incoming edge per node, so there is one addition. (Since with each k-bit-sequence that is shifted into the register a new state is reached). Expressed in a formula

$$
\begin{aligned}
(N-2(K-1))2^{k(K-1)}2^k + \sum_{i=0}^{K-2} 2^{ik}2^k + \sum_{i=0}^{K-2} 2^{ik} &= \\
(N-2(K-1))2^{k((K-1)+1)} + \sum_{i=0}^{K-2} 2^{k(i+1)} + \sum_{i=0}^{K-2} 2^{ik} &= \\
(N-2(K-1))2^{kK} + \sum_{i=0}^{K-2} 2^{k(i+1)} + \sum_{i=0}^{K-2} 2^{ik} &= \\
(N-2(K-1))2^{kK} + \sum_{i=1}^{K-1} 2^{ik} + \sum_{i=0}^{k-2} 2^{ik} &= \\
(N-2(K-1)+1)2^{kK} + 1 + \sum_{i=1}^{K-2} 2^{ik} + 2^{k(K-1)} & \qquad (5.20)
\end{aligned}
$$

The second step in decoding consists of choosing the best incoming edge for each node, which means calculating 2^k sums and comparing the resulting 2^k values:

$$2^k + 2^k - 1 = 2 \cdot 2^k - 1 = 2^{k+1} - 1 \qquad (5.21)$$

This means $2^{k+1} - 1$ additions per node. Again, this is valid for the nodes after the steady state - before, there is only one incoming edge per node which means calculating only one addition and no comparison.

$$
\begin{aligned}
(N-2(K-1))2^{k(K-1)}(2^{k+1}-1) + \sum_{i=0}^{K-2} 2^{ik}(2^{k+1}-1) + \sum_{i=0}^{K-2} 2^{ik} &= \\
(N-2(K-1))(2^{kK+1} - 2^{kK-k}) + \sum_{i=0}^{K-2} (2^{ik+k+1} - 2^{ik}) + \sum_{i=0}^{K-2} 2^{ik} &= \\
(N-2(K-1))2^{kK}(2-2^{-k}) + \sum_{i=0}^{K-2} 2^{k(i+1)+1} &= \\
(N-2(K-1))2^{kK}(2-2^{-k}) + \sum_{i=1}^{K-1} 2^{ik+1} & \qquad (5.22)
\end{aligned}
$$

By adding (5.20) and (5.22) the complexity (expressed in the number of additions) for decoding with the Viterbi algorithm on a trellis follows as

$$2^{kK}[(N-2(K-1)+1)+(N-2(K-1))(2-2^{-k})]\sum_{i=1}^{K-1}2^{ik+1}+\sum_{i=1}^{K-2}2^{ik}+2^k(K-1)1 \quad =$$
$$2^{kK}[(N-2(K-1))(3-2^{-k})+1]+\sum_{i=1}^{K-2}(2^{ik+1}+2^{ik})+2^{k(K-1)}+1 \quad (5.23)$$

For k=1 (5.23) can be simplified again

$$2^K((N-2(K-1))2.5+1)+\sum_{i=1}^{K-2}(2^i2^{i+1})+2^{K-1}+1 \qquad (5.24)$$

Chapter 6

Simulation and Results

The goal of this project was the investigation of a communication system using forward error correction, space-time coding and spatial multiplexing in order to improve link reliability and spectral efficiency. The components like FEC, LSD codes or MAP-DFE have a lot of parameters that need to be defined. To be specific, some parameters are FEC code-rate, block-length, constraint length of the convolutional code, number of subchannels and achieved diversity order, number of transmit and receive antennas, the treshold $P_{e,threshold}$ of the MAP-DFE and some more. Consequently, there is a large range of optimization.

In the following we discuss some of the tradeoffs choosing those parameters. In Section 6.1 the MAP-DFE is investigated. Section 6.2 analyzes the tradeoff between decoding complexity and performance of BCH and convolutional codes. In Section 6.3 the influence of LSD codes and FEC on systems using pure transmit diversity (i.e. only one antenna at the receiver) are studied. In Section 6.4 the tradeoff between the choice of the symbol-alphabet and the number of used subchannels is discussed. And finally, Section 6.5 investigates the tradeoff between the code-rate and spatial multiplexing.

6.1 The MAP-DFE

As we can see in Section 5.1, the decoding complexity of the MAP-DFE depends linearly on the number of required iterations ϑ. In a first step, we want to determine the number of average iterations in function of the threshold $P_{e,threshold}$ and of $\frac{E_b}{N_0}$ and investigate the influence on the symbol error rate. For this purpose we consider a system with 4 transmit and receive antennas. The LSD code has an outer code dimension $N_c = 8$ (i.e. the input vector $\boldsymbol{r}$ of the MAP-DFE introduced in Section 4.1 contains 8 elements), therefore, 8 is the maximum number of iterations. Simulations with $N_u = 4$ subchannels and 4-QAM symbols as well as simulations with $N_u = 2$ subchannels and 16-QAM symbols are performed. It can easily be verified that both systems achieve the same throughput.

In Figures 6.1 and 6.3 the average number of iterations required by the MAP-DFE is shown in function of $\frac{E_b}{N_0}$ and the threshold $P_{e,threshold}$. Figures 6.2 and 6.4 show the influence of the threshold on the symbol error rate for different $\frac{E_b}{N_0}$-values. The uppermost curve corresponds to the smallest $\frac{E_b}{N_0}$-value, namely $\frac{E_b}{N_0} = -3\,\text{dB}$. The lowermost curve corresponds to the largest $\frac{E_b}{N_0}$, namely $\frac{E_b}{N_0} = 9\,\text{dB}$.

We can see that in the high SNR regime (i.e. $\frac{E_b}{N_0}$ is large) the MAP-DFE requires less than 2 iterations for a threshold $P_{e,threshold} \leq 10^{-8}$. The reason is that if $\frac{E_b}{N_0}$ is large the probability of a symbol error is small, and, therefore, many symbols can be decoded jointly. This is an advantage compared to the V-BLAST equalizer according to [5], since V-BLAST decodes only one symbol per iteration even if the SNR is large.

By comparing both considered systems with $N_u = 4$ and $N_u = 2$, we can see that the MAP-DFE needs in both cases many iterations in the low SNR regime (i.e. $\frac{E_b}{N_0}$ is small) and/or in the region where the threshold $P_{e,threshold}$ is small. With increasing $\frac{E_b}{N_0}$ and/or increasing threshold, the number of iterations required decreases. However, the system using 2 subchannels has a steeper slope. This can be explained by the fact that 16-QAM requires a larger $\frac{E_b}{N_0}$-value than 4-QAM in order to achieve an adequate performance. Therefore, in the low SNR regime the system with $N_u = 2$ subchannels requires more iterations. In the high SNR regime, the advantage of the higher diversity order can be exploited. (Note, that the LSD codes use the remaining antennas that are not used for spatial multiplexing for diversity. Therefore, the less subchannels are used, the higher the diversity order.)

This behavior can also be observed by comparing Figures 6.2 and 6.4. In general, it can be stated that the lower the symbol error rate, the more the performance depends on the treshold $P_{e,threshold}$. The reason is that if $\frac{E_b}{N_0}$ is small, then the probability of a symbol error is large and, consequently, the additional error introduced by choosing a higher threshold is small. However, if $\frac{E_b}{N_0}$ is large, then the probability of a symbol error is low and, therefore, the choice of $P_{e,threshold}$ gains in importance. As stated before, in the high SNR regime only a few iterations are required in order to decode the symbols. Therefore, in this case it is reasonable to choose a small threshold $P_{e,threshold}$. In the low SNR regime the influence of the threshold on the symbol error rate is small and hence, it is reasonable to choose a large threshold in order to reduce the decoding complexity.

In order to determine the threshold that is optimum in terms of symbol error rate and decoding complexity, one has to choose the threshold at the sharp bend in the curve.

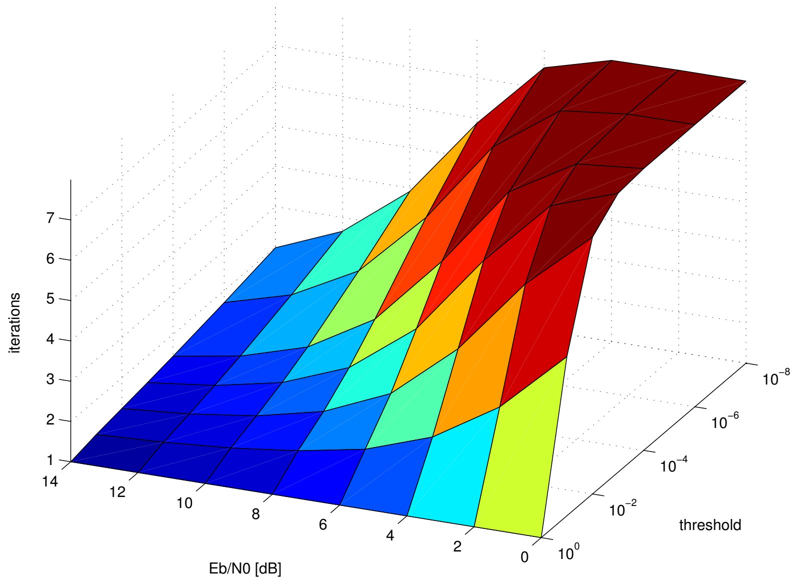

Figure 6.1: *Average number of iterations of the MAP-DFE.* $N_{TX} = 4, N_{RX} = 4, N_u = 4, 4 - \mathrm{QAM}$

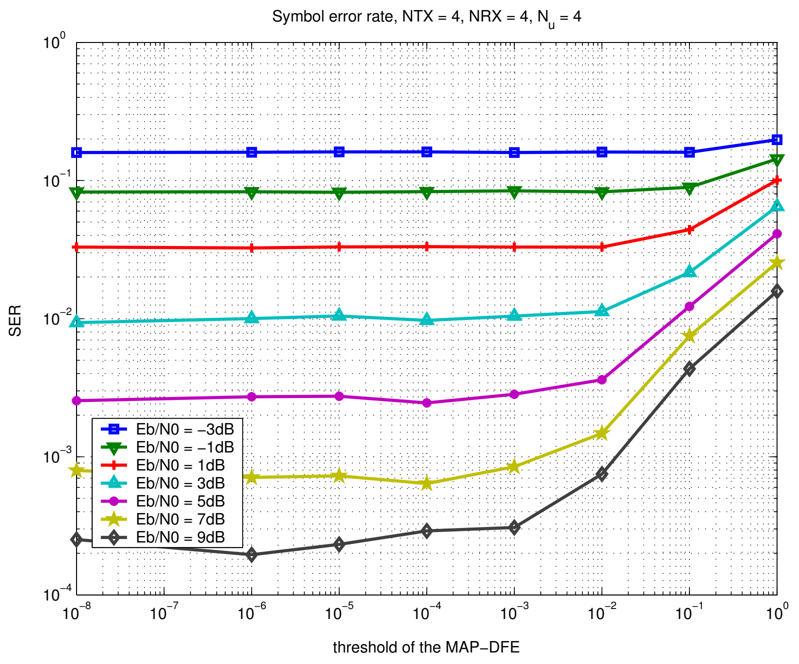

Figure 6.2: *Symbol error rate at the output of the MAP-DFE in function of the threshold* $P_{e,threshold}$. $N_{TX} = 4, N_{RX} = 4, N_u = 4, 4 - \mathrm{QAM}$

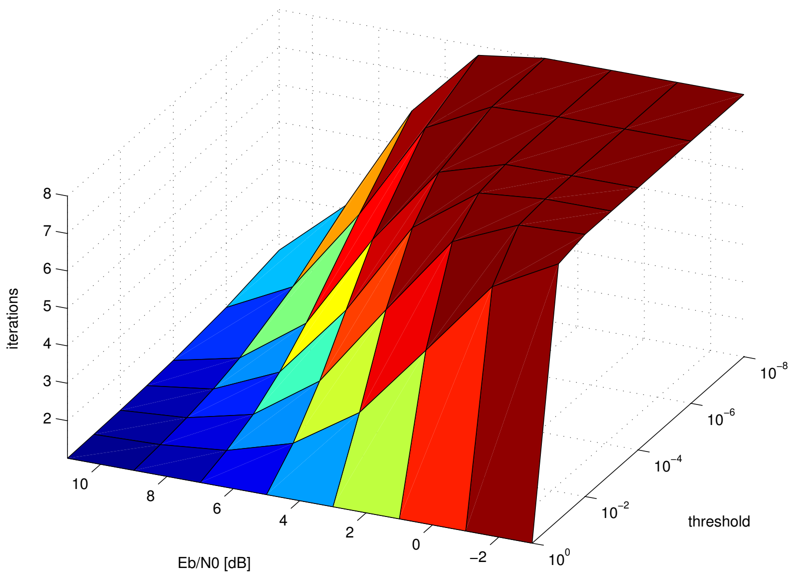

Figure 6.3: *Average number of iterations of the MAP-DFE.* $N_{TX} = 4, N_{RX} = 4, N_u = 2, 16 - \mathrm{QAM}$

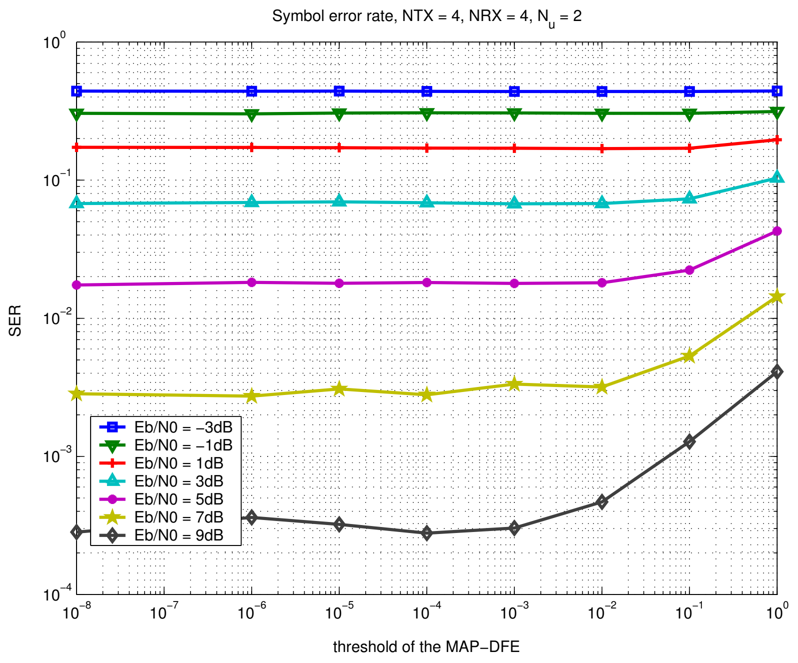

Figure 6.4: *Symbol error rate at the output of the MAP-DFE in function of the threshold* $P_{e,threshold}$. $N_{TX} = 4, N_{RX} = 4, N_u = 2, 16 - \mathrm{QAM}$

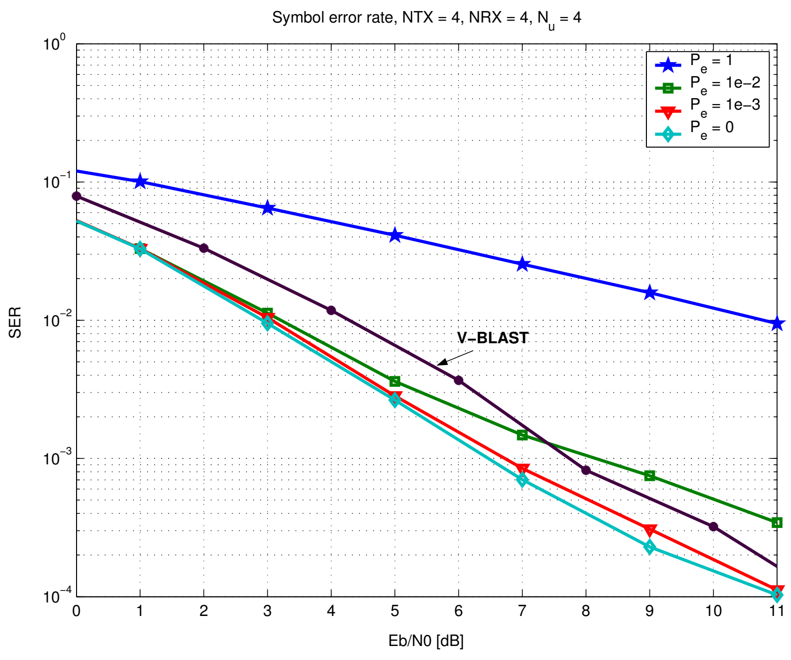

Figure 6.5: *Symbol error rate at the output of the MAP-DFE in function of* $\frac{E_b}{N_0}$. $N_{TX} = 4, N_{RX} = 4, N_u = 4, 4 - \mathrm{QAM}$

Obviously, the optimal threshold depends on the $\frac{E_b}{N_0}$-value. For $\frac{E_b}{N_0} = 9\,\mathrm{dB}$ the optimal threshold $P_{e,threshold}$ is 10^{-3}. If we choose a smaller threshold, there is no significant improvement in terms of symbol error rate but the MAP-DFE has a higher decoding complexity due to the lower threshold that results in a higher number of required iterations. On the other hand, if we choose the threshold larger, the symbol error rate increases significantly. Hence, $P_{e,threshold} = 10^{-3}$ is optimum for $\frac{E_b}{N_0} = 9\,\mathrm{dB}$. For $P_{e,threshold} = 10^{-3}$ and $\frac{E_b}{N_0} = 9\,\mathrm{dB}$, the MAP-DFE requires on average only 1.4 and $\vartheta = 1.1$ iterations for $N_u = 4$ respectively $N_u = 2$ compared to 8 iterations in the case where $P_{e,threshold} = 0$. This yields a complexity that is about 6-times smaller.

In a last step, we analyze the symbol error rate in function of $\frac{E_b}{N_0}$. Figures 6.5 and 6.6 show the symbol error rate for different values of the threshold $P_{e,threshold}$. For comparison, in Figure 6.5 also the symbol error rate of the V-BLAST using MMSE nulling is shown. The uppermost curve shows the symbol error rate of the MAP-DFE with $P_{e,threshold} = 1$ and corresponds to the MMSE estimation. The best performance in terms of error pobability is achieved for $P_{e,threshold} = 0$.

One can see that the performance attained with $P_{e,threshold} = 10^{-3}$ is nearly as good as the optimal performance. Furthermore, for $N_u = 4$ used subchannels the symbol error rate for threshold $P_{e,threshold} = 10^{-2}$ is near to the optimum until $\frac{E_b}{N_0} = 5\,\mathrm{dB}$. Then,

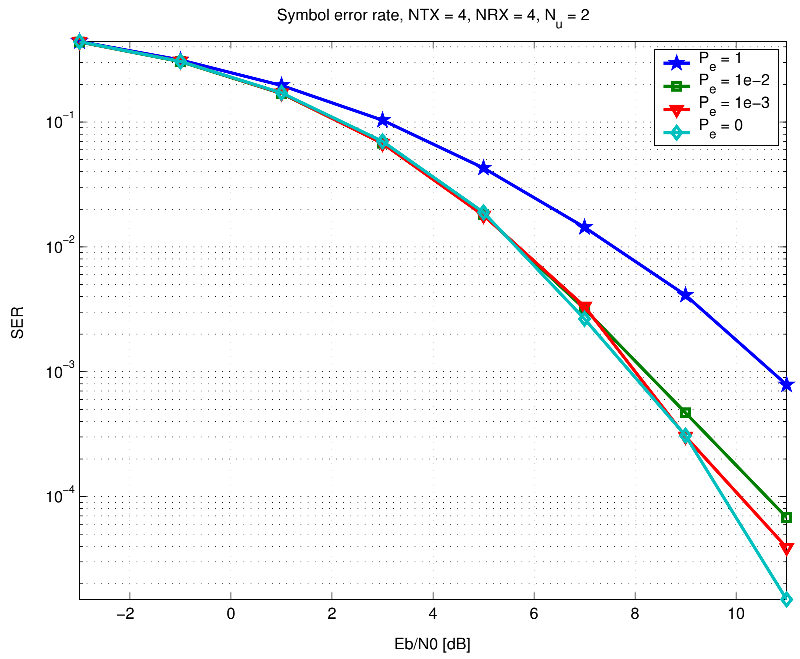

Figure 6.6: *Symbol error rate at the output of the MAP-DFE in function of* $\frac{E_b}{N_0}$. $N_{TX} = 4, N_{RX} = 4, N_u = 2, 16 - \text{QAM}$

the curve corresponding to $P_{e,threshold} = 10^{-2}$ deviates from the optimal curve. This is an indication of choosing the threshold too large. This can also be observed considering Figure 6.2. In addition, we can see in Figure 6.5 that for a proper choice of the threshold, the MAP-DFE outperforms the V-BLAST.

By comparing Figures 6.5 and 6.6 it can be observed that in the high SNR regime the system using $N_u = 2$ subchannels in combination with 16-QAM symbols outperforms the system, using $N_u = 4$ subchannels and 4-QAM symbols. This can be explained by the fact that in the high SNR regime the achieved diversity gain has a wider influence on the performance than the large symbol-alphabet. It should be noted that symbol error rates of 4-QAM and 16-QAM symbols are compared, which in general can lead to a false interpretation. However, in our simulations Gray-coding is used in order to map the bits to the symbols and, consequently, in the high SNR regime 1 symbol error implies only 1 bit error. Hence, in the high SNR regime the symbol error rate curves for 4-QAM and 16-QAM are comparable.

From the results shown above, it follows that in the region we are interested in (i.e. symbol error rate between 10^{-3} and 10^{-4}), $P_{e,threshold} = 10^{-3}$ is optimum in terms of error probability and decoding complexity. In the following, unless otherwise noted, all simulations are performed with $P_{e,threshold} = 10^{-3}$.

6.2 FEC Performance and Complexity Comparison

As described in Chapter 5 BCH and convolutional codes were chosen to act as examples for possible FEC codes. Before we present our results, we explain how we set our parameters.

For the convolutional codes we chose two different configurations, one with 16 states the other with 64. To be precise, a $(n = 2, k = 1, K = 5)$ code with octal generator polynomial (23,33) was implemented. This code was adopted by the GSM standardization committee in 1982 [10]. A more complex and powerful convolutional code with $(n = 2, k = 1, K = 7)$ and octal generator polynomial (171,133) was implemented as comparison. This code was employed by the DVB [10] standard for television, sound, and data services.

The BCH code (unlike the convolutional code) is a block code, thus our first BCH-configuration is defined as a block of 511 coded bits for 259 information bits and $t = 30$ correctable errors. A second configuration consists of a block-length of 255 bits for 131 information bits and $t = 18$. To compare the BCH code and the convolutional code, a bit sequence of approximately the same length (512 coded bits) was passed to the convolutional decoder. A direct comparison of the bit error rates is given in Figures 6.7 and 6.8 for 2 and 4 subchannels, respectively. For both configurations the FEC code-rate is held constant at $r = 1/2$ with 16-QAM and 4-QAM for 2 and 4 subchannels, respectively. The outer code dimension of the LSD code was $Nc = 8$, $N_{TX} = N_{RX} = 4$ transmit and receive antennas.

It is clearly visible that the BCH codes perform better in both cases, however at the cost of higher complexity, as will be shown below. Further it can be stated that for low SNR the performance of the low complexity codes is equal or higher. This means investments in code complexity are rewarded in the high SNR regime after the intersection of the low- and high-complexity curves for both BCH and convolutional codes. For the range considered it follows from comparing both figures that the configuration with 4 subchannels and smaller symbol-alphabet size (i.e 4-QAM) outperforms the configuration with 2 subchannels and higher symbol-alphabet size (rember that the rates are held constant).

For a detailed comparison of the codes not only the performance but also the complexity needs to be considered. As described in Chapter 5 the complexity is expressed in the number of additions necessary to decode these bits. The complexity can be calculated as in (5.24) for the convolutional codes and (5.17) for the BCH codes. Note that for a fair comparison of the BCH code with block-length 255 and the other codes its complexity needs to be doubled - due to the blocksize 255 which is only half as large as the others. The numeric results are listed in Table 6.1.

Clearly, the convolutional codes have a much lower complexity than the BCH codes.

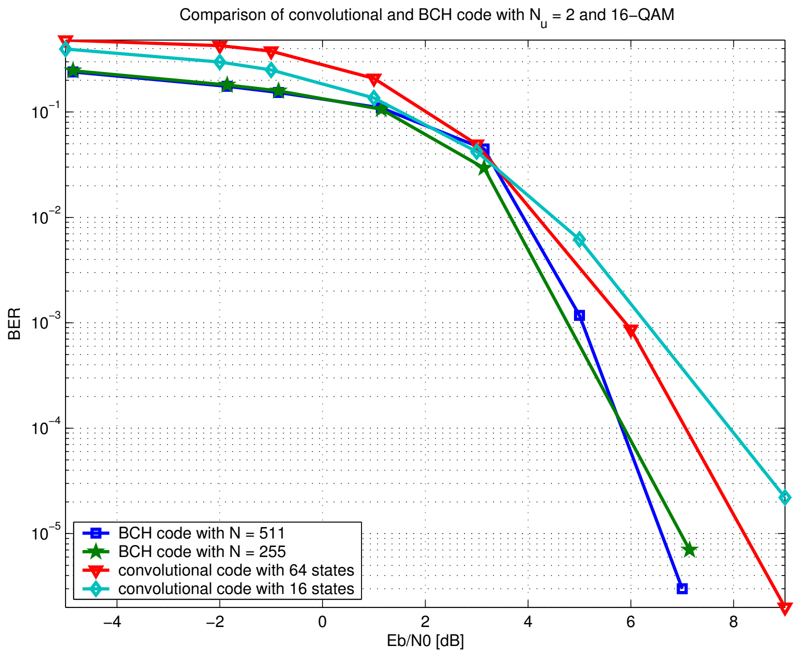

Figure 6.7: *Bit error rate for a system with 2 subchannels, $Nc = 8$, $N_{TX} = N_{RX} = 4$*

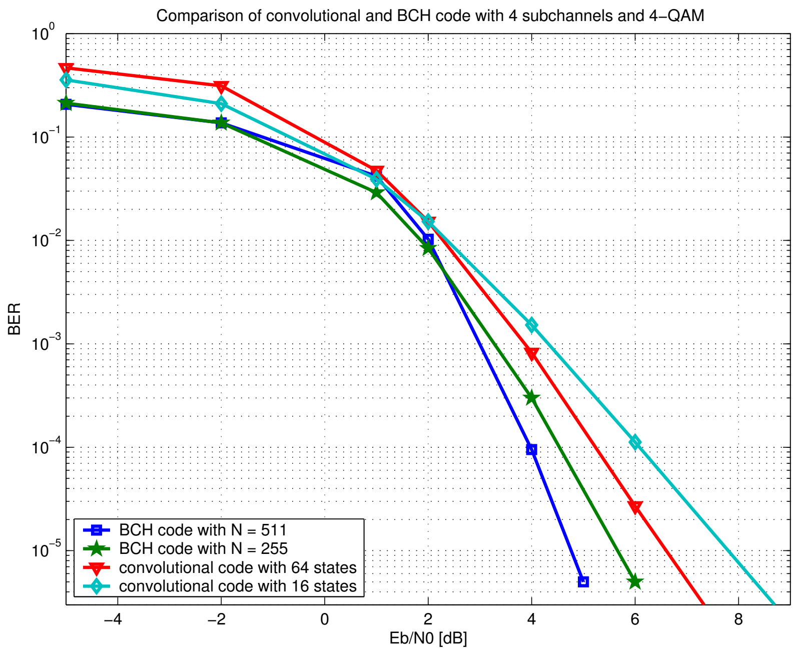

Figure 6.8: *Bit error rate for a system with 4 subchannels, $Nc = 8$, $N_{TX} = N_{RX} = 4$*

code	complexity (No. of additions)
BCH, N=511	$974 \cdot 10^3$
BCH, N=255	$600 \cdot 10^3$ (complexity doubled)
convolutional $(n = 2, k = 1, K = 7)$	$163 \cdot 10^3$
convolutional $(n = 2, k = 1, K = 5)$	$41 \cdot 10^3$

Table 6.1: Complexities of the FEC codes

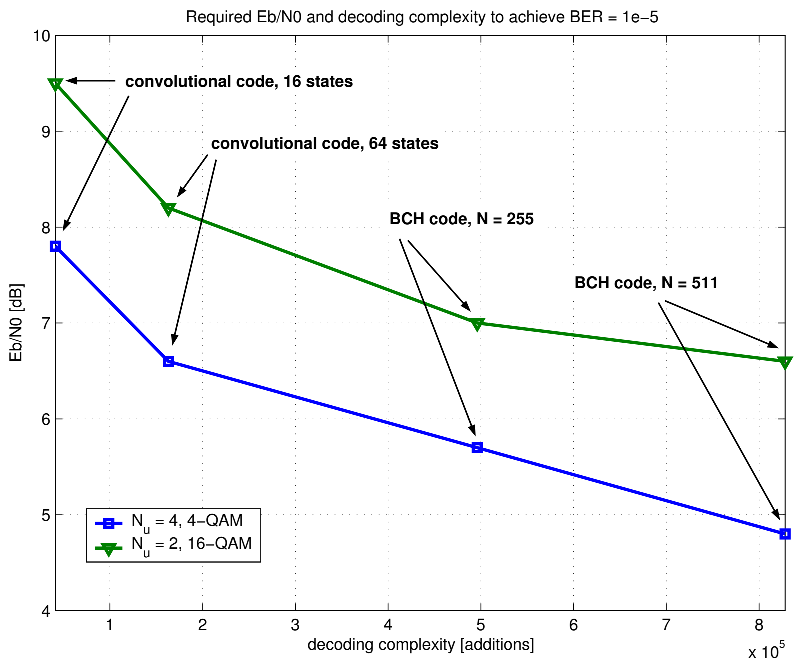

Figure 6.9: *Comparison of the complexities of BCH and convolutional code configurations*

For a visual representation we determined the SNR necessary to achieve a BER of 10^{-5} and depicted it with the corresponding complexity in Figure 6.9. It is visible that for higher complexity-codes a smaller SNR is necessary which means less power is needed to operate the system. The comparison of Figure 6.9 with Figures 6.7 and 6.8 however shows, that the situation changes for higher bit error rates, i.e. the lower complexity codes are positioned higher and the graph will be concave instead convex. For extremely high SNR it might be possible that the curve for two subchannels will be lower than the one for four subchannels since the slope of the curves in Figure 6.7 (i.e. 2 subchannel-configuration) is much steeper than the one in Figure 6.8.

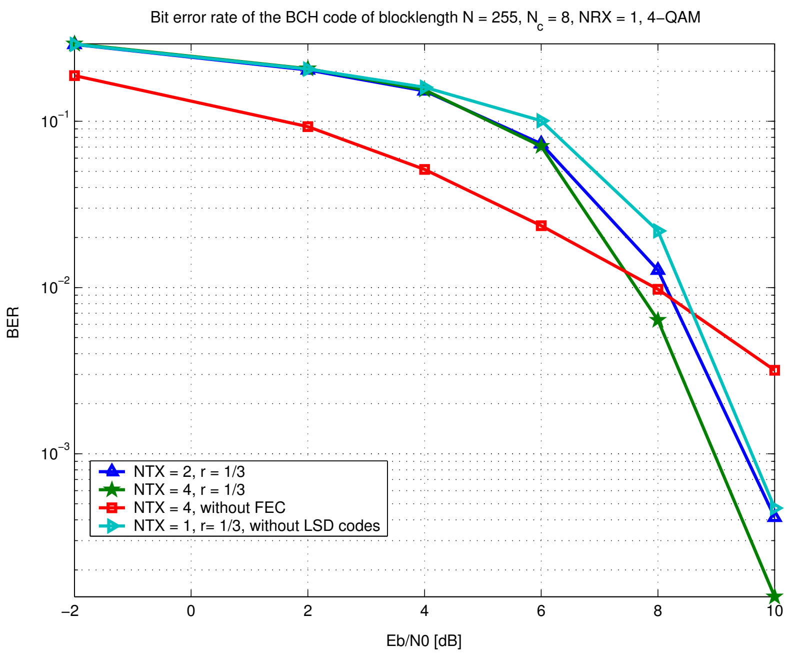

Figure 6.10: *Bit error rate in function of* $\frac{E_b}{N_0}$ *with pure transmit diversity. FEC code-rate* $r = 1/3, 4 - \mathrm{QAM}$

6.3 Combination of LSD Codes with Forward Error Correction

In the section above, a system combining LSD codes, spatial multiplexing and FEC is discussed. However, we have not yet investigated, if such a combination is reasonable. This will be done in this section. For this purpose a system with pure transmit diversity - i.e. several transmit antennas and only one receive antenna - is considered. Several simulations are performed in order to investigate the combination of LSD codes with FEC. In all these simulations a BCH codes of block-length $N = 255$ and rates $r = 87/255 \approx 1/3$ respectively $r = 171/255 \approx 2/3$ in combination with 4-QAM symbols are used. The number of transmit antennas varies.

Figures 6.10 and 6.11 show the bit error rate for several systems with an FEC of code-rate $r = 1/3$ and $r = 2/3$, respectively. The number of transmit antennas N_{TX} is 2 and 4. For comparison, also the bit error rate curve of a system without FEC and the curve of a system without LSD codes is shown. (In the last case, the information bits are coded, mapped to 4-QAM symbols and then directly transmitted over the fading channel.)

One can see that in the low SNR regime the curve corresponding to the system without FEC shows the lowest bit error rate. However, in the high SNR regime this system

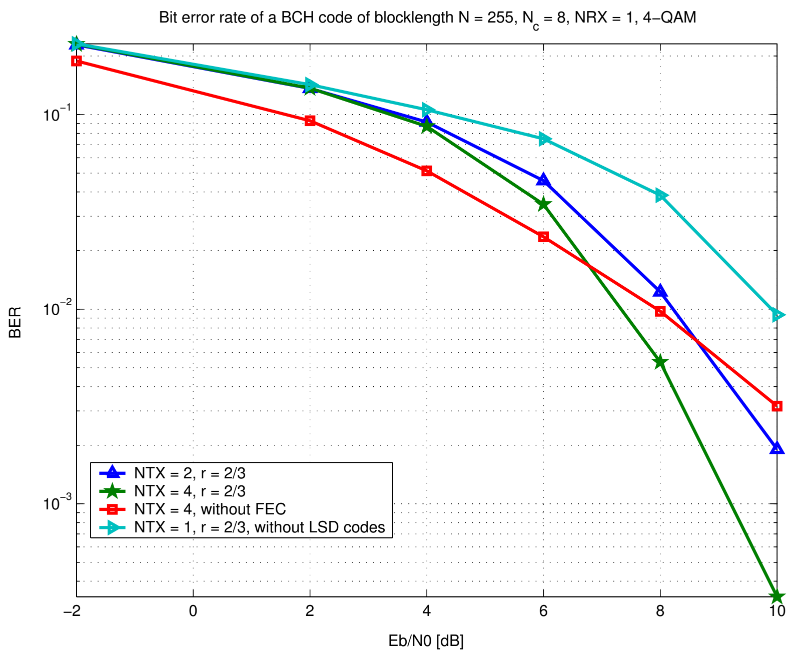

Figure 6.11: *Bit error rate in function of* $\frac{E_b}{N_0}$ *with pure transmit diversity. FEC code-rate* $r = 2/3, 4 - \text{QAM}$

performs worse than the other systems. In addition, it can be seen that in the case where the FEC code-rate $r = 1/3$, the use of FEC without LSD codes results in a performance that is nearly as good as the performance obtained with a system using FEC in combination with LSD codes. In this case, the combination of those codes results only in a small improvement of the bit error rate. However, if the code-rate $r = 2/3$, the combination of FEC and LSD codes achieves a significantly lower error rate than just FEC.

In order to get an intuition about the decoding complexity of pure FEC and LSD coding in the following their decoding complexity is named. The BCH code of rate $r = 1/3$ is able to correct $t = 26$ errors and requires $4.7 \cdot 10^5$ additions, whereas the code of rate $r = 2/3$ corrects up to $t = 11$ errors, which results in a complexity of $1.7 \cdot 10^5$ additions. The MAP-DFE has a decoding complexity of $6.4 \cdot 10^{41}$ additions per input vector $\boldsymbol{r}$ of length $N_c = 8$. Consequently, LSD coding requires $32 \cdot 6.4 \cdot 10^4 = 2 \cdot 10^6$ additions. As can be observed the decoding complexity of the MAP-DFE is significantly higher compared to the complexity of the FEC. However, the use of FEC decreases spectral efficiency. Therefore, it is desirable to use a weak FEC with a high data rate.

Figures 6.12 and 6.13 show the bit error rates for systems with $N_{TX} = 2$ respectively

[1]For the MAP-DFE $\vartheta = 2$ required iterations are assumed which, according to Section 6.1, seems to be a good guess for the high SNR regime.

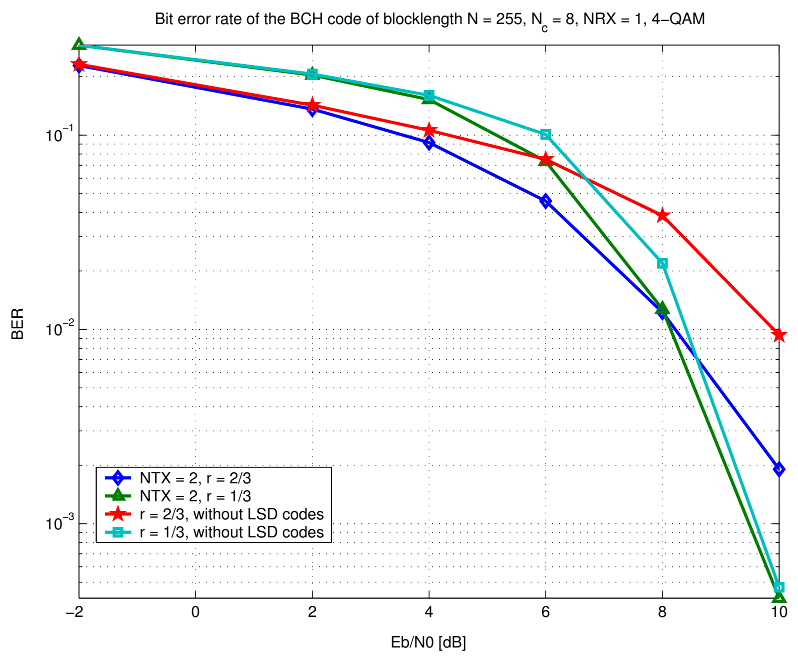

Figure 6.12: *Bit error rate in function of* $\frac{E_b}{N_0}$ *with pure transmit diversity.* $N_{TX} = 2, 4 - \mathrm{QAM}$

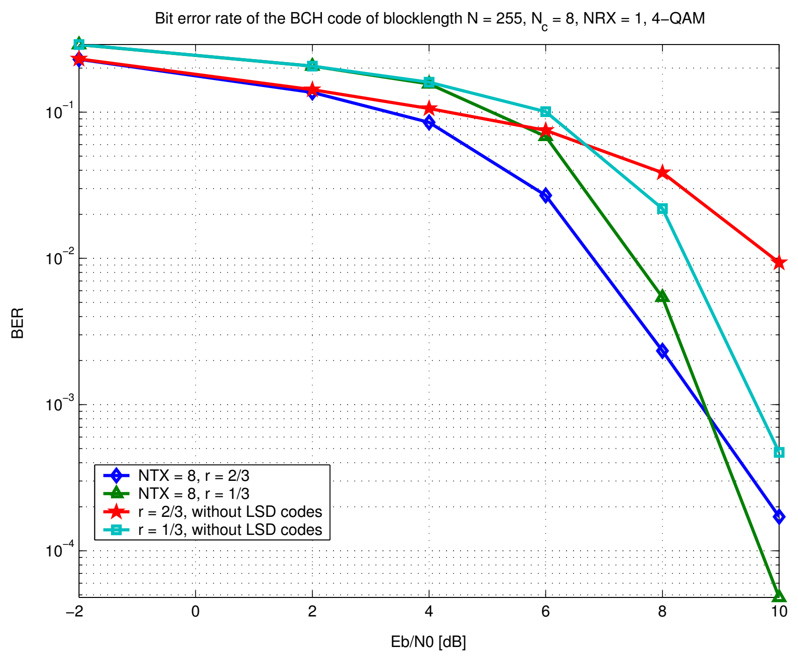

Figure 6.13: *Bit error rate in function of* $\frac{E_b}{N_0}$ *with pure transmit diversity.* $N_{TX} = 8, 4 - \mathrm{QAM}$

$N_{TX} = 8$ transmit antennas. In both cases, FEC with code-rates $r = 1/3$ and $r = 2/3$ is performed. For comparison, in both figures the curves corresponding to a system using only FEC are shown. One can see that in the low SNR regime, the curve corresponding to rate $r = 2/3$ shows a better performance compared to the curve corresponding to rate $r = 1/3$. In the high SNR regime the opposite is true. Consequently, the curves intersect. Consider the intersections of the curves corresponding to the combination of LSD codes and FEC in Figures 6.12 and 6.13. It can be observed that in the case of 8 transmit antennas, this intersection is located at a lower bit error rate than in the case of 2 antennas. Therefore, we reason that for a sufficiently large number of transmit antennas, the combination of LSD codes with a weak FEC outperforms the combination of LSD codes with a strong FEC in the region we are interested in (e.g. where the bit error rate is about 10^{-5}).

Based on the observations above we conclude that in the high SNR regime the combination of LSD codes with FEC outperforms the case where only LSD codes are used. In addition, if a weak FEC with a high data rate is applied, the combination of FEC and LSD codes yields a significantly better performance, compared to a system performing just FEC. Furthermore, the use of LSD codes allows to use a weaker FEC without loss in performance. Consequently, the combination of LSD codes with FEC can improve spectral efficiency. All these statements make a good case for the combination of LSD codes with FEC.

6.4 Symbol-Alphabet vs. Spatial Multiplexing

As described beforehand, the goal of spatial multiplexing is to increase the spectral efficiency, or, equivalently to achieve a higher throughput at the same bandwidth. The question arises if the same effect or even better performance can be achieved by using a higher symbol-alphabet and to use the antennas for diversity instead of multiplexing. In fact, in [9] it was shown that for the same considered system ($Nc = 8, N_{TX} = N_{RX} = 4$, constant data rate) but *without* FEC the introduction of 16-QAM for 2 subchannels (i.e. 2 antennas for diversity) instead of full spatial multiplexing with 4 subchannels (i.e 0 antennas for diversity) and 4-QAM leads to better performance in the high SNR regime. The curves actually intersect in the range of $BER = 10^{-4}$ an area which is still of interest for real systems.

The same simulation was done for the same configuration but with a BCH code with rate $r = 131/255 \approx 1/2$ and block-length 255. The results are visible in Figure 6.14. The effect of the FEC ist that the intersection of the 2 and 4 subchannel configurations are moved to an area (i.e. extremely low BER and high $\frac{E_b}{N_0}$) that is not of any interest

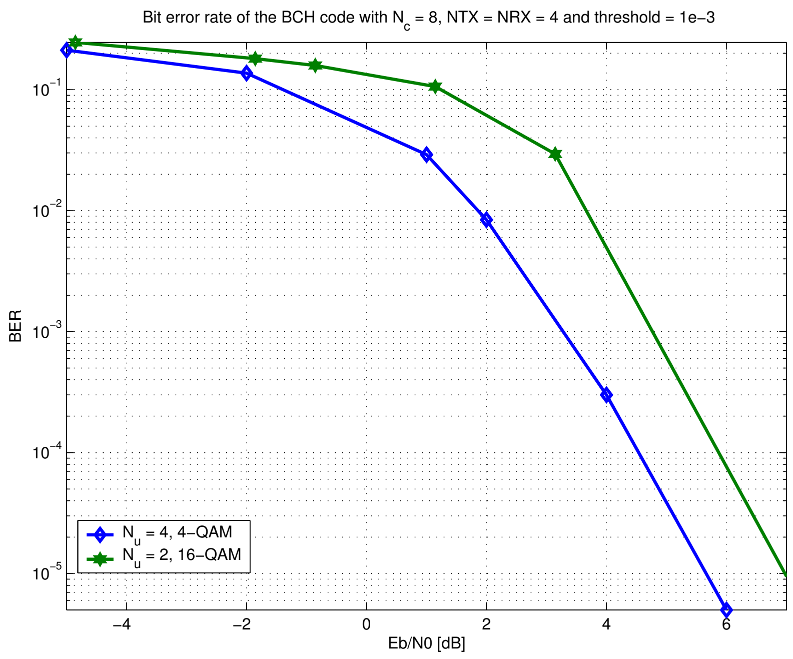

Figure 6.14: *Bit error in function of* $\frac{E_b}{N_0}$ *with constant throughput of* 4 *bits per channel use.* $N_{TX} = 4, N_{RX} = 4$

for our purposes. Thus, unlike in the system without FEC in [9], the configuration with 4 subchannels performs always better in the range of interest. The same behaviour was observed for convolutional codes.

Thus, when using space-time coding in a MIMO system with FEC, spatial multiplexing should be given higher priority opposed to larger symbol-alphabet size to achieve higher throughput.

6.5 FEC Code-Rate vs. Spatial Multiplexing

In Section 6.4 it is stated that spatial multiplexing should be preferred instead of choosing a higher symbol-alphabet. In this section we want to investigate the tradeoff between the code-rate r of the FEC and spatial multiplexing. All simulations are performed using a BCH code of block-length 255. The system transmits 4-QAM symbols and the throughput is held constant (i.e. the choice of a strong FEC with a low data rate is compensated using more subchannels). Figures 6.15 and 6.16 show the bit error rate in function of $\frac{E_b}{N_0}$. In both figures the used BCH codes have code rates $r = 2/3$, $r = 1/3$ and $r = 1/6$.

In Figure 6.15 a system is considered with 4 transmit and receive antennas and a total

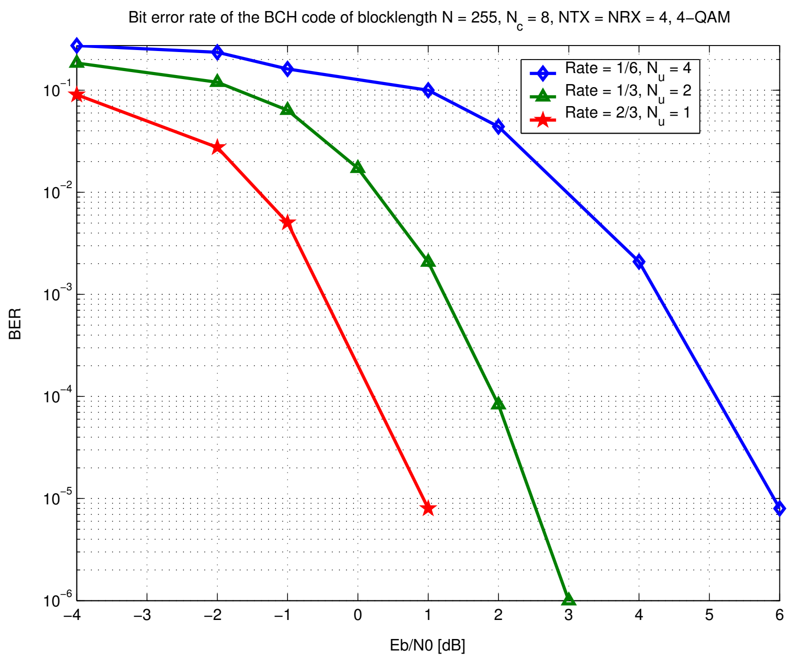

Figure 6.15: *Bit error rate in function of* $\frac{E_b}{N_0}$ *with constant throughput of* $2/3$ *bits per channel use.* $N_{TX} = 4, N_{RX} = 4, 4 - \text{QAM}$

throughput of $2/3$ bits per channel use. In order to achieve a constant throughput, $N_u = 4$ subchannels correspond to the code of rate $r = 1/6$, $N_u = 2$ subchannels correspond to the code of rate $r = 1/3$ and the system using an FEC of rate $r = 2/3$ applies no spatial multiplexing. It should be noted that the less subchannels are used, the higher is the achievable diversity gain. We can see that the curve corresponding to a FEC of rate $r = 2/3$ using no spatial multiplexing shows the best performance. Therefore, it can be observed that the lower diversity gain and the intersymbol interference (ISI) caused by spatial multiplexing has a wider influence on the bit error rate than the gain due to a stronger FEC (i.e. FEC with a lower data rate).

The same behavior can be observed in Figure 6.16. Here, a system with 16 transmit and receive antennas and a total throughput of $8/3$ bits per channel use is considered. Consequently, $N_u = 16$ subchannels correspond to the code of rate $r = 1/6$, $N_u = 8$ subchannels correspond to the code of rate $r = 1/3$ and the system using an FEC of rate $r = 2/3$ transmits the symbols over 4 subchannels. Again, the curve corresponding to the system using the weakest FEC (i.e. the FEC with the highest data rate) shows the best performance. This leads us to the same observation as in the case above, where the throughput was $2/3$ bits per channel use.

From these observations follows that it is reasonable to use a weaker FEC instead

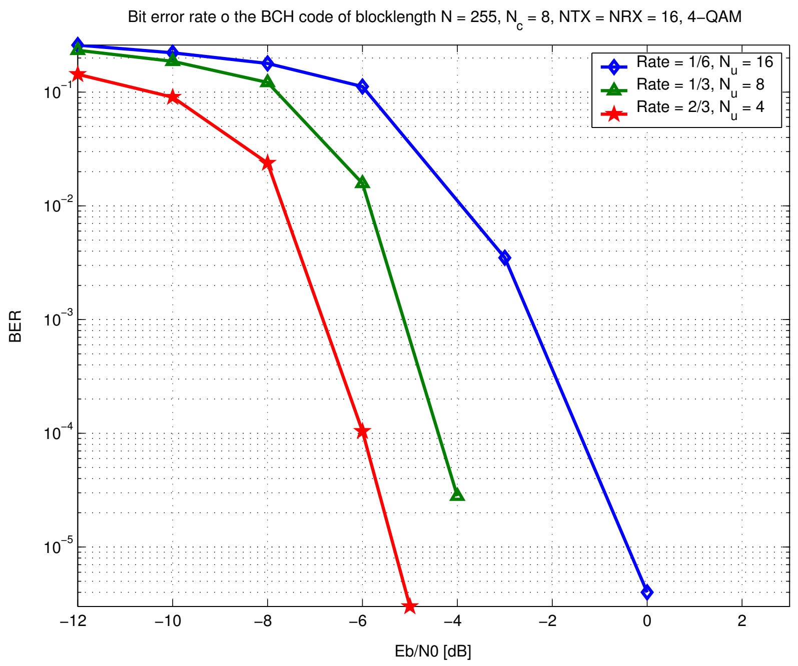

Figure 6.16: *Bit error rate in function of* $\frac{E_b}{N_0}$ *with constant throughput of* $8/3$ *bits per channel use.* $N_{TX} = 16, N_{RX} = 16, 4 - \mathrm{QAM}$

of increasing the number of subchannels in order to achieve a larger throughput. Consequently, if one has to decide between diversity in combination with a weaker FEC or spatial multiplexing in combination with a stronger FEC, it is beneficial to choose higher diversity.

Chapter 7

Outlook and Conclusions

Future wireless communication systems will demand high data rates for delivery of data like multimedia content. The use of MIMO systems (multiple input multiple output) with multiple transmit and/or receive antennas is one technology to support these high data rates in a rich-scattering wireless channel.

The use of multiple transmit and receive antennas enables the introduction of space-time coding to improve link reliability and spatial multiplexing to increase spectral efficiency. Space-time codes introduce spatial and temporal correlations into the signal to benefit from diversity. Gain from diversity can be achieved either in the sender (*transmit diversity*), the receiver (*receive diversity*) or in both, sender and receiver. Spatial multiplexing increases the data rate at constant bandwidth by opening several data streams over the different antennas.

Recently, a class of codes was proposed in [14] which makes it possible to combine space time coding and spatial multiplexing. These *linear scalable dispersion (LSD) codes* are designed in a way that allows to flexibly trade off between improved link reliabity (i.e. diversity) and increased spectral efficiency (i.e. spatial multiplexing). The proposed design for the LSD codes includes an optional forward error correction (FEC). So far, no research has been done on the effects of this FEC block on the performance of the whole system.

Thus, the goal of this project was to investigate the combination of LSD codes with FEC. LSD codes and FEC involve a large number of parameters and, consequently, a wide range of optimization. In order to study the tradeoff between some of these parameters simulations were performed. The results showed that the combination of LSD codes with FEC yields a significant improvement in terms of a lower bit error rate especially if a weak FEC (i.e. an FEC with a high code-rate) is performed. Hence, the combination of LSD codes with FEC is reasonable in order to improve link reliability. Spectral efficiency can be increased by spatial multiplexing or a higher symbol-alphabet. With the number of antennas held constant, spatial multiplexing outperforms the choice of a higher symbol-alphabet and diversity. However, the combination of diversity and a higher FEC code-rate even outperforms spatial multiplexing in terms of link quality (at the same spectral efficiency and with constant number of antennas).

In this work, some of the tradeoffs in the combination of LSD codes with FEC were investigated. However, this project only covered a small part of the whole topic. For example, only FEC with convolutional and BCH codes was studied. It would be interesting to extend the research to other codes like turbo or LDPC codes and to see if their combination with LSD codes is still reasonable. Furthermore, we do not yet understand exactly, how to distribute the degrees of freedom in MIMO systems on diversity vs. spatial multiplexing to reach best performance. Consequently, in order to learn more about the combination of LSD codes with FEC it will be necessary to consider more tradeoffs in this system. So far however, the results raise suspicion that the combination of LSD codes with FEC is a reasonable tool in order to improve link reliability and spectral efficiency.

Appendix A

Fields

A field consists of a set of elements that has two arithmetic operations defined on its elements, namely, addition and multiplication, that satisfy the following properties (by [12]):

Addition:

1. The set is closed under addition, i.e., if $a, b \in F$, then $a + b \in F$.
2. Addition is associative, i.e., if a, b and c are elements of F, then $a+(b+c) = (a+b)+c$.
3. Addition is commutative, i.e., $a + b = b + a$.
4. The set contains an element called *zero* that satisfies the condition $a + 0 = a$.
5. Every element in the set has its own negative element. Hence, if b is an element, its negative is denoted by $-b$. The substraction of two elements, such as $a - b$ is defined as $a + (-b)$.

Multiplication:

1. The set F is closed under multiplication, i.e., if $a, b \in F$, then $ab \in F$.
2. Multiplication is associative, i.e, $a(bc) = (ab)c$.
3. Multiplication is commutative, i.e, $ab = ba$.
4. Multiplication is distributive over addition, i.e., $(a + b)c = ac + ab$.
5. The set F contains and element, called the *identity*, that satisfies the condition $a(1) = a$, for any element $a \in F$.

6. Every element of F, except zero, has an inverse. Hence, if $b \in F(b \neq 0)$, then its inverse is defined as b^{-1}, and $bb^{-1} = 1$. The division of two elements, such as a/b, is defined as ab^{-1}.

Bibliography

[1] S. Alamouti, “A simple transmit diversity technique for wireless communications”, *IEEE J. Select. Areas Commun.* , vol. 16, pp. 1451-1458, October 1998.

[2] J. Becker, H.-J. Dreyer, W. Haacke, R. Nabert, “Mathematik für Ingenieure”, *B. G. Teubner Stuttgart*, 1985.

[3] R. E. Blahut, “Theory and practice of error control codes”, *Addison-Wesley Publishing Company*, 1983.

[4] G. Foschini, Jr. and M. Gans, “On limits of wireless communication in a fading environment when using multiple antennas”, *Wireless Pers. Commun.*, vol. 6, pp. 311-335, Mar. 1998.

[5] G. D. Golden, G. J. Foschini, R. A. Valencuela, P. W. Wolniasky, “Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture”, *Electronic Letters*, vol. 17, November 1998.

[6] G. H. Golub, C. F. Van Loan, “Matrix computations”, *The Johns Hopkins University Press*, 1889.

[7] F. G. Gustavson, “Analysis of the Berlekamp-Massey linear feedback shift-register synthesis algorithm”, *IBM Journal of Research and Development*, vol. 20, nr. 3, p. 204, 1976.

[8] M. Kuhn and A. Wittneben, “A new scalable decoder for linear block codes with intersymbol interference”, *Proc. 55th IEEE Veh. Tech. Conf.* , vol. 4, pp. 1795-1799, May 6-9, 2002.

[9] M. Kuhn, I. Hammerstroem and A. Wittneben, “Linear scalable space-time codes: Tradeoff between spatial multiplexing and transmit diversity”, *IEEE Sig. Proc. Workshop on Advances in Wireless Comm. (SPAWC)*, Rome, 2003.

[10] T. H. Liew and L. Hanzo, “Space-time codes and concatenated channel codes for wireless communications”, *Proc. IEEE*, vol. 90, pp. 187-219, Feb. 2002.

[11] J. D. Lipson, “Elements of algebra and algebraic computing”, *Addison-Wesley Publishing Company*, 1981.

[12] J. G. Proakis, “Digital communications”, *McGraw-Hill*, vol. 4, 2001.

[13] M. J. S. Smith, “Application-specific integrated circuits”, *Addison-Wesley Publishing Company*, 1997.

[14] A. Wittneben and M. Kuhn, “A new concatenated linear high rate space-time block code”, *Proc. 55th IEEE Veh. Tech. Conf.* , vol. 1, pp. 289-293, May 6-9, 2002.

[15] R. Zurmühl, S. Falk, “Matrizen und ihre Anwendungen, Teil 2: Numerische Methoden”, *Springer Verlag*, 1986.