ulm university universität
uulm

Faculty of Engineering
Institute of Telecommunications
and Applied Information Theory

Winter Term 09/10

Study Thesis

October 2009

Turbo Equalization Performance:
The Effect of Precoding and Application of Turbo Codes

Candidate: Christian Häger
Course of Studies: Electrical Engineering
Matrikelnummer: 589903

Supervisor: Dipl.-Ing. Axel Heim
Examiner: Prof. Dr.-Ing. Martin Bossert

Abstract

In this thesis the performance of a turbo equalization scheme is analyzed. First the BCJR algorithm [1] is presented and applied in both the decoder and equalizer. With these two modules a receiver that employs successive equalization and decoding is established. It is shown that the assumptions made to connect the (taken by themselves optimal) modules lead to suboptimal performance. With the knowledge that a discrete intersymbol interference (ISI) channel can be regarded as a rate-1 convolutional encoder a new receiver structure is derived in the same fashion as an iterative decoder would decode a serially concatenated convolutional code [2]. The major shortcoming of this new receiver is a lower bound of the bit error rate which can be overcome by precoding in order to make the ISI channel recursive [3]. This receiver structure allows decoding performance close to the information theoretical limit of an ISI channel. The outer convolutional encoder is then replaced by a parallel concatenated convolutional encoder in order to see if the additional complexity caused by yet another BCJR module and an extra "inner" loop necessary to iteratively decode the outer code can yield performance improvements.

Contents

Chapter 1

Introduction

When transmitting digital data wirelessly we commonly face two problems. Firstly amplifiers have to be installed in order to access the usually weak radio signal at the receiver. An amplifier does not only amplify the signal but also adds noise, thus bit errors might occur as a consequence. Secondly reflections cause multipath propagation and the received signal may not be the transmitted one but rather a superposition of time-shifted and weighted versions of it.

To counter the first problem information bits are encoded by means of a channel encoder for forward error correction. For the second problem an equalizer in the receiver tries to recover the transmitted signal with a certain reliability depending on the noise power. Equalization and channel decoding are often implemented separately, i.e. first an equalizer recovers the transmitted signal and then a channel decoder obtains an estimate of the information bit sequence.

In [4] a method is presented to decode a bit sequence which is protected by two independent error-correcting codes. Two decoders are used (one for each code) which are interchanging information with each other and are operated in an iterative manner. This method is referred to as the turbo decoding principle and offers error correcting performance close to the theoretical boundaries postulated by Shannon [5].

A multipath channel can also be regarded as an encoder so from a more abstract point of view an encoded information bit sequence transmitted over a multipath channel is then also protected by two independent codes – the one employed for error-correction and the code of the channel. Therefore the above mentioned turbo principle can be applied to the problem of equalization and decoding. This so called turbo equalization scheme was first proposed in [6]. The fact that it performed better than the classical approach of separate equalization and decoding inspired many people to further improve the original scheme [7, 2, 8] and especially the use of a precoder [9, 3] lets this scheme perform very close to the theoretical limits.

This thesis is structured as follows. The basic elements and algorithms that are necessary to implement and understand a digital transmission line in which multipath propagation occurs are presented in Chapter 2. In Chapter 3 the general turbo principle is introduced. Then this principle is applied to the problem of equalization and decoding. The resulting turbo equalization scheme is tested and analyzed through simulation. In Chapter 4 a parallel concatenated convolutional code which is commonly known as a turbo code is employed for error-correction as proposed in [10]. The performance of this scheme is compared to the conventional turbo equalization scheme. Finally in Chapter 5 the results are summarized.

Chapter 2

Presentation of the Transmission Line

2.1 Overview

A digital transmission line consists of a transmitter, a channel and a receiver. In a nutshell: The transmitter maps digital data to channel symbols which are sampled and modulated in order to get a continuous-time signal. This signal is distorted by the channel. The receiver demodulates and samples the distorted signal and then tries to provide an estimate of the original digital data. The whole process of sampling, modulating and demodulating is a very complex issue for itself and we refer to standard literature for deeper analysis [11, 12]. Nonetheless it is possible to simplify this transmission line significantly which will be described in this section.
In the introduction a multipath propagation channel is mentioned. A situation giving rise to multipath propagation is depicted in Figure 2.1. The signal $s(t)$ is sent but due to reflections and attenuations

$$g(t) = a_0 s(t - \tau_0) + a_1 s(t - \tau_1) \tag{2.1}$$

is received, where a_0 and a_1 are attenuation factors and τ_0 and τ_1 are the transmission times for the two different paths. In this thesis we consider attenuation factors and delays which are independent of the time. Thus the channel can be modeled by a linear time invariant system where the impulse response of the system is described by

$$h(t) = a_0 \delta(t - \tau_0) + a_1 \delta(t - \tau_1), \tag{2.2}$$

i.e. applying convolution $s(t) * h(t)$ results in the received signal $g(t)$. This is only a short

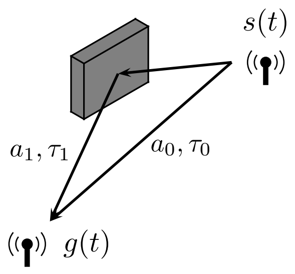


Figure 2.1: Typical multipath propagation for wireless transmission.

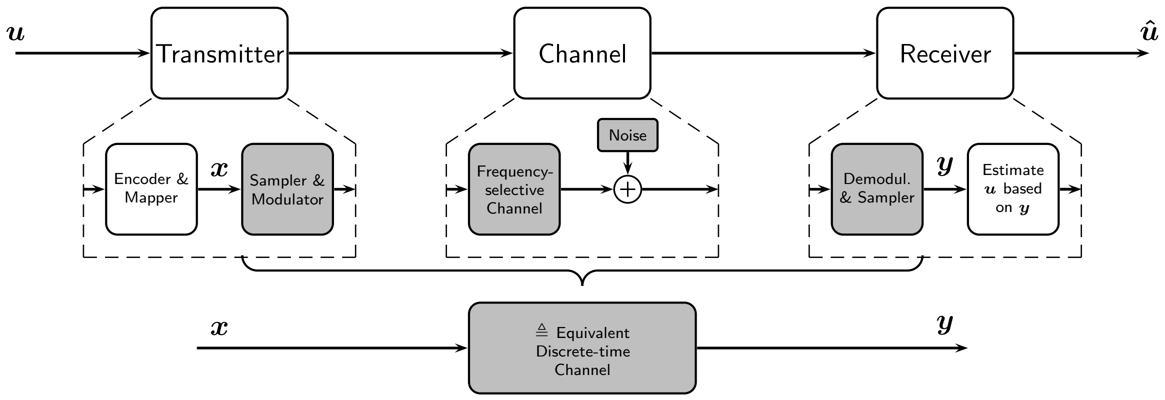


Figure 2.2: A general digital transmission line and the equivalent discrete-time channel replacing the modulator, continuous-time channel and demodulator. Note that in this model the amplifier (and thus the additive noise) is treated as part of the channel.

example and in fact a wireless communication scheme is not the only case where such distortions take place. Transmissions over optical wires where refractions and reflexions appear are another example. All of these channels are said to be frequency selective (transforming $h(t)$ in the above example into frequency domain and taking its absolute value results in a frequency dependency) and they all cause intersymbol interference. That means in the receiver a sample value corresponds not only to one transmitted channel symbol but to an overlapping of successive channel symbols.

In [11] it is shown that the whole cascade of modulator, continuous time frequency selective channel and demodulator can be replaced by a simple equivalent discrete-time channel as pointed out in Figure 2.2 (for a detailed derivation and necessary assumptions see [11, Chapter 10]). A very important requirement is that the channel is time invariant, i.e. its properties are not changing with time. The usually very complex steps which lead from the sequence of channel symbols $\boldsymbol{x}$ to the demodulated sample values $\boldsymbol{y}$ can then be put into the simple equation

$$y_k = h_0 x_k + h_1 x_{k-1} + \cdots + h_L x_{k-L} + n_k \tag{2.3}$$

where $h_0, \ldots, h_L$ represent the tap gains of the discrete-time intersymbol interference channel (ISI channel), L is the length or memory of the channel and n_k is a noise sample. This equation is further analyzed later in this chapter.

The resulting transmission line used in this thesis is shown in Figure 2.3. In the remainder of this chapter every box in this picture will be described in detail.

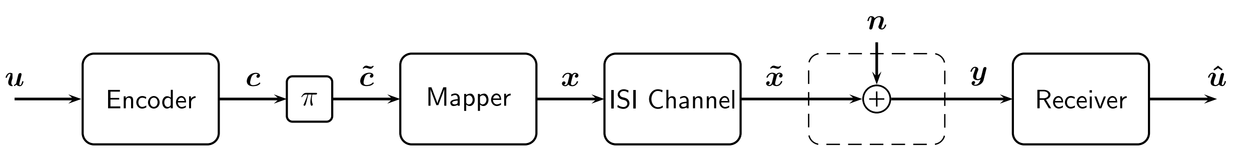
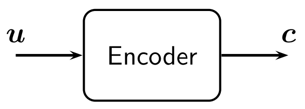


Figure 2.3: Discrete-time model of the transmission line used in this thesis.

2.2 Source

The source generates an information vector $\boldsymbol{u} = (u_1, \dots, u_k)$ containing the symbols 0 and 1 with equal probability. That means $\boldsymbol{u}$ is a binary vector of length k and we write $\boldsymbol{u} \in \mathbb{F}_2^k$, where $\mathbb{F}_2 = \{0, 1\}$.

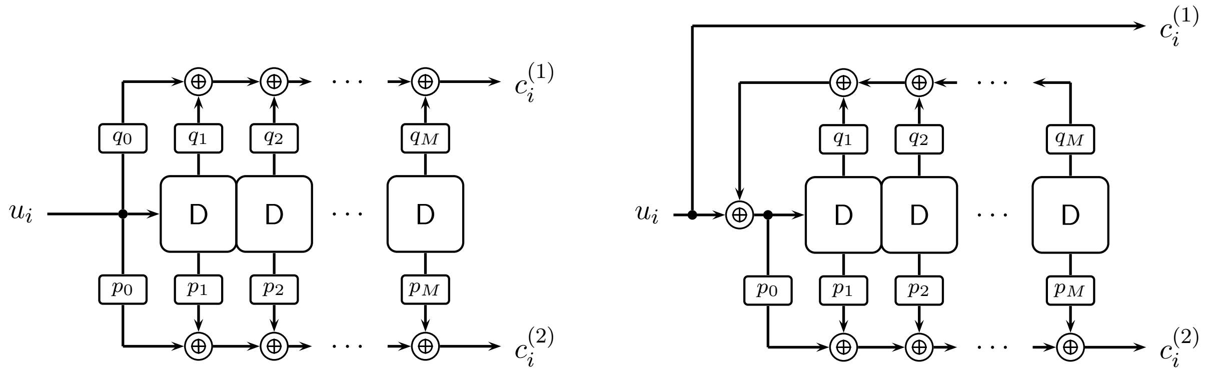
2.3 Encoder



The channel encoder uniquely maps the information vector $\boldsymbol{u}$ to a codeword vector $\boldsymbol{c} = (c_1, \dots, c_n) \in \mathbb{F}_2^n$ of length n, where $n \geq k$. We define $R = k/n$ to be the code rate. The code $\mathcal{C}$ is the set of all 2^k codeword vectors which can occur when encoding all 2^k possible information vectors for a given encoder.
In this paper convolutional encoders are used which can be represented by a sequential circuit consisting of M delay elements. In order to keep the notation as simple as possible those circuits will be limited to one input and two outputs. The rate R is then fixed to 1/2 and M is called the memory or the constraint length of the encoder.
In order to emphasize that at any given time i the ith element of $\boldsymbol{u}$ is mapped to *two* code bits, $\boldsymbol{c}$ can also be written as

$$\boldsymbol{u} = (u_1, \dots, u_k) \quad \mapsto \quad \boldsymbol{c} = (\boldsymbol{c}_1, \dots, \boldsymbol{c}_k) = ((c_1^{(1)}, c_1^{(2)}), \dots, (c_k^{(1)}, c_k^{(2)})). \tag{2.4}$$

The upper number refers to the corresponding output of the encoder. The sequence which corresponds to the first and second output will be denoted as $\boldsymbol{c}^{(1)} = (c_1^{(1)}, \dots, c_k^{(1)})$ and $\boldsymbol{c}^{(2)} = (c_1^{(2)}, \dots, c_k^{(2)})$, respectively.
Figure 2.4 shows two types of convolutional encoders. The first one (a) is a non systematic convolutional (NSC) encoder and the second one (b) is a recursive systematic convolutional (RSC) encoder generating the same code $\mathcal{C}$. A rate-1/2 convolutional encoder is systematic when $u_i \mapsto (u_i, c_i^{(2)})$ or $u_i \mapsto (c_i^{(1)}, u_i)$ for $i = 1, \dots, k$. That means the (input) information bit of the encoder appears also as part of the (output) vector $\boldsymbol{c}_i$ for each i.
The encoder taps $q_0, \dots, q_M$ and $p_0, \dots, p_M$ can only take values 0 and 1. In Figure 2.5 an example NSC encoder with parameters $M = 2$, $q_0 = 1$, $q_1 = 0$, $q_2 = 1$, $p_0 = 1$, $p_1 = 1$ and $p_2 = 1$ is shown.



(a) NSC encoder **(b)** RSC encoder

Figure 2.4: General structure of convolutional encoders with rate 1/2. Note that modulo-2 operators are used.

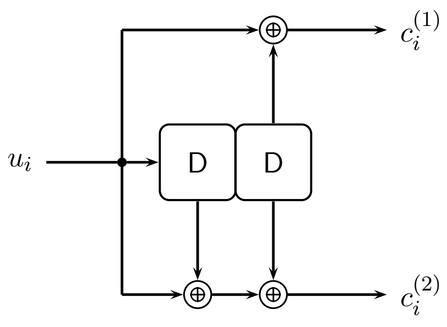


Figure 2.5: Sequential circuit representation of the (5,7) NSC encoder.

It is common to declare the used encoder in octal notation. This can be done by simply interpreting the taps as a binary number and writing them as an octal number. For the example encoder in Figure 2.5 this leads to $(q_0q_1q_2)_2 = (101)_2 = 5_8$ and $(p_0p_1p_2) = (111)_2 = 7_8$. So Figure 2.5 is a graphical representation of the $(5, 7)$ NSC encoder.
We use the convention that for $i < 1$ all delay elements contain zeros. The encoder is said to be initialized in the all-zero state.
Leaving the delay elements in an undetermined state after encoding is obstructive when decoding the received code vector [13]. Therefore the codeword vector is terminated by adding a sequence of M additional bits to the information vector, so that after having encoded all $k + M$ bits the encoder is left in a determined state. In this thesis the encoder is forced back to the all-zero state. When the encoder is not recursive M zeros are added to the information vector. For recursive encoders the corresponding tail-bits can be stored in a look up table [13]. Termination results in an overall rate loss because now for k information bits $2(k + M)$ code bits have to be sent. This loss can be neglected if k is large. To keep the notation simple we further assume that $\boldsymbol{u} \in \mathbb{F}_2^k$ already contains the tail-bits as actual information.

2.3.1 State Diagram and Trellis Representation

A convolutional encoder with M delay elements can also be regarded as a finite state machine having 2^M states $S = \{0, \ldots, 2^{M-1}\}$. The current state of the encoder is determined by "reading" the content of the delay elements as a dual number. So if the first delay element of the example encoder contains a 1 and the second one contains a 0 the encoder is in state $(10)_2 = 2$.
The output of the encoder at time i, i.e. $(c_i^{(1)}, c_i^{(2)})$, is determined by the current state σ_i and the current input bit u_i. After generating the output, the state machine enters the next state σ_{i+1}. Figure 2.6 shows the state machine of the (5,7) NSC encoder.

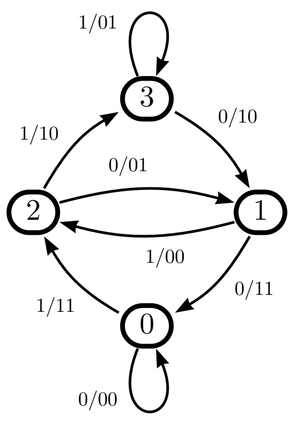


Figure 2.6: State machine of the (5,7) NSC encoder.

The transition labels show the input bit and the two output bits for a particular state transition. A codeword corresponds to a specific path in the state diagram. Starting in the state $\sigma_0 = 0$ the next state σ_1 is entered according to the input bit and two output code bits for that state transition are generated. This is continued until the final state σ_k is reached (which will be $\sigma_k = 0$ due to termination).

A trellis is a directed graph visualizing all possible paths in the state diagram that can occur during encoding. For an example see Figure 2.7 which is the trellis representation of the (5,7) NSC encoder when $k = 6$ information bits are encoded. The bold path corresponds to the information vector $\boldsymbol{u} = (1, 0, 0, 1, 0, 0)$. A dashed line represents a 0 as information bit.
When talking about trellises it is helpful to introduce some conventions for the notation. A trellis consists of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. Every edge $\epsilon \in \mathcal{E}$ connects two vertices $\nu \in \mathcal{V}$. If we define the direction of the trellis from left to right, the (left) start vertex is denoted by $A(\epsilon)$ and the (right) end vertex by $\Omega(\epsilon)$. $\mathcal{E}_{\text{in}}(\nu)$ is the set of all edges that end in ν and $\mathcal{E}_{\text{out}}(\nu)$ is the set of all edges that originate from ν, i.e.

$$\mathcal{E}_{\text{in}}(\nu) = \{\epsilon \in \mathcal{E} : \Omega(\epsilon) = \nu\} \qquad \text{and} \qquad \mathcal{E}_{\text{out}}(\nu) = \{\epsilon \in \mathcal{E} : A(\epsilon) = \nu\}. \tag{2.5}$$

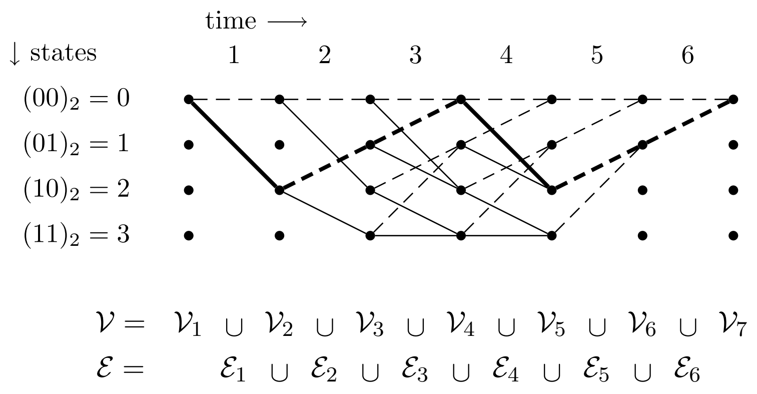

time ⟶
↓ states
1
2
3
4
5
6
$(00)_2 = 0$
$(01)_2 = 1$
$(10)_2 = 2$
$(11)_2 = 3$
$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3 \cup \mathcal{V}_4 \cup \mathcal{V}_5 \cup \mathcal{V}_6 \cup \mathcal{V}_7$
$\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3 \cup \mathcal{E}_4 \cup \mathcal{E}_5 \cup \mathcal{E}_6$

Figure 2.7: Example trellis for the (5,7) NSC encoder when 6 information bits are encoded.

It is helpful to decompose $\mathcal{V}$ and $\mathcal{E}$ into the disjoint subsets

$$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_{k+1} \qquad \text{and} \tag{2.6}$$

$$\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \cdots \cup \mathcal{E}_k \tag{2.7}$$

such that every edge $\epsilon \in \mathcal{E}_i$ connects a vertex $\nu \in \mathcal{V}_i$ to a vertex $\nu \in \mathcal{V}_{i+1}$. In Figure 2.7 those sets are already depicted as an example.
A vertex $\nu \in \mathcal{V}_i$ represents one possible state the encoder can be in when u_i is encoded. An edge ϵ is a tuple $(s_i, s_{i+1}, \lambda(\epsilon), \mu_1(\epsilon), \mu_2(\epsilon))$, where s_i is the state of the vertex it originates from and s_{i+1} is the state of the vertex it ends in. $\lambda(\epsilon)$ is the input information bit that causes the state transition $s_i \rightarrow s_{i+1}$. $\mu_1(\epsilon)$ and $\mu_2(\epsilon)$ refer to the two output code bits for that transition. $\lambda(\epsilon)$ is visualized in trellis diagrams with a dashed line if a $\lambda(\epsilon) = 0$ and with a solid line if $\lambda(\epsilon) = 1$. Figure 2.8 illustrates this formalism.

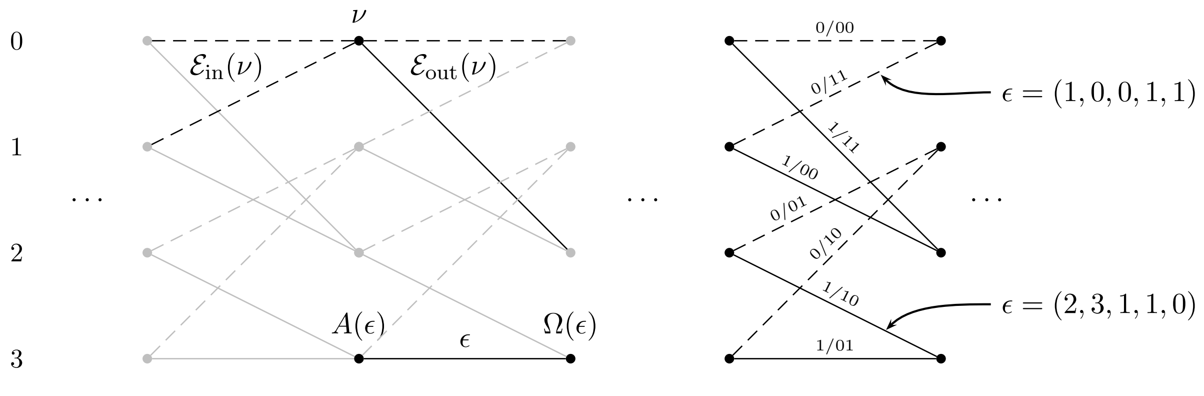
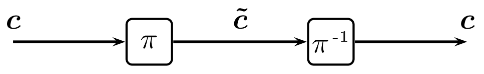

0
1
2
3
ν
$\mathcal{E}_{\text{in}}(\nu)$
$\mathcal{E}_{\text{out}}(\nu)$
$A(\epsilon)$
ϵ
$\Omega(\epsilon)$
. . .
0/00
0/11
1/11
1/00
0/01
0/10
1/10
1/01
$\epsilon = (1, 0, 0, 1, 1)$
$\epsilon = (2, 3, 1, 1, 0)$

Figure 2.8: Visualization of the trellis notation used in this thesis. Note that an edge ϵ contains five pieces of information: starting and ending state, one input and two output bits for a particular state transition.

A sequential circuit, a state diagram and a trellis all describe the same convolutional encoder only from different points of view. The state diagram is the preliminary stage for drawing the trellis and, as will become clear when introducing the receiver, the trellis diagram is the key for efficient decoding.

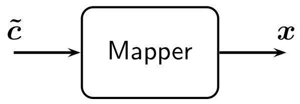
2.4 Interleaver



An interleaver rearranges the order of the bits in the codeword vector $\boldsymbol{c}$. The operation is reversed by a deinterleaver.
When burst errors occur during transmission, i.e. few consecutive received channel symbols are very noisy, restoring the old bit order (deinterleaving) helps to spread out these errors. This is necessary because decoding is impaired when consecutive symbols are heavily distorted. Thus the purpose of an interleaver is to distribute few consecutive extreme noisy channel symbols equally throughout the codeword vector.
The interleaver has another interesting effect. Imagine that consecutive noisy symbols are somehow correlated and that the correlation decreases the further these correlated symbols are apart. By scrambling the order of the symbols it is possible to decrease the correlation between adjacent symbols after deinterleaving. Why this is helpful in the receiver will become obvious later.

Several different types of interleavers exist and in fact interleaver design is a complex issue for itself. Nonetheless for this thesis simple random interleavers (i.e. a device that takes each c_i in $\boldsymbol{c}$ and randomly assigns it to a new position in $\tilde{\boldsymbol{c}}$) are used.

2.5 Mapper



A symbol mapper converts the binary elements of the interleaved codeword vector $\tilde{\boldsymbol{c}}$ to a subspace of the Euclidean vector space $\mathbb{C}$ for subsequent sampling and modulation. For convenience we are assuming binary phase shift keying (BPSK modulation) so that the symbol mapper only performs the operation

$$x_i = (-1)^{\tilde{c}_i}. \tag{2.8}$$

The subspace of $\mathbb{C}$ is thus $\{-1, 1\}$.

2.6 Discrete ISI Channel

As mentioned in the beginning of this chapter the discrete channel we are going to use is described by

$$\tilde{x}_i = \sum_{j=0}^{L} h_j \cdot x_{i-j} \qquad , \tag{2.9}$$

where L is the memory of the channel and $\boldsymbol{h} = (h_0, \ldots, h_L)$ are the channel taps. A graphical representation of this equation can be seen in Figure 2.9.

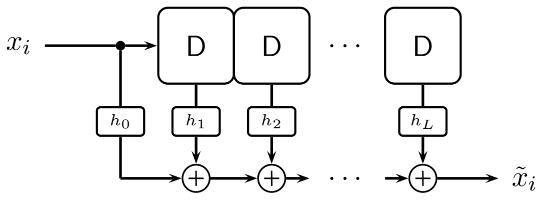


Figure 2.9: Representation of the discrete intersymbol interference channel as a sequential circuit.

The resemblance to an NRC encoder is obvious. The main differences are:

- The channel has only one output $\tilde{x}_i$.
- The input vector $\boldsymbol{x}$ consists of the symbols -1 and +1. Modulo-2 operators are replaced by common addition operators.
- The channel taps $\boldsymbol{h} = (h_0, \ldots, h_L)$ are real numbers. (Assuming normalized signal energy $E_s = 1$ the squared taps have to add up to 1.)
- The input alphabet $\{-1, +1\}$ is a subset of the output alphabet.

The last three points all refer to the fact that the channel does not operate on a finite field ($\mathbb{F}_2$) as it was the case for the channel encoder.
Considering the differences mentioned above the graphical representation in Figure 2.9 still allows us to draw a state diagram and a trellis graph in a similar way as for the encoder. For the example channel $\boldsymbol{h1} = (0.4097,\ 0.8150,\ 0.4097)$ with $L = 2$ which is shown in Figure 2.10 the corresponding state machine and a full trellis transition are depicted in Figure 2.11.
When determining the current state of the channel the symbols -1 or +1 which are stored in the delay elements are mapped back to 0 and 1 before the dual conversion to a decimal number. It can be seen that the state machine is the same for a channel and a (non-recursive) encoder having the same memory $M = L$. Only the transition labels are different because of

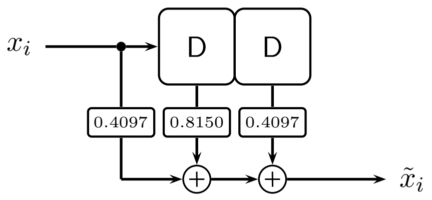
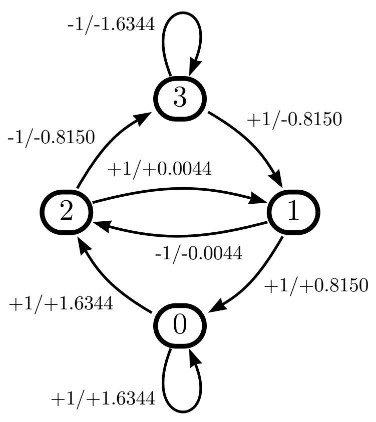
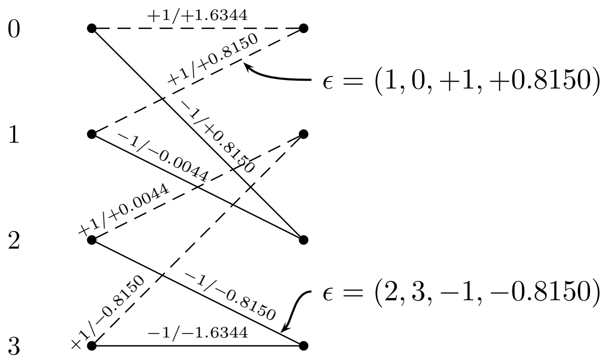


Figure 2.10: Graphical representation of the example channel $\boldsymbol{h1}$.



(a) State machine



(b) Trellis transitions

Figure 2.11: State machine and a full trellis transition for the example channel.

different input and output alphabets. The output alphabet comprises all values of $\tilde{x}_i$ that can occur during transmission. For channel $\boldsymbol{h1}$ the output alphabet is

$$\tilde{x}_i \in \{\pm 0.0044,\ \pm 0.8150,\ \pm 1.6344\}. \tag{2.10}$$

To calculate the alphabet it is assumed that in (2.9) all x_i are 1 for $i < 1$. This is equivalent to the initialization in the all-zero state for an encoder. Initialization for the ISI channel can be achieved by sending L times +1 prior to the real transmission.
For the trellis graph nothing changes except that an edge ϵ is now a tuple $(s_i, s_{i+1}, \lambda(\epsilon), \mu(\epsilon))$ containing only four pieces of information instead of five because only one output has to be considered. $\lambda(\epsilon)$ is visualized by a dashed line when $\lambda(\epsilon) = +1$ and by a solid line when $\lambda(\epsilon) = -1$.

2.6.1 Channel Capacity

In order to evaluate the different receivers in this thesis the bit error rate for various signal to noise ratios E_s/N_0 is simulated where $E_s = 1$ is the normalized signal energy and N_0 is the single sided noise power spectrum density [13]. For an AWGN channel (see Section 2.7) a

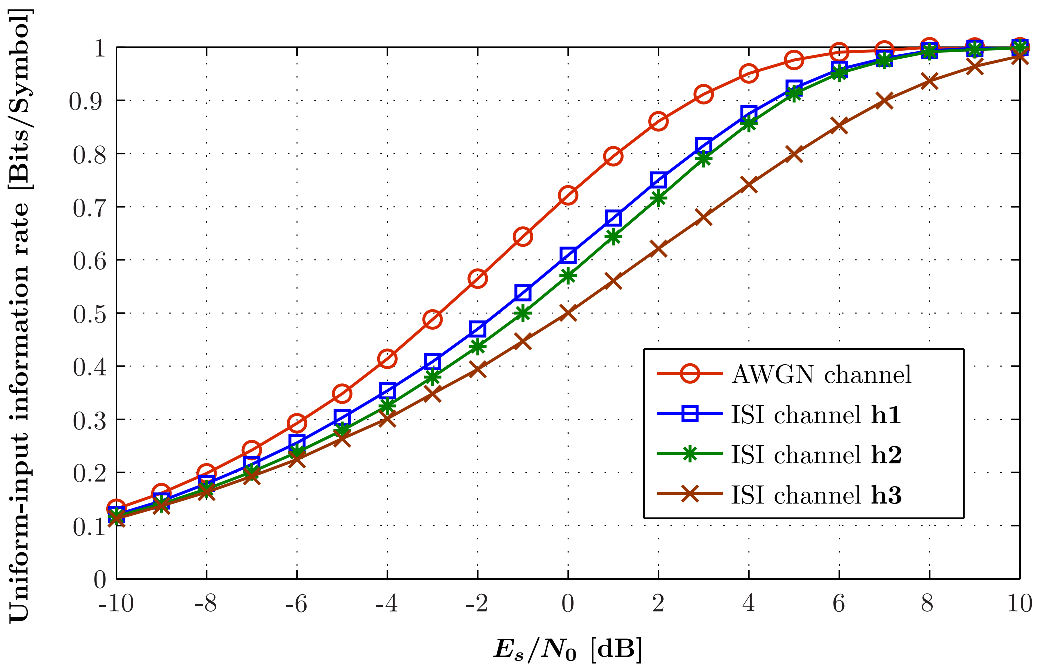


Figure 2.12: Uniform-input information rates for the channels in Table 2.1.

signal to noise ratio corresponds to a certain channel capacity

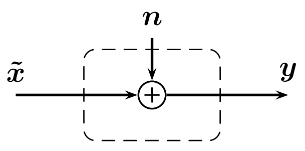
$$C = \frac{1}{2}\log_2\left(1+\frac{E_s}{N_0}\right). \tag{2.11}$$

Information theory then postulates that as long as for the code rate $R < C$ holds it is possible to make the bit error rate arbitrarily small [13]. In our case the code rate is fixed to 1/2, thus we would like to calculate a minimum signal to noise ratio for which near error-free transmission is theoretically possible and then see how the receiver performs above this limit. For $R = C = 1/2$ and an AWGN channel this limit is at $E_s/N_0 = -3\,\text{dB}$. Unfortunately it is not possible to calculate such limits for ISI channels. Nonetheless in [14] a method to calculate the so called uniform-input information rate for such channels is presented. It is shown that this rate coincides with a lower bound of the channel capacity of ISI channels for the case of small memory L. We applied this method to the ISI channels that will be used in the simulations. In Figure 2.12 the uniform-input information rates of these channels are plotted. A summary of the resulting signal to noise ratio limits is shown in Table 2.1.

channel	memory L	E_s/N_0 limit	E_b/N_0 limit
AWGN channel	0	-3.0 dB	0.0 dB
$\boldsymbol{h1} = (0.4097,\ 0.8150,\ 0,4097)$	2	-1.6 dB	1.4 dB
$\boldsymbol{h2} = (\sqrt{0.45},\ \sqrt{0.25},\ \sqrt{0.15},\ \sqrt{0.1},\ \sqrt{0.05},)$	4	-1.0 dB	2.0 dB
$\boldsymbol{h3} = (0.227,\ 0.460,\ 0.688,\ 0.460,\ 0.227)$	4	0.0 dB	3.0 dB

Table 2.1: Minimum signal to noise ratios for various channels in order to transmit 1/2 (information) bits per symbol near error-free by using the uniform-input information rate as a lower bound for the channel capacity. Note that E_b is the energy per information bit and $E_b = E_s/R$.

2.7 Additive White Gaussian Noise (AWGN) Channel



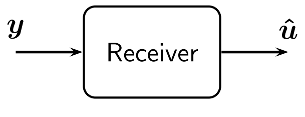
An additive Gaussian noise vector $\boldsymbol{n}$ is added to $\tilde{\boldsymbol{x}}$ where each element in $\boldsymbol{n}$ has zero mean and variance $\sigma^2 = N_0/2$. All elements are independent and identically distributed (i.i.d.). Therefore it is possible to calculate the probability that $\boldsymbol{y}$ is received when assuming $\tilde{\boldsymbol{x}}$ was sent by

$$P(\boldsymbol{y}|\tilde{\boldsymbol{x}}) = \prod_{i=1}^{n} p(y_i|\tilde{x}_i). \tag{2.12}$$

Evaluating the probability densities $p(y_i|\tilde{x}_i)$ can be done with the Gaussian distribution

$$p(y_i|\tilde{x}_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \tilde{x}_i)^2}{2\sigma^2}\right). \tag{2.13}$$

2.8 Receiver



The receiver has to deal with the corrupted channel symbols $\boldsymbol{y}$ and "guess" what was transmitted by providing an estimate of the information vector $\hat{\boldsymbol{u}}$.
The resemblance of an ISI channel and an encoder was already pointed out. Assuming perfect knowledge about the channel taps and a specific information vector $\boldsymbol{u}$ it is then possible to "predict" what happens during transmission. Unfortunately the receiver also has to deal with noise, which only allows to "predict" with a certain reliability depending on how strong the noise is. And that is exactly what the receiver does. It assumes something that could have happened, calculates a probability for this event based on what was received and compares

this probability to all the other events that also could have happened. Then the event with the highest probability is chosen.

Let us recall for a moment what information the receiver can use in order to make a decision. $\boldsymbol{y}$ is observed and the receiver has perfect knowledge about

- the distribution of the source bits,
- which kind of encoder, interleaver and mapper is used,
- the ISI channel taps and
- the noise variance σ^2 of the AWGN channel.

On this understanding the receiver now has to make a decision.

2.8.1 Important Principles

s/s-MAP and s/s-APP

If a decision is made only about one specific information bit the principle above can be put into the decision rule

$$\hat{u}_i = \arg \max_{s \in \{0,1\}} P(u_i = s|\boldsymbol{y}). \tag{2.14}$$

That is observing $\boldsymbol{y}$, calculate the probability for u_i being 1 and 0 respectively. Then make a decision based on the fact which probability is higher.
This principle is referred to as symbolwise maximum a posteriori decision (s/s-MAP). A receiver that does not only provide the decided bits $\hat{u}_i$ but also provides the actual probabilities $P(u_i|\boldsymbol{y})$ and therefore a reliability for its decision is called a symbolwise a posteriori probability receiver (s/s-APP). A decoder and equalizer operating according to the s/s-APP principle is used in this thesis.

Log-Likelihood Algebra

To put the vague word "reliability" on solid mathematical ground we make use of so called L-values. They are defined as

$$L(u_i) = \log\left(\frac{P(u_i = 0)}{P(u_i = 1)}\right). \tag{2.15}$$

This formula converts two related probabilities into a number that can vary from $-\infty$ to $+\infty$, where $-\infty$ means u_i is certainly 1 and $+\infty$ means u_i is certainly 0. The larger the difference between the probabilities, the greater the magnitude of the L-value and therefore the more reliable the decision bout u_i. An L-value of 0 means both possibilities are equally likely. So we could as well switch a coin to make a decision.
To provide reliabilities becomes important when the receiver is subdivided into different stages: equalizing and decoding. To pass only "hard decisions" from one stage to another results in a loss of performance because valuable "soft information", i.e. the reliability information, is neglected. This information can be included into the decision making process and improve the performance.

2.8.2 The Optimal Receiver

An optimal s/s-APP receiver for the transmission line depicted earlier in Figure 2.3 that results in the minimal probability of bit error has to calculate the s/s-APP L-value

$$L(u_i|\boldsymbol{y}) = \log\left(\frac{P(u_i = 0|\boldsymbol{y})}{P(u_i = 1|\boldsymbol{y})}\right) \tag{2.16}$$

for every information bit u_i. A hard decision according to Equation (2.14) is then simply made by evaluating the sign of the L-value, i.e. $\hat{u}_i = 0$ if $L(u_i|\boldsymbol{y}) > 0$ and $\hat{u}_i = 1$ if $L(u_i|\boldsymbol{y}) < 0$. The reliability of the decision is provided by the magnitude of the L-value.
The question is whether it is possible to evaluate Equation (2.16). The answer is yes, and only two steps are involved. The first step is to express $P(u_i|\boldsymbol{y})$ with the help of $P(\boldsymbol{u}|\boldsymbol{y})$ by summing up over all information vectors actually having a 0 or 1 as their ith element. Then Bayes' theorem for conditional probabilities can be applied and we obtain

$$L(u_i|\boldsymbol{y}) = \log\left(\frac{\sum\limits_{\boldsymbol{u}:u_i=0} P(\boldsymbol{u}|\boldsymbol{y})}{\sum\limits_{\boldsymbol{u}:u_i=1} P(\boldsymbol{u}|\boldsymbol{y})}\right) = \log\left(\frac{\sum\limits_{\boldsymbol{u}:u_i=0} P(\boldsymbol{y}|\boldsymbol{u})P(\boldsymbol{u})}{\sum\limits_{\boldsymbol{u}:u_i=1} P(\boldsymbol{y}|\boldsymbol{u})P(\boldsymbol{u})}\right). \tag{2.17}$$

$P(\boldsymbol{u})$ can be canceled out because every information bit is assumed to be equally likely 0 or 1. $P(\boldsymbol{y}|\boldsymbol{u})$ is then evaluated by

1. performing the mapping $\boldsymbol{u} \rightarrow \boldsymbol{c} \rightarrow \tilde{\boldsymbol{c}} \rightarrow \boldsymbol{x} \rightarrow \tilde{\boldsymbol{x}}$ and
2. calculating the probability density with the help of the Gaussian distribution (2.13):

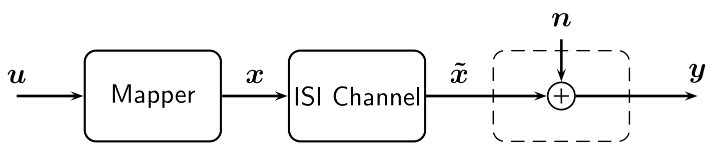
$$P(\boldsymbol{y}|\boldsymbol{u}) = \prod_{j=1}^{n} p(y_j|\tilde{x}_j). \tag{2.18}$$

Nonetheless Equation (2.17) is as simple as it is impractical. Even though it *can* be evaluated a summation over *all* information vectors $\boldsymbol{u}$ has to be done. And for *every* $\boldsymbol{u}$ we have to put ourselves in the position of the encoder, reconstruct the encoding process up to $\tilde{\boldsymbol{x}}$ and then calculate the probability for this event based on what was received. Thus the complexity is proportional to 2^k which is not acceptable even for small k.

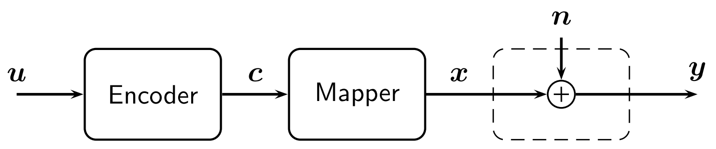
2.8.3 Separate Equalization and Decoding

Whenever a problem seems too complex to solve as a whole it is a good idea to break it down to smaller problems, solve them instead and then try to combine the solutions.
Smaller problems in our case would be to separately solve the equalization and decoding tasks. First an *equalizer* estimates the channel symbols x_i by evaluating $\boldsymbol{y}$ with the s/s-APP rule:

$$L(x_i|\boldsymbol{y}) = \log\left(\frac{P(x_i = +1|\boldsymbol{y})}{P(x_i = -1|\boldsymbol{y})}\right). \tag{2.19}$$



(a) for equalization.



(b) for decoding.

Figure 2.13: Simplified transmitters for deriving the BCJR algorithm

Knowing that x_i and c_i are related through a mapper and an interleaver, an s/s-APP *decoder* could then easily be provided with likelihoods about the code bits $\boldsymbol{L} = (L(c_1|\boldsymbol{y}), \dots, L(c_n|\boldsymbol{y}))$. This decoder could use $\boldsymbol{L}$ in order to perform another s/s-APP calculation by evaluating

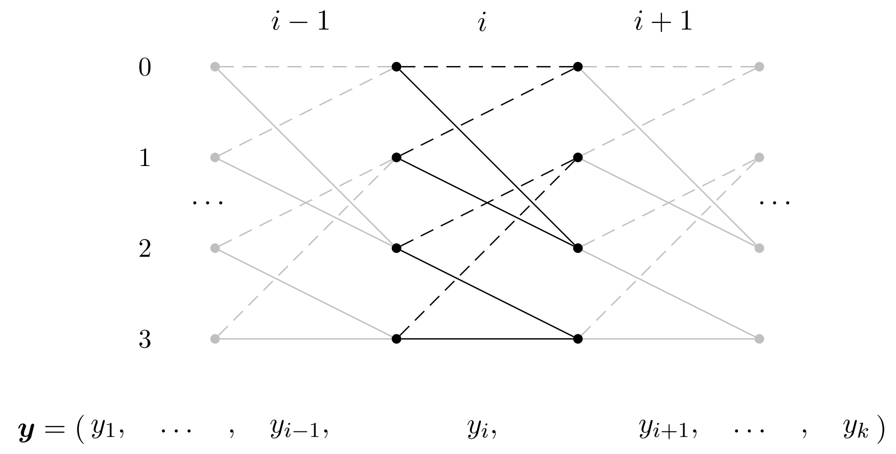
$$L(u_i|\boldsymbol{L}) = \log\left(\frac{P(u_i = +1|\boldsymbol{L})}{P(u_i = -1|\boldsymbol{L})}\right). \tag{2.20}$$

This would all be senseless if there was not any efficient solution for Equations (2.19) and (2.20) because still, the only approach for solving these equations so far would be the same as for the optimal receiver resulting in an unacceptable complexity. Fortunately s/s-APP calculations can be efficiently solved with the help of trellis-based approaches using the so called BCJR algorithm [1].

In order to derive the BCJR algorithm the problem of joint equalizing and decoding is broken down into two simple (and almost similar) tasks. For that purpose have a look at Figure 2.13. In (a) a simplified transmitter is depicted where the encoder is missing and in (b) encoded bits are transmitted over an AWGN channel where no intersymbol interference takes place. In both cases an optimal s/s-APP receiver has to compute

$$L(u_i|\boldsymbol{y}) = \log\left(\frac{P(u_i = 0|\boldsymbol{y})}{P(u_i = 1|\boldsymbol{y})}\right) \tag{2.21}$$

but now considering only one encoding module (either the encoder or the ISI channel) according to Figure 2.13.
The BCJR algorithm will be derived in detail for the equalization task and then only the differences for the decoding task will be pointed out.



$i-1$
i
$i+1$
0
1
2
3
$\cdots$
$\cdots$
$\boldsymbol{y} = (y_1, \; \cdots \;, \; y_{i-1}, \; y_i, \; y_{i+1}, \; \cdots \;, \; y_k)$

Figure 2.14: Excerpt of the trellis for channel $\boldsymbol{h1}$ at time i.

Trellis based s/s-APP Equalization

In Figure 2.14 an excerpt of a trellis for channel $\boldsymbol{h1}$ at time i is depicted. To calculate $P(u_i = 0|\boldsymbol{y}) = P(x_i = +1|\boldsymbol{y})$ the first approach would be analogous to the approach of the optimal receiver and to sum up over all $\boldsymbol{x}$ that have $+1$ at the ith place. This way too many $\boldsymbol{x}$ have to be considered. But by looking at the trellis this probability can be rewritten by summing up over all edges $\epsilon \in \mathcal{E}_i$ where $\lambda(\epsilon) = +1$ (i.e. all black, dashed lines in Figure 2.14) and for every ϵ calculate the probability $P(s_i, s_{i+1}|\boldsymbol{y})$. That is given $\boldsymbol{y}$ how likely it is that the trellis at time i changes from state s_i to state s_{i+1}. This leads to

$$\begin{aligned} L(u_i|\boldsymbol{y}) &= \log\left(\frac{P(u_i = 0|\boldsymbol{y})}{P(u_i = 1|\boldsymbol{y})}\right) = \log\left(\frac{P(x_i = +1|\boldsymbol{y})}{P(x_i = -1|\boldsymbol{y})}\right) \\ &= \log\left(\frac{\sum\limits_{\epsilon \in \mathcal{E}_i^{(\lambda:+1)}} P(s_i, s_{i+1}|\boldsymbol{y})}{\sum\limits_{\epsilon \in \mathcal{E}_i^{(\lambda:-1)}} P(s_i, s_{i+1}|\boldsymbol{y})}\right) \end{aligned} \tag{2.22}$$

where $\mathcal{E}_i^{(\lambda:\pm 1)}$ is a short notation for $\{\epsilon \in \mathcal{E}_i : \lambda(\epsilon) = \pm 1\}$. Let us assume for a moment it is possible to calculate $P(s_i, s_{i+1}|\boldsymbol{y})$ for all edges $\epsilon \in \mathcal{E}_i$. Then by using the trellis approach the number of summations in nominator and denominator reduce from 2^{k-1} to only $2^L = 4$ for the case of channel $\boldsymbol{h1}$.
It is now important to find an efficient way to calculate $P(s_i, s_{i+1}|\boldsymbol{y})$ for a given edge ϵ. First Bayes' theorem is applied and $P(\boldsymbol{y})$ is canceled out resulting in

$$L(u_i|\boldsymbol{y}) = \log\left(\frac{\sum\limits_{\epsilon \in \mathcal{E}_i^{(\lambda:+1)}} P(s_i, s_{i+1}, \boldsymbol{y})}{\sum\limits_{\epsilon \in \mathcal{E}_i^{(\lambda:-1)}} P(s_i, s_{i+1}, \boldsymbol{y})}\right). \tag{2.23}$$

Then $\boldsymbol{y}$ is rewritten with a head and tail part $\boldsymbol{y} = (\boldsymbol{y}_h, y_i, \boldsymbol{y}_t) = ((y_1, \ldots, y_{i-1}), y_i, (y_{i+1}, \ldots, y_n))$ and the joint probability is decomposed to

$$\begin{aligned} P(s_i, s_{i+1}, \boldsymbol{y}) &= P(s_i, s_{i+1}, \boldsymbol{y}_t, y_i, \boldsymbol{y}_h) \\ &= \underbrace{P(s_i, \boldsymbol{y}_h)}_{\alpha(A(\epsilon))} \underbrace{P(s_{i+1}, y_i | s_i)}_{\gamma(\epsilon)} \underbrace{P(\boldsymbol{y}_t | s_{i+1})}_{\beta(\Omega(\epsilon))}. \end{aligned} \tag{2.24}$$

This decomposition allows us to split the probability that the transmitted sequence included ϵ in the trellis path under the assumption of observing $\boldsymbol{y}$ into three parts:

1. $\alpha(A(\epsilon))$: The probability of observing the starting state s_i and the symbols $\boldsymbol{y}_h$.
2. $\gamma(\epsilon)$: The probability of observing both the state transition to s_{i+1} and y_i at time i when assuming the starting state s_i as given.
3. $\beta(\Omega(\epsilon))$: The probability of observing $\boldsymbol{y}_t$ when assuming the state transition to s_{i+1} takes place at time i.

It is possible to calculate α and β by recursively evaluating

$$\alpha(\nu) = \sum_{\epsilon \in \mathcal{E}_{\text{in}}(\nu)} \gamma(\epsilon)\alpha(A(\epsilon)) \qquad \forall \nu \in \mathcal{V}_i \quad (i = 2, \ldots, k), \tag{2.25}$$

$$\beta(\nu) = \sum_{\epsilon \in \mathcal{E}_{\text{out}}(\nu)} \gamma(\epsilon)\beta(A(\epsilon)) \qquad \forall \nu \in \mathcal{V}_i \quad (i = k-1, \ldots, 2) \tag{2.26}$$

and initializing $\alpha(\nu) = 1$ for $\nu \in \mathcal{V}_1$ and $\beta(\nu) = 1$ for $\nu \in \mathcal{V}_{k+1}$. The initialization reflects the convention to let every trellis start and end in the all-zero state respectively, so obviously the probability of observing those states is 1.

A further decomposition of γ results in

$$\gamma(\epsilon) = P(s_{i+1}, y_i | s_i) \tag{2.27}$$

$$= \underbrace{P(s_{i+1} | s_i)}_{\text{a priori}} P(y_i | s_i, s_{i+1}) \tag{2.28}$$

The first a priori probability does not depend on any observation. When we look at the trellis and pretend to be in any state at time i, the question of which edge to chose to go to the next state is only determined by the distribution of the input symbol x_i or $P(s_{i+1}|s_i) = P(x_i = \lambda(\epsilon))$ for a given ϵ. This probability is fixed *before* transmitting the data over the channel, thus it is named a priori. The second probability only depends on the observation y_i or, in other words, how the AWGN channel distorts a transmitted symbol $\tilde{x}_i$. Because when we assume the state transition $s_i \to s_{i+1}$, then the output of the ISI channel $\mu(\epsilon)$ is determined. Thus when assuming $\mu(\epsilon)$ of being transmitted over the AWGN channel, the probability density of receiving y_i is $p(y_i|\mu(\epsilon))$ and can be calculated with the Gaussian distribution (2.13). In summary γ can be calculated by

$$\gamma(\epsilon) = P(x_i = \lambda(\epsilon)) \cdot p(y_i|\mu(\epsilon)). \tag{2.29}$$

It is now possible to formulate the BCJR algorithm for the problem of equalization.

BCJR algorithm: Given the full trellis graph and the observation $\boldsymbol{y}$ the s/s-APP L-values $L(u_i|\boldsymbol{y})$ are calculated by

- assigning every edge $\epsilon \in \mathcal{E}$ a metric $\gamma(\epsilon)$ according to (2.29),
- calculating all $\alpha(\nu)$ with the forward recursion in (2.25),
- calculating all $\beta(\nu)$ with the backward recursion in (2.26) and finally
- calculating

$$L(u_i|\boldsymbol{y}) = \log\left(\frac{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\lambda:+1)}} \alpha(A(\epsilon))\cdot\gamma(\epsilon)\cdot\beta(\Omega(\epsilon))}{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\lambda:-1)}} \alpha(A(\epsilon))\cdot\gamma(\epsilon)\cdot\beta(\Omega(\epsilon))}\right). \tag{2.30}$$

The complexity of this algorithm of course depends on k but it does not grow exponentially when increasing k. The prevailing complexity factor is not the “length” of the trellis but its “height” or the number of states in the trellis. As the number of states in the trellis grows exponentially with the number of delay elements in the ISI channel this algorithm is feasible for “short” channels only.
Inserting Equation (2.29) into (2.30) leads to (note that L-values for u_i and for x_i are the same for the simplified transmitters in Figure 2.13)

$$L(x_i|\boldsymbol{y}) = \log\left(\frac{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\lambda:+1)}} \alpha(A(\epsilon))\cdot P(x_i=\lambda(\epsilon))\cdot p(y_i|\mu(\epsilon))\cdot\beta(\Omega(\epsilon))}{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\lambda:-1)}} \alpha(A(\epsilon))\cdot P(x_i=\lambda(\epsilon))\cdot p(y_i|\mu(\epsilon))\cdot\beta(\Omega(\epsilon))}\right). \tag{2.31}$$

The factor $P(x_i = \lambda(\epsilon))$ will be the same in every summand for the nominator ($P(x_i = +1)$) and denominator ($P(x_i = -1)$). Thus it is possible to split the L-value into

$$\begin{aligned} L(x_i|\boldsymbol{y}) &= \log\left(\frac{P(x_i=+1)}{P(x_i=-1)}\right) + \log\left(\frac{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\lambda:+1)}} \alpha(A(\epsilon))\cdot p(y_i|\mu(\epsilon))\cdot\beta(\Omega(\epsilon))}{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\lambda:-1)}} \alpha(A(\epsilon))\cdot p(y_i|\mu(\epsilon))\cdot\beta(\Omega(\epsilon))}\right) \\ &= L_{\text{apr}}(x_i) + L_{\text{ext}}(x_i|\boldsymbol{y}). \end{aligned} \tag{2.32}$$

The s/s-APP L-value consists of two parts:

1. The a priori L-value L_{apr} or “what is known about x_i prior to the equalization process”.
2. The extrinsic L-value L_{ext} or “what the equalizer adds to the a priori knowledge about x_i during the equalization process”.

The expressions in quotation marks might seem a little sloppy but the new L-values deserve a descriptive interpretation because they address to a concept that will play a dominant role

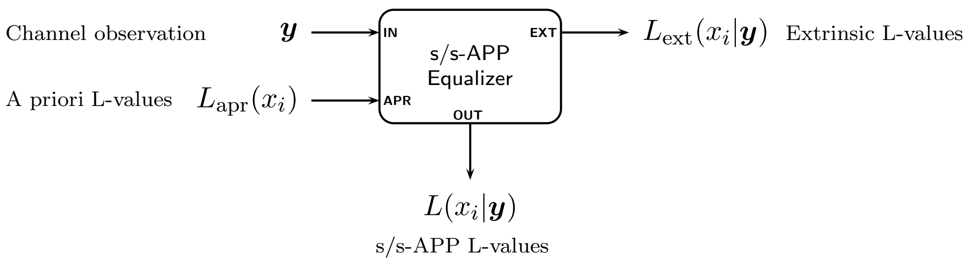


Figure 2.15: Block diagram for the equalizer module. The equalizer takes the corrupted channel symbols and a priori information about the channel symbols as its input and generates s/s-APP and extrinsic L-values as its output.

when introducing the turbo principle in the next chapter. Note that for a transmission line where $L_{\text{apr}}(x_i) = 0$ for all x_i, i.e. the channel symbols are equally likely 0 and 1, the s/s-APP L-value $L(x_i|\boldsymbol{y})$ will be the same as the extrinsic L-value $L_{\text{ext}}(x_i|\boldsymbol{y})$.
In Figure 2.15 a module which implements the BCJR algorithm for optimal s/s-APP equalization of an ISI channel is shown.
The module expects the channel observation $\boldsymbol{y}$ and the a priori L-values about the channel symbols $\boldsymbol{x}$ as its input. In order to convert the a priori L-values to probabilities that are needed for calculating γ in (2.29) some algebra can be applied to the definition of an L-value resulting in

$$P(x_i = \lambda(\epsilon)) = \frac{e^{-\lambda(\epsilon)L_{\text{apr}}(x_i)}}{1 + e^{-L_{\text{apr}}(x_i)}}. \tag{2.33}$$

Trellis based s/s-APP Decoding

The necessary steps for optimal symbolwise decoding when a transmitter according to Figure 2.13 (b) is assumed are similar to the steps in the last section. The s/s-APP L-value $L(u_i|\boldsymbol{y})$ is simplified similarly by exploiting the trellis representation of the convolutional encoder. The decomposition

$$P(s_i, s_{i+1}, \boldsymbol{y}) = \underbrace{P(s_i, \boldsymbol{y}_h)}_{\alpha(A(\epsilon))} \underbrace{\underbrace{P(s_{i+1}|s_i)}_{\text{a priori}} P(\boldsymbol{y}_i|s_i, s_{i+1})}_{\gamma(\epsilon)} \underbrace{P(\boldsymbol{y}_t|s_{i+1})}_{\beta(\Omega(\epsilon))} \tag{2.34}$$

is the same as above, with one slight difference. For an encoder of rate $R = 1/2$ a state transition $s_i \rightarrow s_{i+1}$ generates two output bits. Therefore for one state transition the likelihood of observing two distorted channel symbols $\boldsymbol{y}_i = (y_i^{(1)} y_i^{(2)})$ has to be calculated (the notation is in analogy to $\boldsymbol{c}_i$ in section 2.3). Because of the AWGN channel in Figure 2.13 (b) the channel symbols are distorted by independent noise samples and therefore the metric γ can be written as

$$\begin{aligned} \gamma(\epsilon) &= P(u_i = \lambda(\epsilon)) \cdot P(y_i^{(1)}|c_i^{(1)} = \mu_1(\epsilon)) \cdot P(y_i^{(2)}|c_i^{(2)} = \mu_2(\epsilon)) \\ &= P(u_i = \lambda(\epsilon)) \cdot p(y_i^{(1)}|(-1)^{\mu_1(\epsilon)}) \cdot p(y_i^{(2)}|(-1)^{\mu_2(\epsilon)}). \end{aligned} \tag{2.35}$$

Recall that $\mu_1(\epsilon)$ and $\mu_2(\epsilon)$ refer to the first and second output *bit* of the encoder when ϵ and thus a specific state transition $s_i \rightarrow s_{i+1}$ is given. Therefore an implicit BPSK mapping is necessary when declaring the probability densities which are again calculated using the Gaussian distribution (2.13).
The BCJR algorithm can now be formulated in the same manner as before with the only difference that for decoding the metric in (2.35) is used. The resulting s/s-APP L-value can be written as

$$L(u_i|\boldsymbol{y}) = \log\left(\frac{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\lambda:0)}} \alpha(A(\epsilon))\cdot\gamma(\epsilon)\cdot\beta(\Omega(\epsilon))}{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\lambda:1)}} \alpha(A(\epsilon))\cdot\gamma(\epsilon)\cdot\beta(\Omega(\epsilon))}\right). \tag{2.36}$$

For a decoder is it also possible to calculate s/s-APP L-values for the *code* bits. Therefore the above equation is slightly modified:

$$L(c_i^{(1)}|\boldsymbol{y}) = \log\left(\frac{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\mu_1:0)}} \alpha(A(\epsilon))\cdot\gamma(\epsilon)\cdot\beta(\Omega(\epsilon))}{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\mu_1:1)}} \alpha(A(\epsilon))\cdot\gamma(\epsilon)\cdot\beta(\Omega(\epsilon))}\right), \tag{2.37}$$

$$L(c_i^{(2)}|\boldsymbol{y}) = \log\left(\frac{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\mu_2:0)}} \alpha(A(\epsilon))\cdot\gamma(\epsilon)\cdot\beta(\Omega(\epsilon))}{\sum\limits_{\epsilon\in\mathcal{E}_i^{(\mu_2:1)}} \alpha(A(\epsilon))\cdot\gamma(\epsilon)\cdot\beta(\Omega(\epsilon))}\right). \tag{2.38}$$

Before the above equations are split into a priori and extrinsic part in the same manner as for equalization, it is helpful to define the channel L-value

$$L_{\text{ch}}(y_i) = \log\left(\frac{P(y_i|c_i=0)}{P(y_i|c_i=1)}\right). \tag{2.39}$$

This channel L-value expresses a likelihood about the code bits resulting solely from the corresponding distorted channel symbol y_i. Equations (2.36), (2.37) and (2.38) can then be split for

$$\begin{array}{llrcccccl}
u_i, & \text{NSC encoder:} & L(u_i|\boldsymbol{y}) & = & L_{\text{apr}}(u_i) & + & & & L_{\text{ext}}(u_i|\boldsymbol{y}), & (2.40)\\
u_i, & \text{RSC encoder:} & L(u_i|\boldsymbol{y}) & = & L_{\text{apr}}(u_i) & + & L_{\text{ch}}(y_i^{(1)}) & + & L_{\text{ext}}(u_i|\boldsymbol{y}), & (2.41)\\
\boldsymbol{c}_i, & \text{first output:} & L(c_i^{(1)}|\boldsymbol{y}) & = & & & L_{\text{ch}}(y_i^{(1)}) & + & L_{\text{ext}}(c^{(1)}|\boldsymbol{y}), & (2.42)\\
\boldsymbol{c}_i, & \text{second output:} & L(c_i^{(2)}|\boldsymbol{y}) & = & & & L_{\text{ch}}(y_i^{(2)}) & + & L_{\text{ext}}(c^{(2)}|\boldsymbol{y}). & (2.43)
\end{array}$$

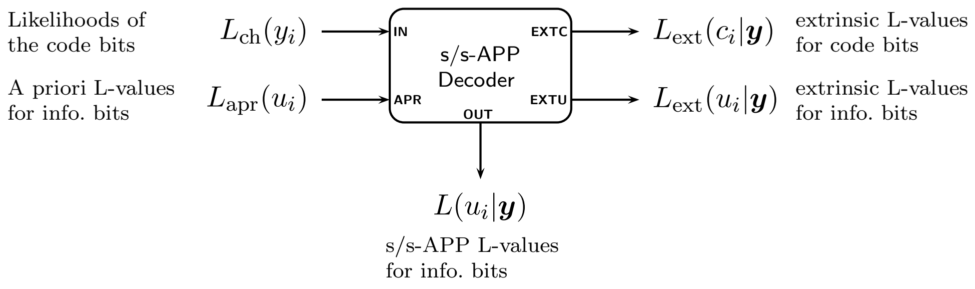

Likelihoods of the code bits
$L_{\text{ch}}(y_i)$
A priori L-values for info. bits
$L_{\text{apr}}(u_i)$
IN
APR
s/s-APP Decoder
EXTC
EXTU
OUT
$L_{\text{ext}}(c_i|\boldsymbol{y})$
extrinsic L-values for code bits
$L_{\text{ext}}(u_i|\boldsymbol{y})$
extrinsic L-values for info. bits
$L(u_i|\boldsymbol{y})$
s/s-APP L-values for info. bits

Figure 2.16: Block diagram for the decoding module. The module requires likelihoods of the code bits and not actual observation $\boldsymbol{y}$ as its input.

It is possible to separate the channel L-value from the s/s-APP L-value for the code bits. For the information bits this further separation is only possible when a systematic encoder is used. This separation of channel information and a priori and extrinsic information respectively is not possible for the equalizer because an ISI channel always represents a non systematic encoder. In figure 2.16 a block diagram of the decoding module is shown.
Note that this module expects the observation to be passed as channel L-values $L_{\text{ch}}(y_i)$ or likelihoods about the code bits not as actual observation y_i. For an AWGN channel those likelihoods can be calculated by evaluating

$$L_{\text{ch}}(y_i) = \log\left(\frac{P(y_i|c_i=0)}{P(y_i|c_i=1)}\right) = \log\left(\frac{p(y_i|+1)}{p(y_i|-1)}\right) = \log\left(\frac{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i-1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i+1)^2}{2\sigma^2}\right)}\right) = \frac{2y_i}{\sigma^2}. \tag{2.44}$$

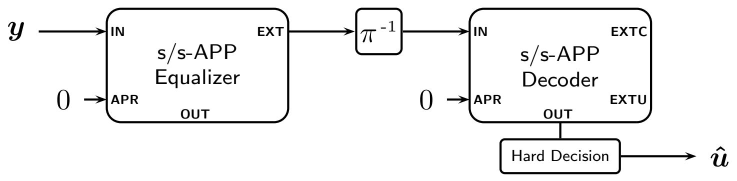


Figure 2.17: Classical receiver design employing successive equalization and decoding. Note that passing extrinsic L-values (EXT) and s/s-APP L-values (OUT) from equalizer to decoder is the same when no a priori information is available.

Combining Equalization and Decoding

The s/s-APP modules derived in the last two sections can now be used to design a receiver for the transmission line. In Figure 2.17 the classical approach of separate equalization and decoding that was already described roughly at the beginning of this section is shown. The first module equalizes the ISI channel and calculates s/s-APP L-values of the channel symbols. These L-values are then used as input for the decoder (after deinterleaving).
Two implicit assumptions are made. First it is implied that $\boldsymbol{x}$ is generated by an ideal source, i.e. the equalizer assumes the a priori probability of x_i to be equally likely 1 or 0 and identical for all i. This is not necessarily true because $\boldsymbol{x}$ is the output of an encoder and not of an ideal source. Second the decoder assumes the input likelihoods about the code bits to be independent of each other (like they would be when resulting from an AWGN channel). But the s/s-APP L-values for successive channel symbols calculated by the equalizer are correlated as shown in Figure 2.18. The implemented deinterleaver shuffles these L-values before passing them to the decoder thus lowering the correlation between adjacent L-values.

We tested the receiver design for a block length of $k = 10000$ with the $(37, 21)$ RSC encoder and ISI channel $\boldsymbol{h2}$. Figure 2.19 shows the bit error rate over the signal-to-noise ratio. Results are obtained by Monte-Carlo simulation. The interleaver improves the performance to a large extent for high SNR due to the reason stated above. It also can be seen that for a bit error rate of 10^{-5} performance is about 6.4 dB away from the limit.

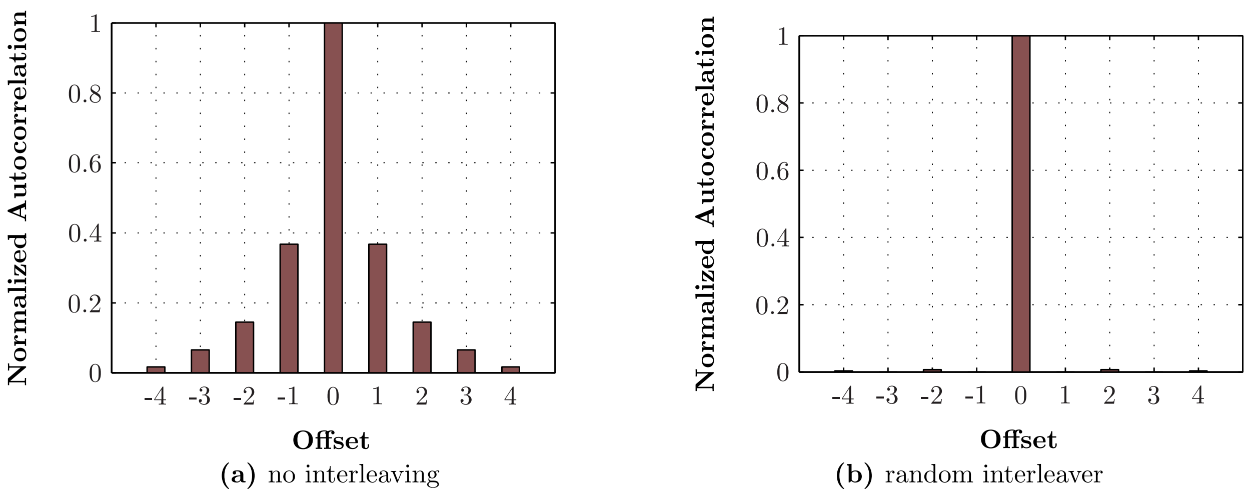


Figure 2.18: Autocorrelation for an example sequence of s/s-APP L-values $(L(x_1|\boldsymbol{y}), \ldots, L(x_n|\boldsymbol{y}))$ obtained with parameters $k = 10000$, channel $\boldsymbol{h2}$ and $E_b/N_0 = 6$ dB.

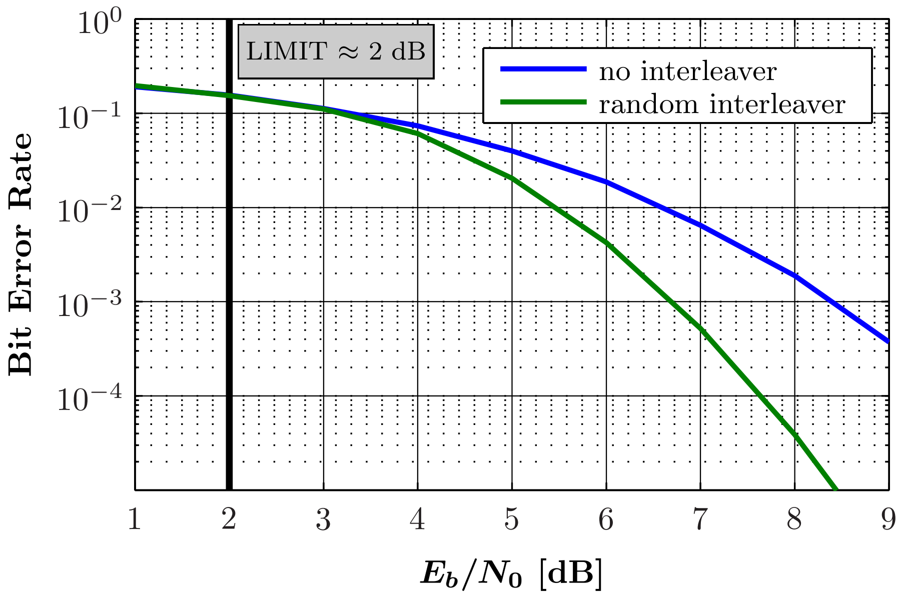


Figure 2.19: Simulation results for separate equalization and decoding ($k = 10000$, ISI channel $\boldsymbol{h2}$ and (37,21) RSC encoder).

Chapter 3

Iterative Equalization

3.1 The Turbo Principle

The classical receiver which was presented at the end of the last chapter performs successive equalization and decoding and has two major problems:

1. The output L-values of the equalizer are correlated.
2. No a priori information about $\boldsymbol{x}$ is available for the equalization process.

The impact of the correlation can be reduced with the help of a random interleaver which shuffles the correlated likelihoods of the channel symbols before passing them to the decoder. The turbo principle addresses the second problem.

Turbo principle: Perform iterative s/s-APP estimations about symbols x_i with successively refined a priori L-values $L^j_{\text{apr}}(x_i)$. For the calculation of $L^j_{\text{apr}}(x_i)$ use all the preferably statistically independent information which is available at iteration $(j-1)$.

This definition is taken from [15] and slightly adjusted to the notation used in this thesis. The turbo principle is very general and can be applied to a variety of problems [15]. In the next section it is described how this principle can be applied to the problem of equalization and decoding.

3.2 Turbo Equalization

We define the successive process of equalization and decoding to be one iteration for the purpose of the turbo principle. Initially during the first iteration $(j = 1)$ the equalizer has to estimate the channel symbols with a priori L-values $L^1_{\text{apr}}(x_i) = 0$ for all x_i simply because there is no a priori information available. This leads to a suboptimal estimation due to the fact that $\boldsymbol{x}$ results from encoding $\boldsymbol{u}$ and not from an ideal source. According to the turbo principle a new s/s-APP estimation should now be performed with refined a priori L-values $L^2_{\text{apr}}(x_i)$. A refined version of the a priori L-values is obtained by considering the extrinsic L-values of the decoder about the code bits $\boldsymbol{c}$ as statistically independent information about the channel symbols $\boldsymbol{x}$. Therefore a feedback loop between the decoder and the equalizer is established as shown in Figure 3.1. With these refined a priori L-values a new estimation about the channel

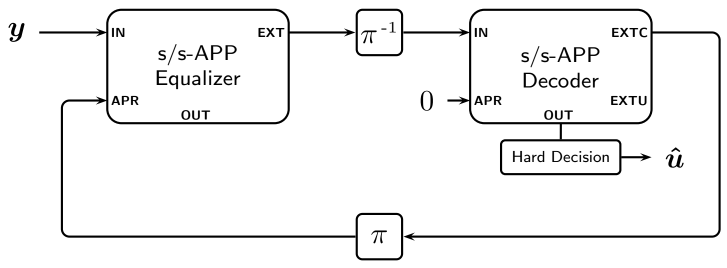


Figure 3.1: Iterative receiver employing turbo equalization. The equalizer performs iterative s/s-APP estimations about the channel symbols. Therefore the extrinsic L-values from the decoder about the code bits are used as a priori L-values in the next iteration.

symbols is done by the equalizer. Extrinsic L-values are passed to the decoder providing even more refined a priori L-values and so on. This iterative process is finally stopped after a defined number of iterations or by using a stop criterion. A very simple but effective method to stop the iterative process is to evaluate the hard decisions about the information bits in each iteration and stop if no change occurs for two successive iterations [16].

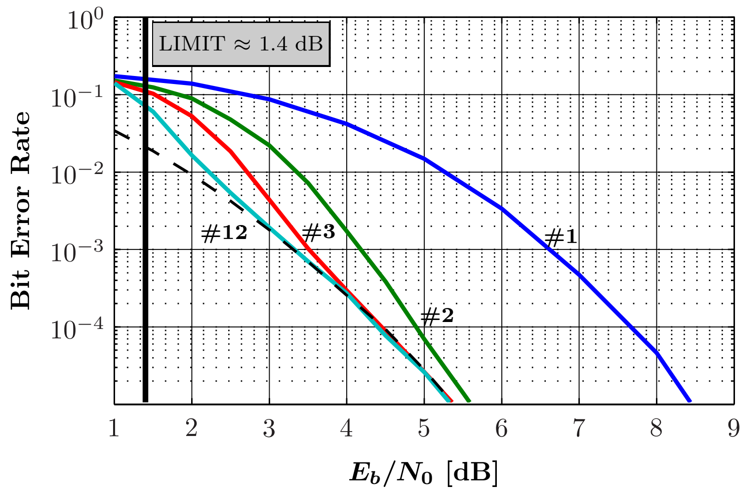
Note that only extrinsic L-values are passed to the next module. This applies to both the equalizer and the decoder. To verify this we will describe the "flow" of L-values during the iterative process and see what would happen if s/s-APP L-values instead of extrinsic L-values are passed to the next module.
During the first iteration the equalizer takes $\boldsymbol{y}$ and decides about the channel symbols in form of L-values $L(x_i|\boldsymbol{y})$ which are passed to the decoder. The decoder assumes these L-values to be uncorrelated likelihoods about the code bits and treats them as channel L-values L_{ch}. The decoder performs s/s-APP decoding solely based on those L-values. The resulting s/s-APP L-values about the code bits contain the input channel L-values L_{ch} as an independent portion of information (see Equations (2.42) and (2.43)). If s/s-APP L-values are now fed back as a priori information back to the equalizer the decoder would simply reinform the equalizer about something that it already knows. Positive feedback would be created. In [16] it is investigated that this positive feedback has a negative impact on the convergence of the turbo equalization process. The problem is that once the L-values passed from one module to the other become too large, i.e one module is too sure about its decision, further iterations yield no improvements in terms of bit error rate. When positive feedback is created *the same* information about symbols is used several times and treated as *independent* information. Thus during the first few iterations the modules would get too optimistic about their decisions and no further improvements can be obtained. For the same reason only extrinsic L-values are passed from the equalizer to the decoder. The s/s-APP L-values of the equalizer contain the a priori L-values from the decoder as independent information about the channel symbols (see Equation (2.32)) thus feeding them back again to the decoder would create positive feedback. The key is how to define which information can be used to calculate refined a priori L-values for the next iteration without creating positive feedback. This is a crucial factor in turbo

equalization as well as in any other situation where the turbo principle is applied.

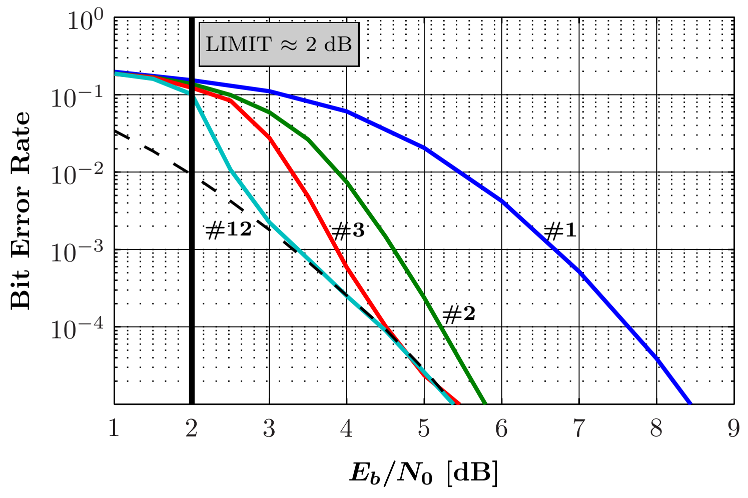
3.2.1 Simulation

We tested the iterative receiver in Figure 3.1 by simulating the bit error rate for different signal to noise ratios with ISI channels $\boldsymbol{h1}$, $\boldsymbol{h2}$ and $\boldsymbol{h3}$. A random interleaver, $k = 10000$ and the (37,21) RSC encoder were used. Perfect channel knowledge is assumed. The results can be seen in Figure 3.2. The graphs are ordered from top to bottom according to the theoretical limits of the channels. The "best" ISI channel is displayed at the top and the "worst" ISI channel at the bottom. ISI channel $\boldsymbol{h3}$ is also referred to as a worst case scenario for a time-invariant intersymbol interference channel with five taps [7].
The number of performed iterations is indicated by a number next to the corresponding curve. Note that when only one iteration is performed the receiver structure is equivalent to the conventional approach of successive equalization and decoding.

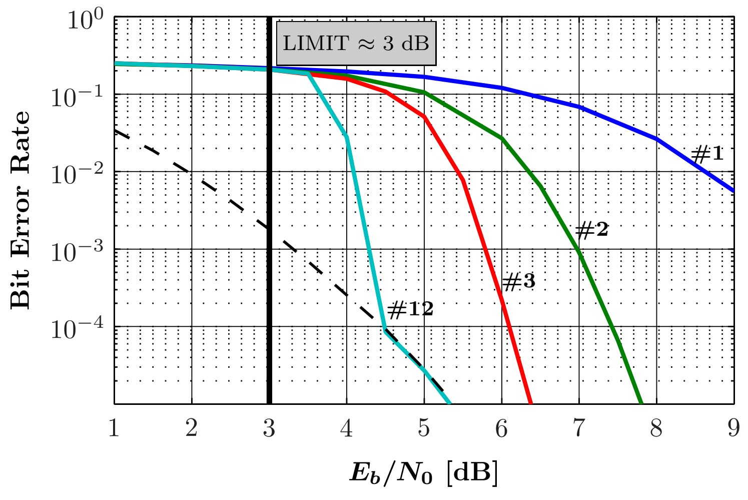
The first thing to observe is that the performance significantly improves for all channels when using the turbo equalization approach. During the second iteration the performance gain is largest and decreases with each further iteration. One can also see that most performance gain is achieved for channel $\boldsymbol{h3}$ which has the most severe frequency distortions.
The bit error rate seems to be lower bound by the performance of the used convolutional encoder on an AWGN channel without ISI. Thus at about 5.3 dB a bit error rate of 10^{-5} is reached after performing 12 iterations independent of the ISI channel. In other words the iterative equalization approach allows to completely overcome the negative effects of the tested intersymbol interference channels after 12 iterations.
Nonetheless the fact that the performance does not exceed this lower bound has an impact on the evaluation of this scheme in absolute terms when comparing the results to the information theoretical limits. For channel $\boldsymbol{h3}$ the gap to the limit is about 2.3 dB at a bit error rate of 10^{-5}. This gap increases for the other channels because their theoretical limits are lower. So for channel $\boldsymbol{h1}$ the gap between the limit and achievable signal to noise ratio for a bit error rate of 10^{-5} is still about 3.9 dB.



(a) ISI channel $\boldsymbol{h1}$



(b) ISI channel $\boldsymbol{h2}$



(c) ISI channel $\boldsymbol{h3}$

Figure 3.2: Simulation results for turbo equalization as a function of the performed iterations. The dashed line shows the performance of the (37,21) RSC encoder when transmitting over an AWGN channel without ISI.

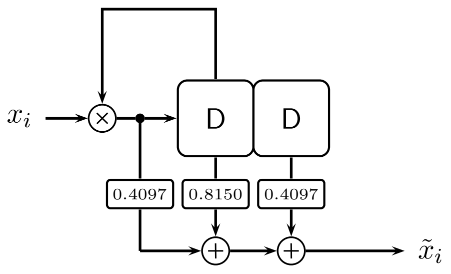


Figure 3.3: ISI channel $\boldsymbol{h1}$ having a recursive structure. Note the resemblance to an RSC encoder.

3.3 Precoding

Iterative equalization can provide great performance improvements over the conventional approach up to a point where the bit error rate performance reaches that of a convolutional encoder when transmitting over an AWGN channel without ISI. It seems as though the overall error-correcting capability of the scheme is determined by that of the outer convolutional encoder and that the bit error rate can not be lower for an ISI channel than for an AWGN channel without ISI for the same signal to noise ratio using identical error-correcting encoders.

However the convolutional encoder and the ISI channel can be regarded as a serially concatenated encoder where symbols are transmitted over an AWGN channel without ISI. This is only a different point of view on the exact same transmission line. In [17] serially concatenated encoders that consist of two convolutional encoders are studied and design rules for these encoders are presented. It is derived that " [...] the inner encoder must be a convolutional recursive encoder". Basically precoding addresses to this design rule and tries to make the ISI channel, i.e. the inner encoder of the serially concatenated encoder, recursive. An example of an ISI channel having a recursive structure is depicted in Figure 3.3.
Unfortunately the ISI channel itself cannot be modified because it represents the underlying real-life communication channel. Conceptually modifications can only be done in the transmitter. Therefore a precoder is introduced in the transmission line between mapper and the ISI channel. Thus the channel symbols are modified before being transmitted as shown in Figure 3.4.
The recursive ISI channel has great resemblance to an RSC encoder and it is of course possible to design different precoders. For simplicity we are restricting ourselves to the differential precoder shown in Figure 3.4.
It should be noted that the implementation of a precoder does not increase the complexity in the equalizer as long as the memory of the precoder is equal to or smaller than the length of the ISI channel. Because the cascade of precoder and ISI channel can then be regarded as a recursive ISI channel with the same length as shown in Figure 3.3. Thus the states in the trellis and therefore the complexity of the BCJR algorithm will remain the same.

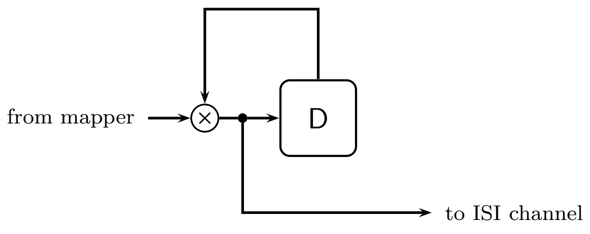
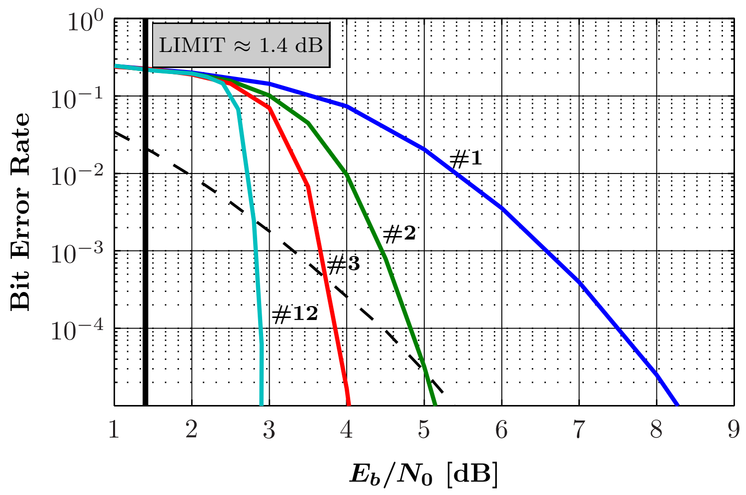


Figure 3.4: Differential precoder which is implemented in the transmitter to make the cascade of precoder and ISI channel – and thus the inner encoder of the serially concatenated code system – recursive.

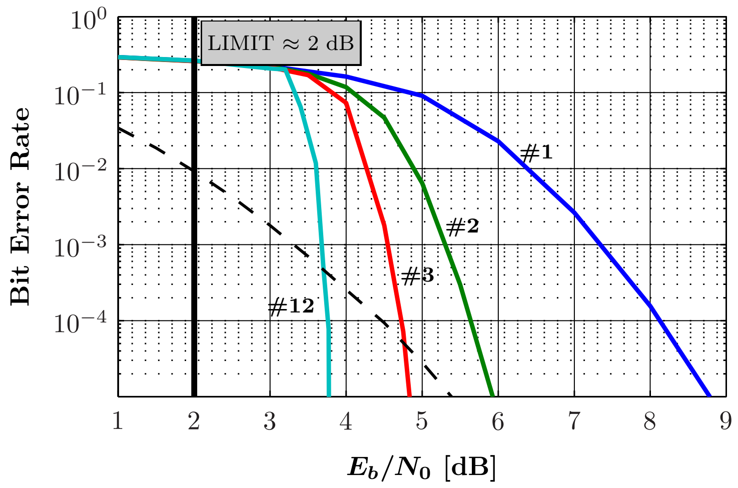
3.3.1 Simulation

The bit error rate of the transmission line including the differential precoder was simulated using turbo equalization. The same parameters as before were used, i.e. the same random interleaver, $k = 10000$ and the (37,21) RSC encoder. Results are shown in Figure 3.5.

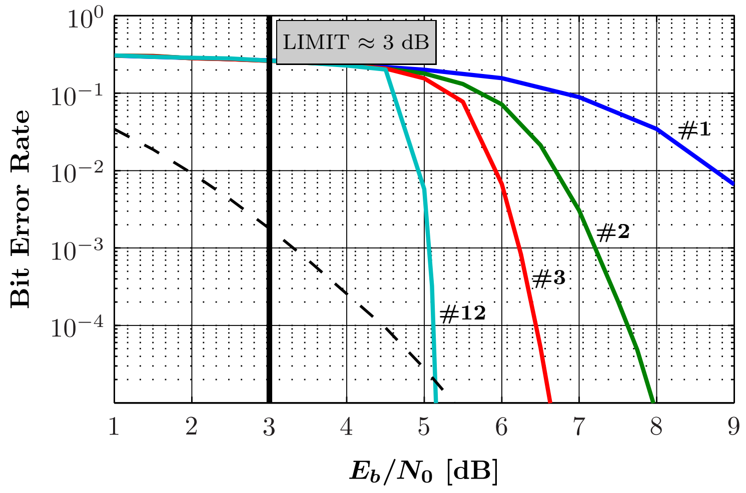
It can be seen that the differential precoder has a great impact on the bit error rate of the transmission line when turbo equalization is used in the receiver.
Compared to the case where no precoder is present the bit error rate does not seem to be lower bound by the performance of the used encoder on an AWGN channel without ISI and the curves do not converge towards the dashed line. Besides that all curves decrease faster once a "breaking point" with a bit error rate of about 10^{-1} is reached. This is very distinctive when 12 iterations are performed in which cases the curves almost deviate vertically.
When comparing the curves for 12 iterations for precoded and not precoded channels (Figure 3.2) it can be seen that for regions below the limit and closely above the limit (approximately within 1 dB) performance significantly decreases when the channel is precoded. Compare for example the two curves for channel $\boldsymbol{h2}$ at 3 dB after 12 iterations. In the case where no precoder is present the bit error rate is at about $2 \cdot 10^{-3}$ and close to the dashed line. However when a precoder is used the bit error rate is at about $2 \cdot 10^{-1}$. Similar observations can be made for the other channels for signal to noise ratios up to about 1 dB above their limits. So for some signal to noise ratios performance decreases when a precoder is used. It should be noted though that the corresponding bit error rates for these signal to noise ratios are usually not of great practical interest.
By using a differential precoder and turbo equalization in the receiver it is possible to approach the capacity limit for a bit error rate of 10^{-5} up to about 1.5 dB for channel $\boldsymbol{h1}$, up to about 1.8 dB for channel $\boldsymbol{h2}$ and up to about 2.1 dB for channel $\boldsymbol{h3}$ after 12 iterations.



(a) Precoded ISI channel $\boldsymbol{h1}$



(b) Precoded ISI channel $\boldsymbol{h2}$



(c) Precoded ISI channel $\boldsymbol{h3}$

Figure 3.5: Simulation results for turbo equalization as a function of the performed iterations. The dashed line shows the performance of the (37,21) RSC encoder when transmitting over an AWGN channel without ISI.

3.4 Discussion

In this chapter the performance of an iterative receiver structure employing turbo equalization was tested for different ISI channels. It was shown that iterative estimations of the channel symbols done with “refined” a priori L-values resulting from the decoder can improve error correcting performance of the overall scheme to a large extent. Performance significantly changed with the implementation of a precoder which appears to be the crucial factor to achieve performance close to the theoretical limits of an ISI channel.

Through a union bound analysis it can be shown that the performance of the iterative receiver is determined by the free distance of the outer convolutional code when no precoder is implemented [3]. Therefore one way to improve the performance in that case is to employ a more potent convolutional outer code having a higher free distance. This can be done by increasing the memory of the encoder, but unfortunately this is accompanied by an exponential increase in decoding complexity which is usually not desirable.
The idea of implementing a precoder however seems to exploit the structure of the transmission line very well. The reason for the design rule derived in [17], i.e. to always use a recursive inner encoder in a serially concatenated code system, relies on an analysis showing that a recursive inner encoder always yields a so called interleaver gain. It is shown in [3] that by introducing a precoder a similar gain can be achieved in a transmission line where an ISI channel is employed as the inner encoder. This so called precoder weight gain can be roughly explained by the fact that a recursive convolutional encoder like a precoder has an infinite unit impulse response. Low weight codewords resulting from the outer encoder are more likely of being transformed into high weight codewords transmitted over the ISI channel. It can also be shown that different precoders have different effects on the performance and the choice of the precoder has to be adjusted to the overall needs of the communication system [3].
An interesting observation can be made by looking at the simulation results presented in [8]. With the choice of very “costly” parameters ($k = 25000$, s-random interleaver, 20 iterations) the limit of channel $\boldsymbol{h1}$ can be approached within 1 dB at a bit error rate of 10^{-5}. This is especially interesting because “only” a (7,5) NSC encoder was employed. The outer code therefore has a smaller free distance than the code of the (37,21) RSC encoder and significantly less decoding time is needed. It seems to be more important to fully take advantage of the interleaver gain, i.e. to employ a precoder, to design good interleavers and to use large block lengths than to employ a potent convolutional outer encoder at least for the bit error rates and channels considered here.
It was also pointed out that for some signal to noise ratios performance significantly decreases when a precoder is used. An explanation can be found in [9] where it is shown that precoding results in a loss of reliability during the first iteration of turbo equalization.

Chapter 4

Equalization and Turbo Decoding

4.1 Turbo Codes

Error-correcting capability of rate-1/2 convolutional codes can be improved by increasing the memory M of the underlying encoder. Unfortunately this is accompanied by the problem that decoding complexity also increases exponentially with M up to a point where optimal decoding becomes impractical.
The general idea behind turbo codes, which are a class of so called concatenated codes, is to improve error-correcting performance by combining two or more convolutional encoders having relatively small memory M such that the code of each encoder can be decoded with reasonable complexity. Performance is improved by applying the turbo principle in the decoder, i.e. allowing the decoders to exchange their extrinsic information about the information bits.

4.1.1 Encoder

A turbo code is the set of all codewords resulting from a parallel concatenated convolutional encoder which is depicted in Figure 4.1 for $R = 1/3$ [4].
The encoder consists of two identical RSC encoders but only the redundancy part (i.e. the second output) is considered. The information vector $\boldsymbol{u}$ is interleaved by a random interleaver before being passed to the second RSC encoder. Thus $\boldsymbol{c}_i$ (the part of the codeword at time

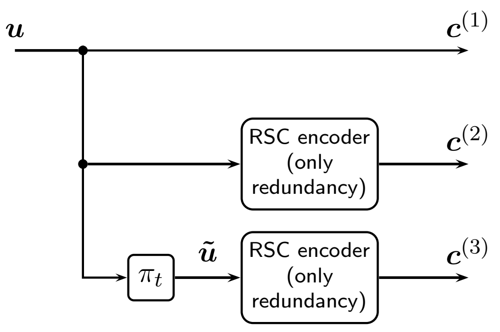


Figure 4.1: Parallel concatenated convolutional encoder with rate 1/3. The set of all codewords is a turbo code.

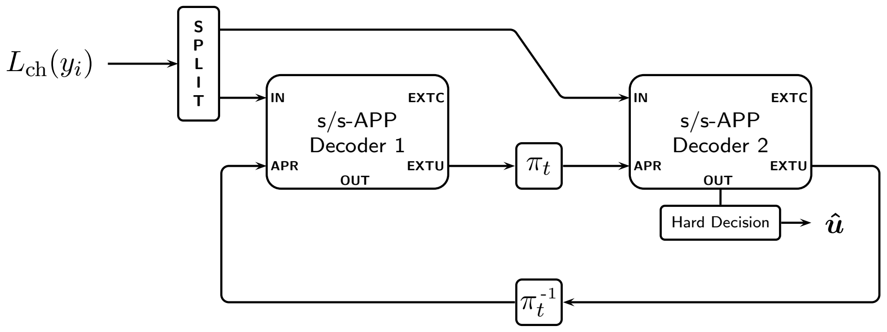


Figure 4.2: In a turbo decoder two s/s-APP modules work together on the same observation about the information bits by iteratively exchanging their extrinsic information.

i) consists of the information bit u_i and two parity bits $c_i^{(2)}$ and $c_i^{(3)}$ respectively. This can be seen as a conventional concatenation of two convolutional codes [13].

A rate-1/2 code can be obtained by periodically eliminating code bits in the codewords of a rate-1/3 code such that the resulting codewords have length $2k$. This method is called puncturing. In this work only parity bits in $\boldsymbol{c}^{(2)}$ and $\boldsymbol{c}^{(3)}$ are punctured and the systematic part $\boldsymbol{c}^{(1)} = \boldsymbol{u}$ of the codeword remains unaltered because observation about $\boldsymbol{c}^{(1)}$ will be used by both decoding modules in a turbo decoder (see below). The puncturing pattern is known to the receiver and missing observations about punctured code bits are added in the receiver by assuming L-values $L_{\mathrm{ch}}(y_i) = 0$ for all punctured code bits.

4.1.2 Decoder

A turbo decoder is shown in Figure 4.2. The simplified transmitter (Figure 2.13 (b)) is assumed, i.e. a codeword is BPSK modulated and then transmitted over an AWGN channel. The SPLIT-module allocates the input likelihoods about the code bits $L_{\mathrm{ch}}(y_i) = 2y_i/\sigma^2$ to the two decoder modules. The likelihoods about $\boldsymbol{c}^{(1)}$ and $\boldsymbol{c}^{(2)}$ are passed to the first decoder and the interleaved likelihoods about $\boldsymbol{c}^{(1)}$ together with the likelihoods about $\boldsymbol{c}^{(3)}$ are passed to the second decoder. That means both decoders work with the same observations about the information bits.
The essential idea is the application of the turbo principle in the decoder as follows. Initially the first decoder calculates s/s-APP L-values based solely on the input observation likelihoods. Extrinsic information about the information bits is extracted, interleaved and passed to the second decoder. This module assumes these L-values to be statistically independent a priori L-values of the information bits. Together with its channel observation s/s-APP L-values are calculated. Further iterations are performed by introducing a feedback loop from the second to the first decoder passing forward again only the extrinsic L-values. The first decoder can then calculate new s/s-APP L-values with “refined” a priori L-values of the information bits and so on. The iterative process can be stopped either by using a stop criterion or after a defined number of iterations.

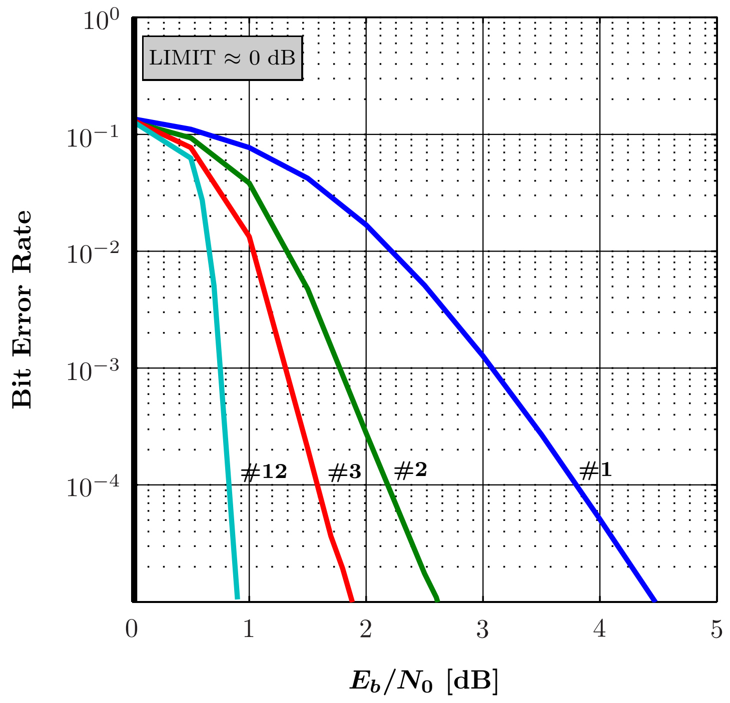


Figure 4.3: Simulation results for a turbo decoder when the encoded bits are transmitted over an AWGN channel without ISI.

Performance on an AWGN channel without ISI

Turbo codes have been shown to provide good error-correcting performance close to the information theoretical limits of an AWGN channel when large block lengths k are used [4].

The bit error rate of a transmission line consisting of a parallel concatenated encoder with two (37,21) RSC encoders ($R = 1/2$) and a turbo decoder according to Figure 4.2 was simulated. Bits were BPSK modulated and transmitted over an AWGN channel without ISI. A block length of $k = 10000$ was used. The results are plotted in Figure 4.3.
It can be seen that for a bit error rate of 10^{-5} the gap to the theoretical limit is only 0.9 dB after 12 iterations. This “near Shannon limit error-correcting coding and decoding” [4] was the starting point that led to a thorough investigation of turbo codes and the underlying theory in a variety of applications.

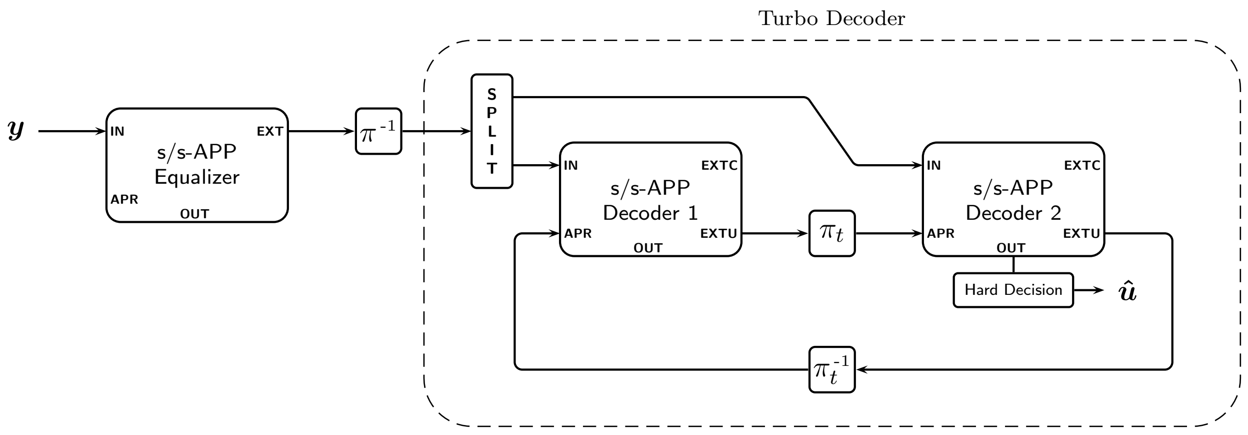


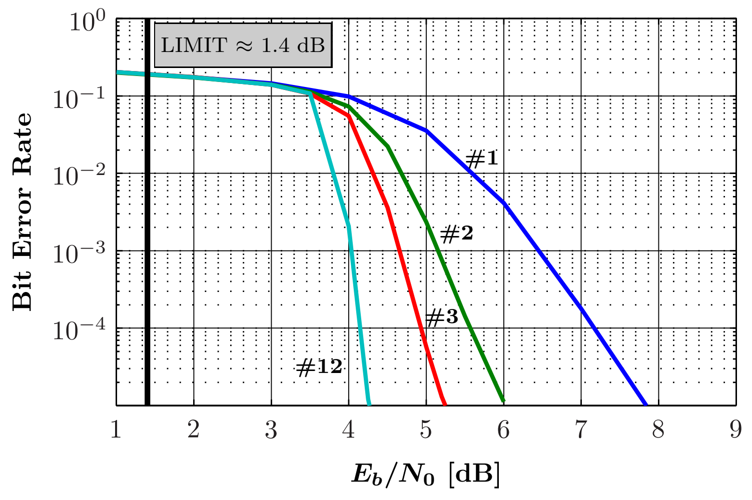
Figure 4.4: Separate equalization and turbo decoding.

4.2 Separate Equalization and Turbo Decoding

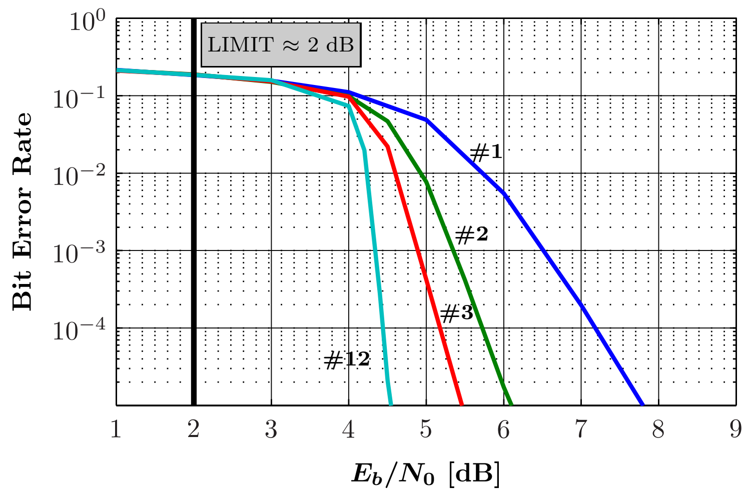
The convolutional encoder in the original transmission line (Figure 2.3) is now replaced by a parallel concatenated convolutional encoder with rate $R = 1/2$.
The first receiver design is equivalent to the approach of separate equalization and decoding which was presented at the end of Chapter 2. The receiver structure is shown in Figure 4.4.
The equalizer takes the received channel symbols $\boldsymbol{y}$ and provides s/s-APP L-values of the channel symbols $\boldsymbol{x}$. These L-values are interleaved and passed to the turbo decoder. Turbo decoding is performed as described in Section 4.1.2.
Note that the used interleavers have different lengths. The interleaver separating the equalizer and the turbo decoder permutes the code bits and has length $n = 2k$. The interleavers employed in the turbo decoder (which are denoted with an index t) correspond to the interleaver in the parallel concatenated encoder and have length k.

4.2.1 Simulation

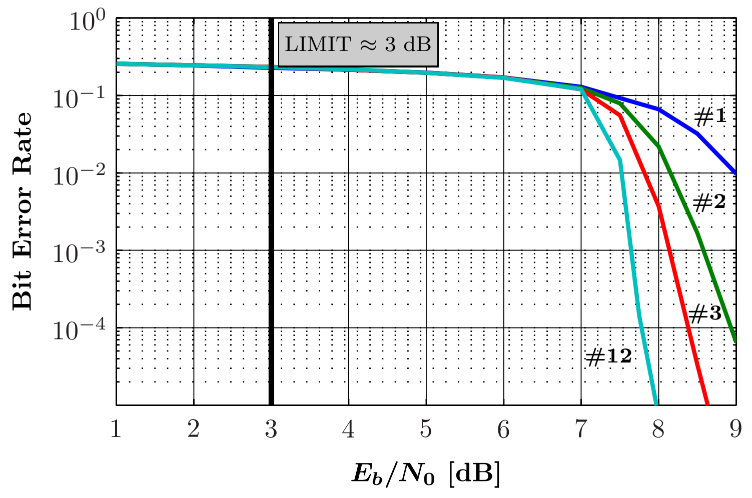
The receiver in Figure 4.4 was tested by using a block length of $k = 10000$, random interleavers and two (37,21) RSC encoders in the encoder. Results are shown in Figure 4.5. The number next to a curve corresponds to the number of iterations performed by the turbo decoder.
The resulting curves look similar to the curves presented for the turbo decoder when transmitting over an AWGN channel without ISI – only with different "offsets" to the limits of the ISI channels. The gap between limit and achievable minimum signal to noise ratio for a bit error rate of 10^{-5} after 12 iterations increases with the limit of the ISI channel. This suggests that the good error-correcting performance of turbo codes is strongly impaired when likelihoods about code bits do not result from an ideal AWGN channel but from an equalizer trying to cope with intersymbol interference - especially when the ISI channel has severe frequency distortions like channel $\boldsymbol{h3}$. The performance of the scheme suffers from a suboptimal estimation of the channel symbols because no a priori input is available for the equalizer.



(a) ISI channel $\boldsymbol{h1}$



(b) ISI channel $\boldsymbol{h2}$



(c) ISI channel $\boldsymbol{h3}$

Figure 4.5: Simulation results for separate equalization and turbo decoding as a function of iterations performed in the turbo decoder.

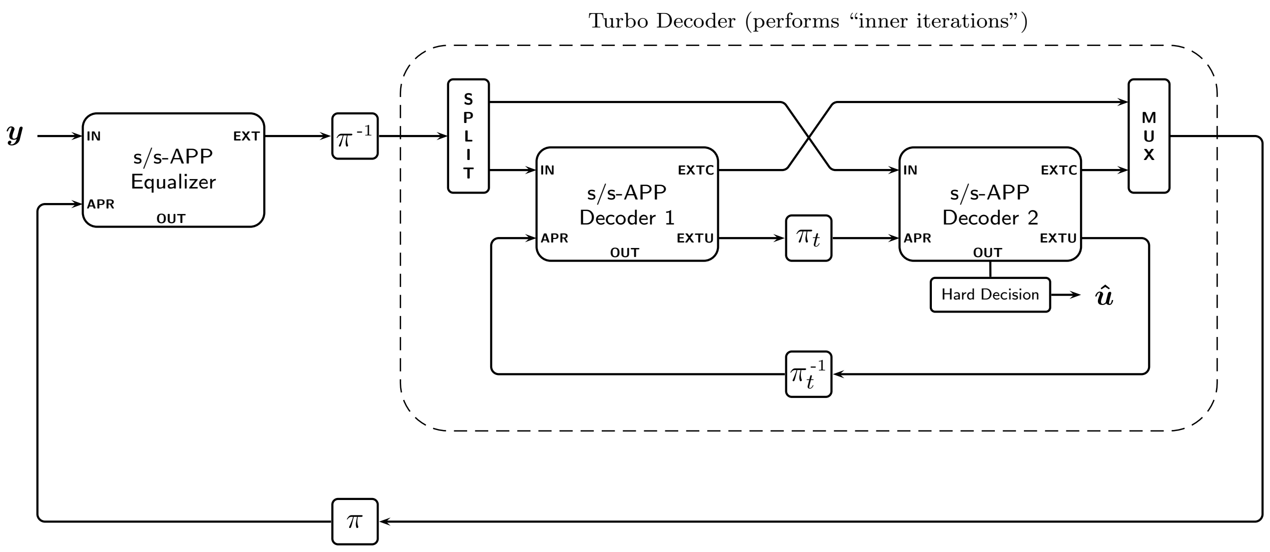


Figure 4.6: Combined turbo equalization and turbo decoding. The extrinsic information about the code bits is fed back to the equalizer and used as a priori information in the next iteration.

4.3 Combined Turbo Equalization and Turbo Decoding

Now the turbo principle is applied to the equalization process. Refined a priori L-values of the channel symbols $\boldsymbol{x}$ are obtained by feeding back extrinsic information about the code bits from the turbo decoder to the equalizer. A complete receiver employing combined turbo equalization and turbo decoding is shown in Figure 4.6.

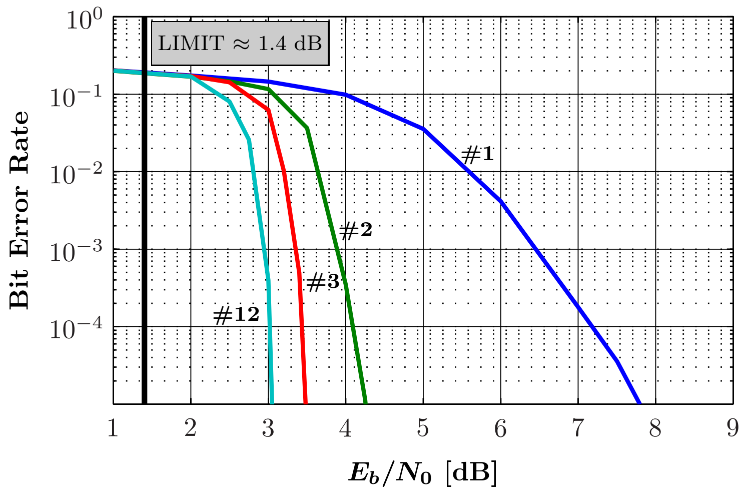
It is necessary to define extrinsic information about the code bits with respect to the turbo decoder. So far extrinsic information about code bits is only defined with respect to a particular s/s-APP module (see Equations (2.42) and (2.43)). However the turbo decoder consists of two s/s-APP decoding modules, so that the two outputs about extrinsic code bit information (EXTC) of each decoder have to be combined. This is done by the MUX-module as follows.
Obviously the extrinsic L-values of the code bits $\boldsymbol{c}^{(2)}$ must result from the first decoder whereas the extrinsic L-values of the code bits $\boldsymbol{c}^{(3)}$ must result from the second decoder. However the two modules operate with the same observation about the systematic part of the codeword $\boldsymbol{c}^{(1)}$ (i.e. the information bits). It follows that both decoders provide extrinsic information about this part of the codeword. Therefore we define the superposition of the latest refined extrinsic L-values of both decoders about a specific information bit to be the extrinsic information about the corresponding code bit in the systematic part of the codeword with respect to the turbo decoder.
A closer look at Equations (2.42) and (2.41) reveals that the extrinsic L-value of a code bit in the systematic part of the codeword for each decoder always consists of two parts: the extrinsic L-value of the information bit plus the a priori L-value of that information bit. In a turbo decoder the *a priori* L-value of an information bit used by one decoder is the *extrinsic*

L-value of that information bit of the other decoder. Therefore the extrinsic L-values of the code bits $\boldsymbol{c}^{(1)}$ of the second decoder can be used as extrinsic information about this part of the codeword (after being deinterleaved) with respect to the turbo decoder.
The MUX-module combines the extrinsic L-values of $\boldsymbol{c}^{(1)}$, $\boldsymbol{c}^{(2)}$ and $\boldsymbol{c}^{(3)}$ in the correct order so that they can be used as a priori information about the channel symbols by the equalizer after interleaving.

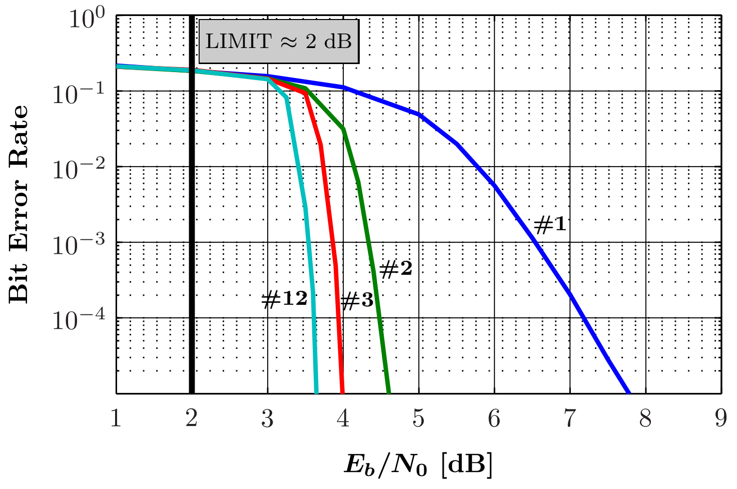
4.3.1 Simulation

The simulation setup used to test the combined receiver was similar to the setup in the last section for separate equalization and turbo decoding, i.e. $k = 10000$, random interleavers and (37,21) RSC encoders. Results are shown in Figure 4.7. The number of iterations performed by the turbo decoder (inner iterations) is equal to the overall number of iterations for turbo equalization (outer iterations). So when b outer iterations are performed the total number of inner iterations add up to b^2.

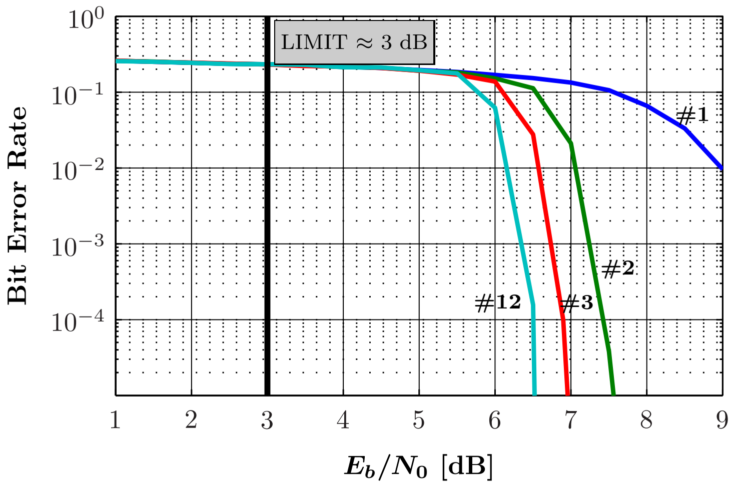
For the combined approach a large performance gain can be observed during the second outer iteration which is very distinctive. After this second outer iteration the combined approach offers comparable performance to the separate approach after 12 iterations (done solely by the turbo decoder) at a bit error rate of 10^{-5}. With other words a total of 4 iterations done by the turbo decoder in the combined approach is sufficient to allow performance similar to the separate approach where 12 iterations are performed by the turbo decoder.
Nonetheless one can also see that performing additional outer iterations only achieves a minor performance gain. Even for 12 outer iterations and a total of 144 inner iterations only about 1 dB can be gained compared to 2 outer iterations and a total of 4 inner iterations at a bit error rate of 10^{-5} for each channel.
When evaluating the combined approach in absolute terms with respect to the theoretical limits it is necessary to differentiate between the three ISI channels. For channels $\boldsymbol{h1}$ and $\boldsymbol{h2}$ and a bit error rate of 10^{-5} the signal to noise ratio is about 1.6 dB away from the limit which is comparable to the approach where a convolutional outer encoder and a differential precoder is used. However a large amount of total iterations has to be performed. For channel $\boldsymbol{h3}$ the gap to the limit is about 3.5 dB at a bit error rate of 10^{-5}. This performance is significantly worse than the performance of a turbo equalization scheme where a conventional outer convolutional encoder is employed – regardless of whether the ISI channel is precoded or not.



(a) ISI channel $\boldsymbol{h1}$



(b) ISI channel $\boldsymbol{h2}$



(c) ISI channel $\boldsymbol{h3}$

Figure 4.7: Simulation results for combined turbo equalization and turbo decoding as a function of iterations for turbo equalization. Number of iterations performed by the turbo decoder are equal to the overall number of iterations for turbo equalization.

4.4 Discussion

The approach of combined turbo equalization and turbo decoding provides good performance for ISI channels $\boldsymbol{h1}$ and $\boldsymbol{h2}$. The minimum signal to noise ratio which is needed to achieve a bit error rate of 10^{-5} after 12 iterations is comparable to that of the approach where a convolutional encoder and a differential precoder is used. However the results for channel $\boldsymbol{h3}$ show that the performance of the scheme decreases rapidly for worse ISI channels. This is very apparent when comparing the performance results for channel $\boldsymbol{h3}$ with the approaches in the last chapter where a convolutional outer encoder is used. For both the normal ISI channel and the precoded ISI channel turbo equalization offers a bit error rate of at least 10^{-5} at 5.5 dB whereas the approach of combined turbo equalization and turbo decoding only achieves a bit error rate of 10^{-1}. This has to be considered when thinking about practical applications of this scheme for example in mobile communication systems. Because usually the underlying ISI channel is not time-invariant but changes with time and is only *assumed* to be time-invariant for short periods. Thus a receiver should provide good performance both when the ISI channel suffers from minor *and* severe frequency distortions because both scenarios might only a be a short time instant apart.

Another problem of the combined approach is its high complexity and the question how it can be reduced. 12 iterations performed for turbo equalization means that for each new estimation of channel symbols the turbo decoder has to perform 12 inner iterations such that a total of 144 inner iterations are performed. This shows the need for a suitable stop criterion so that the total number of iterations performed by the turbo decoder can be reduced. The question however is how to define a criterion as suitable.
The combined approach was also tested including a differential precoder. The effect was a general decrease in performance regardless of the ISI channel. More importantly for some constellations of outer iterations and inner iterations the combined decoder also did not seem to provide a stable solution, i.e. L-values increased rapidly resulting in numerical instabilities in the simulation setup. This strongly underlines the need for suitable stop criterion besides complexity reasons.
Unfortunately the paper in which the combined approach of turbo equalization and turbo decoding is proposed [10] contains very little information about the inner turbo decoder and it only can be guessed how many inner iterations are performed with respect to one outer iteration. The list of possible combinations is literally endless and can comprise a stop criterion of some sort as well as a fixed number of iterations which somehow depend on the number of outer iterations (equal, increasing, decreasing, . . .). For the latter there is no reason to prefer or avoid one of those methods apart from a brute force method to check all combinations despite maybe heuristic considerations.
This is closely related to the question of convergence and stability of the combined receiver because it contains two feedback loops. Regardless of the stop criterion or whether a fixed number of inner iterations is used the receiver should ensure stability in any case. In fact no simple answer for this question exists even when only considering either turbo equalization or turbo decoding alone.
The decoder structure presented in [10] also contains some inconsistencies like the fact that the channel likelihoods about the code bits are not subtracted from the s/s-APP L-values when

passing extrinsic information from one decoder to the next but are subtracted when passing extrinsic L-values back the equalizer. It also remains uncertain why a transformation from the channel MAP's extrinsic L-values to equivalent soft channel input is needed. This seems to be merely a necessity resulting from the introduced s/s-APP modules which only accept soft channel input rather than to have any effect on the decoding process itself.

A problem that arises with the use of parallel concatenated convolutional encoders is the error floor. Bit error rates tend to decrease rapidly for medium signal to noise ratios but then hit an error floor where the bit error rate remains almost constant even when the signal to noise ratio increases. The fact that the curves in Figure 4.5 do not deviate very steeply supposes that an error floor is present at around a bit error rate of about 10^{-6}. This topic was avoided in thesis by considering only bit error rates up to 10^{-5} but applications like magnetic disc recoding require much lower bit error rates so that this topic becomes relevant there.

As a summary it can be said that in some situations the combined decoder can offer comparable performance to the approach where a classical convolutional outer encoder and a differential precoder is used. Unfortunately performance is highly dependent on the underlying ISI channel. Apart from that an immense amount of iterations has to be performed and the problem of convergence and stability remains unsolved.

Chapter 5
Conclusion

In this thesis different approaches to counter the detrimental effects of noisy ISI channels have been presented.
It was shown that the application of the turbo principle in the receiver and performing iterative estimations about the channel symbols can significantly increase performance. Extrinsic information about the code bits is therefore fed back from the decoder to the equalizer and used as a priori information about the channel symbols. This is equivalent to a decoder which iteratively decodes a serially concatenated convolutional code. The introduction of a precoder was motivated by design rules derived for such serially concatenated convolutional codes which state (among other things [17]) to always employ a recursive inner encoder in order to achieve an interleaver gain. Simulations revealed that implementing a precoder in fact seems to let the receiver exploit the structure of the transmission line very well. It was derived that for further improvements it might be more important to fully exploit the interleaver gain than to employ potent error-correcting outer convolutional codes – at least for the bit error-rates that have been considered. It is also important that different precoders have different effects on the performance and the precoder choice should therefore be adopted to the needs of the communication system [3].
Then a parallel concatenated convolutional encoder was employed because it is well known that theses encoders together with a turbo decoder provide good error-correcting performance for AWGN channels without ISI. However a first approach of successive equalization and turbo decoding revealed that the performance is highly dependent on the underlying ISI channel and significantly decreased for worse ISI channels. In a receiver which combines turbo equalization and turbo decoding it is necessary to perform inner iterations in order to decode the outer code resulting in a large complexity. Despite its high complexity the performance of this approach could not exceed that of the approach where a conventional convolutional code and a differential precoder was used. This and the fact that the problem of convergence and stability is not sufficiently solved let this scheme appear to be a suboptimal choice. In this context it is noticeable that publications in which results on the field of turbo equalization are summarized, combined approaches including turbo decoding are only mentioned as a side note [18]. In fact reading those papers suggests that most recent research focuses on ways to make turbo equalization more practical by *reducing* complexity (for example by including suboptimal equalization techniques) instead of *increasing* complexity by introducing more feedback loops in the receiver.

Bibliography

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). 20(2):284–287, March 1974.

[2] M. Tüchler, R. Kötter, and A. C. Singer. Turbo equalization: principles and new results. *IEEE Transactions on Communications*, 50(5):754–767, May 2002.

[3] I. Lee. The effect of a precoder on serially concatenated coding systemswith an ISI channel. *IEEE Transactions on Communications*, 49(7):1168–1175, July 2001.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. volume 2, pages 1064–1070 vol.2, 1993.

[5] C. E. Shannon. *A Mathematical Theory of Communication.* CSLI Publications, 1948.

[6] C. Douillard, A. Picart, P. Didier, M. Jezequel, C. Berrou, and A. Glavieux. Iterative correction of intersymbol interference: turbo equalization. *European Trans. Telecommun*, (6):507–511, 1995.

[7] G. Bauch, H. Khorram, and J. Hagenauer. Iterative equalization and decoding in mobile communications systems. In *Second European Personal Mobile Communications Conference (2. EPMCC '97)*, pages 307–312, 1997.

[8] R. Kötter, A. C. Singer, and M. Tüchler. Turbo equalization. *IEEE Signal Processing Mag*, 21:67–80, 2004.

[9] K. R. Narayanan. Effect of precoding on the convergence of turbo equalization forpartial response channels. In *Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE*, volume 3, pages 1865–1871, San Francisco, CA, USA, 2000.

[10] D. Raphaeli and Y. Zarai. Combined turbo equalization and turbo decoding. In *Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE*, volume 2, pages 639–643, Phoenix, AZ, November 3–8, 1997.

[11] Proakis. *Digital Communication.* McGraw-Hill, 2001.

[12] J. Lindner. *Informationsuebertragung.* Springer Verlag, 2005.

[13] M. Bossert. *Kanalkodierung.* Teubner Verlag, 1998.

[14] D. Arnold and H.-A. Loeliger. On the information rate of binary-input channels with memory. In *Proc. IEEE International Conference on Communications ICC 2001*, volume 9, pages 2692–2695 vol.9, 2001.

[15] J. Hagenauer. The turbo principle: Tutorial introduction and state of the art. 1997.

[16] H. Khorram. Iterative Entzerrung und Kanaldecodierung bei frequenzselektiven Kanälen mit dem MAP-Algorithmus. Master's thesis, Technische Universität München, 1997.

[17] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding. *IEEE TRANS. INFORM. THEORY*, 44(3):909–926, 1998.

[18] M. Tögel, W. Pusch, and H. Weinrichter. Combined serially concatenated codes and turbo-equalization, 2000.