Convolutional Coupled Codes

Dem Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
vorgelegte

Dissertation

von
Dip.-Ing. Slim Chaoui
geboren am 20. Mai 1972 in Sfax

Referent:	Prof. Dr.-Ing. Bernhard Dorsch
Korreferent:	Prof. Dr. Han Vinck
Tag der Einreichung:	25. Oktober 2002
Tag der mündlichen Prüfung:	03. Februar 2003

D 17
Darmstädter Dissertation

To my parents

Fatma Hammami and Habib Chaoui

Acknowledgements

This Thesis is an outgrowth of my activities as a scientific staff at the Institute for Communications Technology at the Technical University of Darmstadt (TUD). I would like thank all who made this possible. I wish to express my gratitude to my advisor Professor Bernhard Dorsch for having helped inspire this research, for having generously provided the academic freedom and also for having supplied the much-needed encouragement throughout this time. I would also like to thank professor Han Vinck form the department of Experimental Mathematics at the University of Essen that was willing to be the co-referee.

During my activities at the TUD, I have had the opportunity to work with and to learn from a number of colleagues and friends, who I would like to express my appreciation. In particular, I thank my colleague Norbert Stolte for the attendance to proof-read this work.

Darmstadt, February 2003
Slim Chaoui

Abstract

This thesis is about convolutional coupled codes - codes constructed via concatenation of several outer systematic convolutional encoders and several inner systematic block encoders linked by divers interleavers. The code is nonsystematic, since only the redundancy produced from the outer and inner encoders is transmitted. The focus of the work is on the understanding and design of convolutional coupled codes. This includes thorough investigation of central components that influence convolutional coupled code performance, such as the constituent encoders, as well as the procedure of iterative decoding. The investigations are carried out for transmission on additive white Gaussian noise channels.

Two aspects that influence the performance of convolutional coupled codes are considered: (1) code properties, in terms of effective free distance, and (2) decoding properties, in terms of the performance of the iterative decoding. It is asserted that both these aspects are influenced by the choice of the inner and outer constituent encoders. Guidelines for the choice of constituent encoders are outlined. It is demonstrated that the “error floor” convolutional coupled codes are claimed to suffer from at medium- to high signal to noise ratios can be significantly lowered by proper chose of the inner encoding matrix.

The aspect of convergence behavior of the iterative decoding is investigated. The influence of code memory, code polynomials as well as different inner codes on the convergence behavior is studied.

Inside this thesis we show that convolutional coupled codes have potential of being realistic alternative to other concatenated schemes especially Turbo codes.

Kurzfassung

Die vorliegende Arbeit befasst sich mit verkoppelten Faltungscodes - Codes die durch Verkettung von mehreren systematischen äusseren Faltungscodes und mehreren systematischen inneren Blockcodes konstruiert sind, wobei die einzelnen Eingangssequenzen der äusseren Codes verschachtelt werden, bevor sie in die inneren Codes hineingehen. Anders als bei den parallelen verketteten Codes, sind die verkoppelten Codes nicht systematisch, da nur die Redundanz, erzeugt von inneren und äusseren Codierern, übertragen wird. Der Schwerpunkt dieser Arbeit ist das Verständnis und das Design von verkoppelten Faltungscodes. Dies umfasst eine gründliche Untersuchung der zentralen Bestandteile, die Einfluss auf die Codefähigkeit haben, wie das Zusammensetzen der Codekomponenten und auch das iterative Decodierungsverfahren. Die Untersuchungen wurden nur für den additiven weissen gaußschen Rauschkanal (AWGN) betrachtet.

Zwei Aspekte, die Einfluss auf die Leistungsfähigkeit von verkoppelten Faltungscodes ausüben, werden berücksichtigt: (1) Codeeigenschaften, im Sinne von effectiver freier Distanz, und (2) Decodierungseigenschaften, im Sinne von Leistungsfähigkeit der iterativen Decodierung. Man

stellt fest, dass diese beiden Aspekte von der Wahl der inneren und äusseren Komponentencodes beeinflusst werden. Richtlinien für die Wahl der Komponentencodes werden angegeben. Es stellt sich heraus, dass die durch die minimale Distanz bedingte Abflachung der Fehlerkorrigierfähigkeit von verkoppelten Faltungscodes im mittleren bis höheren Signal-Rauschverhältniss, durch die Wahl von geeigneten inneren Codes bedeutend gesenkt werden kann.

Der Aspekt des Konvergenzverhaltens der iterativen Decodierung wird untersucht. Die Einflusse der Generatorpolynome von Faltungscodes sowie unterschiedliche inneren Codes werden studiert.

In dieser Arbeit wird gezeigt, dass verkoppelte Faltungscodes das Potential besitzen eine realistische Alternative zu anderen verketteten Codes, insbesondere zu den Turbo Codes, zu sein.

Contents

1 Introduction **1**

1.1 A Model of Communication Systems 3

1.1.1 Channel Models 4

1.1.2 Channel Capacity 5

1.1.3 The Optimum Decoder 6

1.2 Outline of the Thesis 7

2 Block and Convolutional Codes **9**

2.1 Linear Binary Block Codes 9

2.1.1 Distance Properties 11

2.1.2 Trellis Presentation 11

2.2 Convolutional Codes 12

2.2.1 Finite Code Sequences 15

2.2.2 State Diagram 16

2.2.3 Trellis 16

2.2.4 Distance Properties of Convolutional Codes 17

3 Concatenated Codes **21**

3.1 Serial Concatenated Codes 21

3.1.1 Product Codes 22

3.1.2 Choice of the Component Codes 24

3.1.3 Iterative Decoding of Serially Concatenated Codes 24

3.2 Parallel Concatenated codes 25

3.2.1 The Code Rate Gain of Parallel Concatenated Codes 27

4 Turbo Codes **29**

4.1 Encoder Structure 30

4.1.1 The Constituent Encoders 30

4.1.2 Interleaving 31

4.1.3 Trellis Termination . . . 32
4.1.4 Puncturing . . . 33
4.2 Distance Spectra and Union Bound . . . 33
4.3 Iterative Decoding . . . 36
4.3.1 APP Decoding . . . 38
4.4 Conclusion . . . 40

5 Convolutional Coupled Codes 41
5.1 Encoder Structure . . . 41
5.1.1 Basic Idea . . . 43
5.1.2 Generator Matrix of a Parallel Composition of Encoders . . . 45
5.1.3 Generator Matrix of the Convolutional Coupled Code . . . 47
5.1.4 Distance Properties of the Coupled Codes . . . 49
5.2 Decoding of Convolutional Coupled Codes . . . 50
5.2.1 Decoding Complexity . . . 54
5.3 Effective Free Distance . . . 55
5.4 Interleaving . . . 57
5.4.1 The Modulo Interleaver . . . 58
5.4.2 The Random Interleaver . . . 59
5.4.3 The s-Random Interleaver . . . 59
5.4.4 Simulation Results . . . 60
5.5 Stop Criteria . . . 61
5.6 Performance Bound . . . 65
5.7 Error Performance Simulations . . . 66
5.7.1 Comparison between the Performance of Convolutional Coupled Codes and Turbo Codes . . . 69
5.8 Conclusion . . . 70

6 Iterative Decoding Trajectories 73
6.1 Extrinsic Transfer Characteristics . . . 73
6.2 Extrinsic Information Transfer Chart . . . 78
6.2.1 Trajectories of Iterative Decoding . . . 79
6.2.2 Estimation of the BER from the EXIT chart . . . 80
6.3 Conclusion . . . 81

7 Choosing Constituent Codes 83
7.1 Choice of the RSC Outer Codes . . . 83
7.1.1 Code search based on the effective free distance . . . 83

7.1.2 Code Search Based on the EXIT Chart Technique 86
7.2 Conclusion 88

8 CCC Extended with Outer BCH Code **91**
8.1 BER Estimation for High SNRs 91
8.2 Simulation 92
8.3 Conclusion 94

9 Conclusions **95**

A The BCJR Algorithm **99**
A.1 Theoretical Description 99
A.2 Implementation Aspects 105

B Tables with good RSC codes **107**

C List of Symbols and Abbreviations **109**

Chapter 1

Introduction

The mathematical foundation for digital communication was established by C. Shannon in a series of fundamental papers in 1948 [58] and [59]. In these papers, he proved the channel coding theorem, which since 1948 has been the prime motive behind channel coding research. Shannon's central theme was that if the signalling rate of the system is less than the channel capacity, reliable communication can be achieved if one chooses proper encoding and decoding techniques.

Practical codes like block- and convolutional codes are far from this theoretical boundary. The error-correcting capability of block codes improves as the block length n increases, or in the case of convolutional codes with increasing constraint length m. However the implementation complexity of Maximum Likelihood (ML) decoders of such codes increases exponentially with n respectively m up to a point where decoding becomes physically unrealizable. A practical method to obtain long codes with only a linear increase in decoding complexity is code concatenation. This technique was proposed by Forney in 1966 [28]. Forney invented an encoding scheme where a code of large alphabet size, the *outer code* A, is combined with a code of small alphabet size, the *inner code* B. The message digits are first encoded over the large alphabet size resulting in a code word of Code A. Each code symbol of code A is then encoded with the encoder for the code B over the small alphabet size. In this way, a long code is obtained by concatenation of two shorter codes. The length of the concatenated code is the product of the lengths of the component codes A and B. Usually the minimum distance[1] can be made at least as large as the product of the two minimum distances. But the decoding complexity increases only with the decoding complexity of the codes A and B, because the component codes can be decoded separately and one after the other. In order to improve this coding principle, it has been suggested to introduce an interleaver between the two encoders, which provides the correction of error bursts from the inner decoder by the outer decoder.

In 1993, these two principles, code concatenation and interleaving, are associated by C. Berrou,

[1]The minimum distance of a code means the minimum Hamming distance - the minimum number of positions in which one codeword of the code differs from any other codeword.

A. Glavieux and P. Thitimajshima with a third principle known as "iterative decoding" from Gallager [35]. The result was the so called "Turbo codes" [11], whose astonishing performance has given rise to a large interest in the coding community. They are *parallel concatenated codes* (PCC) whose encoder is formed by two (or more) constituent systematic encoders joined through an interleaver. The input information bits are fed to the first encoder and, after having been interleaved by the interleaver, enter the second encoder. The code word of the parallel concatenated code consists of the input bits to the first encoder followed by the parity check bits of both encoders. Since the interleaving length is normally very large, maximum likelihood decoding would be of astronomical complexity and is, thus out of question. The proposed suboptimal decoder [11] implements an iterative algorithm whose central core is a maximum *a posteriori* symbol by symbol decoder. On account of good capability, many other codes which are characterized through code concatenation, interleaving and iterative decoding are proposed. In the scope of this thesis, we will investigate a new class of code concatenation, the so called *convolutional coupled codes*, in which a number of convolutional codes are coupled with a number of block codes. Differently from the parallel concatenated Turbo codes, only the redundancy produced by the constituent codes is transmitted.

In the literature, it was shown for Turbo codes, that flattening of the error rate performance at moderate and high signal to noise ratios (SNR), the so called *error floor* is a consequence of the relatively low free distance of the code. It was shown, that the error floor can be lowered by increasing the size of the interleaver or by optimizing the polynomials of the component codes. Alternatively, the performance at this region may be improved by use of sophisticated interleaver. Therefore, took interleaver design a great importance during the investigation of Turbo codes. Nevertheless, one couldn't master analytically the phenomenon of error floor. With convolutional coupled codes, we have achieved a constructive controlled error floor, without essential disadvantage at small SNRs, where Turbo codes works very well. The minimum distance can hence be estimated analytically. In application where the error floor is a problem, convolutional coupled codes may be an effective solution. Thus, convolutional coupled codes have the potential of being realistic alternative to Turbo codes.

The basic idea in this thesis is based on the cognizance, that coding not only redundancy (i.e. the transmission of additional parity digits), but also diversity (i.e. in coding: the smearing of the information over many digits). Thereby, it is generally known that the information digits in a code word contributes relatively little to the minimum distance of a code. Thus, we transmit a unique reversible mapping of the information word with the objective, that in the case where the parity weight is relatively little, the weight of the over-all code word will be larger. The part of the (new) information word on the minimum distance is described by the so called *coupling factor*. According to this, the codes are called *coupled codes*.

In this introductory chapter, the general concept of a communication system is introduced and further the role of channel coding is described. Finally the outline of the thesis is given.

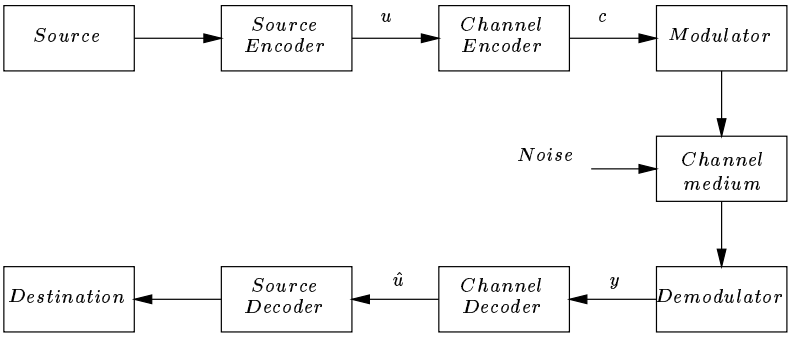

Figure 1.1: Block diagram of a communication system.

1.1 A Model of Communication Systems

A complete communication system includes numerous areas of interesting and challenging problems. Most modern communication systems operate in the digital domain, which offers a large amount of signal processing possibilities for system design. Both for conceptual- and implementation purposes, it is common to partition the chain of processing performed in a communication system into separate building blocks, thereby forming a comprehensible model of the system. Figure 1.1 shows such a model, whose building blocks are briefly introduced in the sequel.

The *source* can be either a person or a machine (e.g., a digital computer). The source output, which is to be transmitted to the destination, can be either a continuous waveform or a sequence of discrete symbols. The *source encoder* transforms the source output into a sequence of binary digits (bits) called the information sequence u. In the case of a continuous source, this involves analog-to-digital (A/D) conversion. The source encoder is ideally designed so that the number of bits per unit time required to represent the source output is minimized in a way that the source output can be reliably reconstructed. The next block in the communication model is the *channel encoder.* While the source encoder is chosen with respect to the particular information source, the channel encoder is typically chosen with respect to the channel the messages are transmitted on. The purpose of the channel encoder is to make the transmitted messages less susceptible to, for example, noise and interference introduced by the channel. In contrast to the source encoder, which removes redundancy from the source sequence, the functionality of the channel encoder relies on the principle of adding structured redundancy. Next, the *modulator* converts digital output symbols c in analog signals which can be transmitted over the *channel medium.* Because the channel is subject to various types of noise, distortion, and interference, the channel output differs from the channel input. The *demodulator* processes each received waveform of duration T and produces an output that may be discrete or continuous. The sequence of demodulator outputs corresponding to the encoded sequence c is called the received sequence y. The *channel decoder* uses the redundancy in a channel code word to correct the errors in the received word and then produces an estimate of the source code word. If all errors are corrected, the estimated source code word matches the original source code word. The source decoder transforms the estimated sequence $\hat{u}$ into an estimate of the source output and delivers this estimate to the

destination. When the source is continuous, this involves digital-to-analog (D/A) conversion. In a well designed system, the estimate will be a reliable reproduction of the source output except when the channel is very noisy.

1.1.1 Channel Models

As the task of channel coding is to adjust the information stream to a given channel we have to discuss some channel models. A model of a communication system is a mathematical representation defined to realistically describe the way signals are constructed and processed and the way they are affected by the real-world communication environment. We need to formalize the transmission model in order to compute how much information we can transmit over a given channel. Moreover we need a quantitative measure of information.

The Discrete-Time Channel

It is convenient to define a channel model to include the modulator, the demodulator, and all the intermediate transmission equipment and media. This model is compactly defined by the set of modulator inputs, the set of demodulator outputs, and the statistics that relate the possible outputs to each possible input. This is commonly called a *discrete-time channel model* or simply a *discrete channel*. The input-to-output statistics represent the ways in which the modulated signals are affected by amplitude and phase fluctuations, noise, interference, and equipment nonidealities and impairments. In most cases it is very difficult to define a model that thoroughly accounts for all the disturbances affecting the signals and one must resort to reasonable approximations. However, experience has shown that even reasonably simple channel models can provide a sufficient degree of realism to enable proper design of efficient systems. Furthermore simplified models often yield insights into underlying principles, which can be obscured by a myriad of details in more elaborate, though more accurate models. In theoretical considerations, usually *memoryless* channel is considered. That means that the time discrete output of the demodulator, depends only on the corresponding signal and not on previous signals. When in addition the input and output signals are from finite alphabet, we a *discrete memoryless channel* (DMC). It is defined by an q-ary set of input symbols x_i, a Q-ary set of output symbols y_j, and a set of conditional probabilities, called *transition probabilities*, which we can write as

$$P(y = y_j | x = x_i) = P(y_j | x_i)$$

where $i = 0, 1, \cdots, q-1$ and $j = 0, 1, \cdots, Q-1$. The description of the channel as memoryless refers to the assumption that the output symbol at any instant of time depends statistically only on the input symbol at that time.

Binary Symmetrical Channel

A *binary symmetrical channel* (BSC) is a special case of a DMC; the input and output alphabet sets consists of the binary elements (0 and 1). The conditional probabilities are symmetric:

$$\begin{array}{llll} P(0|1) = & P(1|0) = & p \\ P(1|1) = & P(0|0) = & 1-p \end{array} \tag{1.1}$$

Equation (1.1) states the channel transition probabilities. That is, given that a channel symbol was transmitted, the probability that is received in error is p, and the probability that is received correctly is $(1-p)$.

The Additive White Gaussian Noise Channel

This thesis is about understanding and designing coupled codes. Due to the complex nature of these codes, it is desirable to use fairly simple channel models which do not introduce additional intricacies that further hinder the understanding. At the same time it is of course desirable to use a model with practical relevance. An attractive compromise between these objectives is the *additive white Gaussian noise* (AWGN) channel.

White Gaussian noise is defined to be a random process, each sample of which is a zero-mean Gaussian random variable and whose power spectral density is flat over the used frequency range, with a level of $N_0/2$ Watts per Hertz. Equivalently, the one-sided noise spectral density is N_0, so that, for example, a rectangular passband filter of width W Hertz will pass N_0W Watts of noise power. The additive white Gaussian noise channel can now be described simply in terms of the input x and the output y, which are related by

$$y = x + n_G$$

where n_G is a zero-mean *Gaussian random variable* with variance σ^2 and independent of the input x. The input x can have any one of M discrete values, where $M \geq 2$. That is, the conditional probability density function of the output y, given an input x, is given by

$$f(y_j|x = x_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{-(y_j - x_i)^2}{2\sigma^2}\right]. \tag{1.2}$$

The AWGN channel is an accurate model for many communication links, such as satellite and deep-space links, in which the dominant effect limiting communication performance is additive thermal or galactic noise.

1.1.2 Channel Capacity

Now let us consider a DMC having an input alphabet $X = x_0, x_1, \cdots, x_{q-1}$ and output alphabet $Y = y_0, y_1, \cdots, y_{Q-1}$, and the set of transition probabilities $P(y_j|x_i)$ as defined in section 1.1.1.

Suppose that the symbol x_i is transmitted and the symbol y_j is received. The mutual information provided about the event $X = x_i$ by the occurrence of the event $Y = y_j$ is $log[P(y_j|x_i)/P(y_j)]$, where

$$P(y_j) = P(Y = y_j) = \sum_{k=0}^{q-1} P(x_k)P(y_j|x_k).$$

Hence, the average mutual information provided by the output Y about the input X is

$$I(X;Y) = \sum_{i=0}^{q-1}\sum_{j=0}^{Q-1} P(x_i)P(y_j|x_i)log\frac{P(y_j|x_i)}{p(y_j)}. \tag{1.3}$$

The channel characteristic determines the transition probabilities $P(y_j|x_i)$ but the probabilities of the input symbols are under the control of the discrete channel encoder. The value of $I(X;Y)$ maximized over the set of input symbol probabilities $P(x_i)$ is a quantity that depends only on the characteristics of the DMC through the conditional probabilities $P(y_j|x_i)$. This quantity is called the *capacity* of the cannel and is denoted by C. That is, the capacity of a DMC is defined as

$$C = \max_{P(x_i)} I(X;Y). \tag{1.4}$$

The maximization of $I(X;Y)$ is performed under the constraints that

$$P(x_i) \geq 0,$$

$$\sum_{i=0}^{q-1} P(x_i) = 1.$$

The Unit of C is bits per input symbol into the channel (bit/channel use) when the logarithm is with base 2, and nats/input symbol when the natural logarithm (base e) is used. If a symbol enters the channel every τ seconds, the channel capacity in bits/s or nats/s in C/τ.

1.1.3 The Optimum Decoder

In this section, we assume, that the data is transmitted block-wise in the code word **c** and **y** is the corresponding observation demodulator output vector. We describe the optimum decision rule based on the observation vector **y**. For this consideration we assume there is no memory in signals transmitted in successive signal intervals. One wishes to design a signal detector that makes a decision on the transmitted signal in each signal interval based on the observation of the vector **y** in the corresponding interval such that the probability of a correct decision is maximized. With this goal in mind, we consider a decision rule based on the computation of the *posterior probabilities* defined as

$$P(signal \ \ \mathbf{c}_m \ \ was \ \ transmitted|\mathbf{y}), \ \ m = 1, 2, \cdots, M$$

which we abbreviate as $P(\mathbf{c}_m|\mathbf{y})$. M is the number of possible code words. The decision criterion is based on selecting the signal corresponding to the maximum in the set of posterior probabilities $\{P(\mathbf{c}_m|\mathbf{y})\}$. This decision criterion is called the *maximum a posteriori probability* (MAP) criterion.

Using Baye's rule, the posterior probabilities may be expressed as

$$P(\mathbf{c}_m|\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{c}_m)P(\mathbf{c}_m)}{P(\mathbf{y})} \tag{1.5}$$

where $P(\mathbf{y}|\mathbf{c}_m)$ is the conditional pdf of the observed vector given $\mathbf{c}_m$, and $P(\mathbf{c}_m)$ is the *a priori probability* of the signal being transmitted.

Some simplification occurs in the MAP criterion when the M signals are equally probable a priori, i.e., $P(\mathbf{c}_m) = 1/M$ for all M. Furthermore, we note that the denominator in (1.5) is independent of which code word is transmitted. Consequently the decision rule based on finding the signal that maximizes $P(\mathbf{c}_m|\mathbf{y})$ is equivalent to finding the signal that maximize $P(\mathbf{y}|\mathbf{c}_m)$.

The conditional pdf $P(\mathbf{y}|\mathbf{c}_m)$ or any monotonic function of it is usually called the *likelihood function*. The decision criterion based on the maximum of $P(\mathbf{y}|\mathbf{c}_m)$ over the M code words is called the *maximum likelihood* (ML) *criterion*. We observe that a detector based on the MAP criterion and one that is based on the ML criterion makes the same decision as long as the a priori probabilities $P(\mathbf{c}_m)$ are all equal, i.e., the code words $\mathbf{c}_m$ are equiprobable.

1.2 Outline of the Thesis

The purpose of channel coding is to establish reliable communication over channels that corrupt transmitted messages with noise and interference. The performance of a specific channel code can be measured in the SNR required to obtain a certain frame- or bit-error rate. Using information theory, it is possible to prove that there exist codes with which essentially error-free communication at rates approaching the channel capacity is possible. However, in practice no codes have been found that perform according to these capacity bounds with a reasonable decoding complexity. The introduction of Turbo codes [11] constitutes a very important step forward, both in the search of good codes as well as in the search of efficient decoding algorithms. This thesis focuses on the coupled codes especially the very effective class of convolutional coupled codes. We start our discussion by giving a short introduction to linear binary block codes and convolutional codes in chapter 2. The basic properties of their structure and distances are defined. The concept of the trellis, which is used several times throughout the thesis, is defined.

In the following chapter we discuss the different forms of concatenated codes. The basic properties of their structure and distances are defined. Chapter 4 provides an introduction to the Turbo coding principles. This includes a description of the different components of a Turbo code

encoder, namely the constituent encoders and the interleaver, as well as to the concept of iterative decoding. The latter includes a brief review of the BCJR-algorithm adopted for iterative decoding.

In chapter 5 convolutional coupled codes are introduced. The generator matrix together with some important structural properties of their generation is derived. Then, we discuss the distance properties of the code, thereby the minimum distance is upper and lower bounded. Moreover, we define a new parameter, called effective free distance, that strongly influences the performance of a convolutional coupled code. The iterative decoding scheme for convolutional coupled codes is described and simulation results are given. Throughout the chapter 5, extensive comparison with parallel concatenated convolutional codes "Turbo codes" are performed, showing that the new scheme can offer superior performance.

In chapter 6 mutual information transfer characteristics of soft in/soft out decoders are proposed as a tool to better understand the convergence behavior of iterative decoding schemes of the convolutional coupled code. The exchange of extrinsic information is visualized as a decoding trajectory in the Extrinsic Information Transfer Chart (EXIT chart). This allows the prediction of turbo cliff position and bit error rate after an arbitrary number of iterations. The influence of code memory, code polynomials as well as different constituent codes on the convergence behavior is studied for convolutional coupled codes.

In chapter 7 we propose design guidelines to find "optimum" convolutional codes for a given memory. At first, the choice of the constituent encoders is considered from a distance property point of view in order to get a large effective free distance. Then, a code search based on the EXIT chart technique was proposed yielding constituent codes exhibiting turbo cliffs at lower signal-to-noise ratios.

In chapter 8, an extended coding scheme is proposed including an outer BCH code correcting a few bit error. This improves the bit error rate (BER) performance of the convolutional coupled code and can be useful in applications where the "error floor" at high signal to noise ratios, is a problem.

Finally, in chapter 9 some concluding remarks and a short discussion on future investigations are given.

Chapter 2

Block and Convolutional Codes

There are two major categories of error correcting codes: block codes and convolutional codes. In this chapter we give a summary of the important definition and properties of linear block codes and convolutional codes which are used in later chapters.

2.1 Linear Binary Block Codes

We assume that the output of an information source is a sequence of binary digits 0 or 1. In block coding, this binary information sequence is segmented into blocks of fixed length. Each information block respectively information word consists of k information digits and is denoted by $\mathbf{u}_t = (u_t^1, u_t^2, \cdots, u_t^k)$ at time t. There is a total of 2^k distinct information words. The encoder maps each information word $\mathbf{u}_t$ into a binary n-tuple, the code word $\mathbf{c}_t = (c_t^1, c_t^2, \cdots, c_t^n)$ with $n \geq k$. The set of all 2^k code words is called a block code or more precisely: Let $I\!F_2$ denote the finite binary field and $I\!F_2^n$ the n-dimensional vector space over the field $I\!F_2$. A block code of rate $R = k/n$ is called a linear binary block code $C(n, k)$ if and only if its 2^k code words form a k-dimensional subspace of the vector space $I\!F_2^n$.

Linearity means that the sum of two code words is again a code word. The information sequence $\mathbf{u}_t$ can be encoded into an output sequence $\mathbf{c}_t$ by a generator $k \times n$ matrix $\mathbf{G}$ of rank k with entries from $I\!F_2$, as follows:

$$\mathbf{c}_t = \mathbf{u}_t \mathbf{G}. \tag{2.1}$$

For each linear block code exists a set of equivalent generator matrices, also called encoding matrices. Each of them generates the same code but with different mappings from information to code words. An encoding matrix is called *systematic encoding matrix* if the information word with unaltered information bits is part of the resulting code word. Without loss in generality, we assume that the first k bits of the code word are the information word. Let $\mathbf{I}_k$ denote the $k \times k$ identity matrix and $\mathbf{P}$ any $k \times (n-k)$ matrix, then a generator matrix $\mathbf{G}$ is systematic if it can be

written as:

$$\mathbf{G}^{'} = [\mathbf{I}_k \mathbf{P}]. \tag{2.2}$$

Associated with any linear (n, k) code is the dual code of dimension $n - k$. The dual code is a linear $(n, n - k)$ code with $2^{(n-k)}$ code vectors, which is the null space of the (n, k) code. The generator matrix for the dual code, denoted by **H**, consists of $n - k$ linearly independent code vectors selected from the null space. Any code word $\mathbf{c}_m$ of the (n, k) code is orthogonal to any code word in the dual code. Hence, any code word of the (n, k) code is orthogonal to every row of the matrix **H**, i.e,

$$\mathbf{c}_m \mathbf{H}^T = \mathbf{0} \tag{2.3}$$

where **0** denotes an all-zero row vector with $n - k$ elements, and $\mathbf{c}_m$ is a code word of the (n, k) code. Since (2.3) holds for every code word of the (n, k) code, it follows that

$$\mathbf{G}\mathbf{H}^T = \mathbf{0} \tag{2.4}$$

where **0** is now a $k \times (n - k)$ matrix with all-zero elements.

Now suppose that the linear (n, k) code is systematic and its generator matrix **G** is given by the systematic form (2.2). Then since $\mathbf{G}\mathbf{H}^T = \mathbf{0}$, it follows that the generator matrix of the dual code can be written as

$$\mathbf{H} = [-\mathbf{P} | \mathbf{I}_{n-k}]. \tag{2.5}$$

The negative sign in (2.5) may be dropped when dealing with binary codes, since modulo-2 substruction is identical to modulo-2 addition.

Example 2.1

A binary $(7, 4)$ Hamming code has for example the following generator and parity check matrices:

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \boldsymbol{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Equivalent systematic matrices $G^{'}$ and $H^{'}$ are given by:

$$\boldsymbol{G}^{'} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad \boldsymbol{H}^{'} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

2.1.1 Distance Properties

The distance properties of a code determine its error-correcting and error detecting capabilities.

The *Hamming weight* of a code word $\mathbf{c}$, denoted $w(\mathbf{c})$, is defined as the number of nonzero components of $\mathbf{c}$. *The Hamming distance*, $d_H(\mathbf{c}_1, \mathbf{c}_2)$, between n-tuple $\mathbf{c}_1$ and $\mathbf{c}_2$ defined as the number of coordinates which $\mathbf{c}_1$ and $\mathbf{c}_2$ differ. The Hamming distance is a *metric* for the set of n-tuples over any non-empty set, i.e. it satisfies the following three axioms for any three n-tuples $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$:

$$\begin{array}{ll} d_H(\mathbf{c}_1, \mathbf{c}_2) \geq 0 \ with \ equality \ if \ and \ only \ if \ \mathbf{c}_1 = \mathbf{c}_2 & (positive \, definiteness) \\ d_H(\mathbf{c}_1, \mathbf{c}_2) = d_H(\mathbf{c}_2, \mathbf{c}_1) & (symmetrie) \\ d_H(\mathbf{c}_1, \mathbf{c}_3) + d_H(\mathbf{c}_3, \mathbf{c}_2) \geq d_H(\mathbf{c}_1, \mathbf{c}_2) & (triangle \ inequality) \end{array}$$

The minimum distance of a code is defined as

$$d_{min} = \min_{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_1 \neq \mathbf{c}_2} d_H(\mathbf{c}_1, \mathbf{c}_2) = \min_{\mathbf{c}_1 \in C, \mathbf{c}_1 \neq \mathbf{0}} w(\mathbf{c}_1) \tag{2.6}$$

2.1.2 Trellis Presentation

Usually, codes are represented through a so-called trellis. A trellis consists of a set S of nodes, one alphabet A and a set of branches (s_t, s_{t+1}, i_t), $s_t, s_{t+1} \in S, i_t \in A$ between the nodes. $i_t \in A$ marked the transfer between two nodes of the stage t and $t+1$. A path of length l is a series of l concatenated branches. Each code word is described through a path of length n, which begins in a node s_0 and ends in a node s_n.

Each code word of a block code must fulfill the equation:

$$\mathbf{c}\mathbf{H}^T = \mathbf{0}. \tag{2.7}$$

Let now $\mathbf{h}_t$ be the t-th column of the $(n-k) \times n$ Matrix $\mathbf{H}$ and c_t be the t-th bit of a code word. Then, the equation (2.7) can also be written as

$$\sum_{t=0}^{n-1} c_t \mathbf{h}_t = \mathbf{0}. \tag{2.8}$$

The set S of nodes is formed from all possible 2^{n-k} binary vectors s_t of length $n-k$, which are called syndromes. Since the considered codes are binary, then $A = I\!F_2$. Starting from the zero-node s_0, one computes the nodes of the $t+1$ stage through

$$s_{t+1} = s_t + i_t \mathbf{h}_t. \tag{2.9}$$

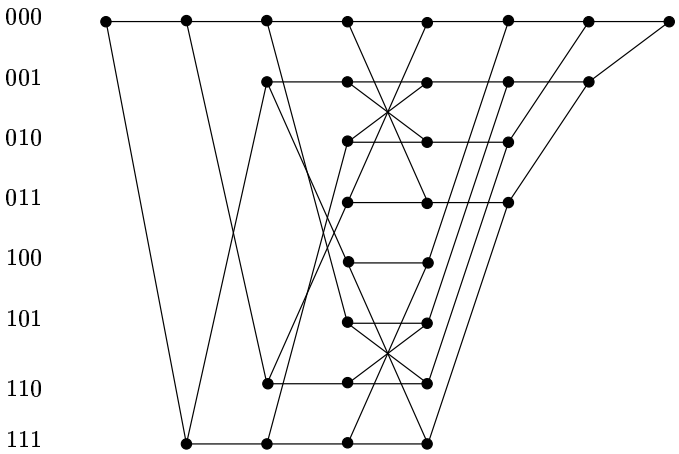

Figure 2.1: Trellis of the systematic $(7, 4)$ Hamming code.

The bit between tow nodes of the stage t and $t+1$ corresponds to the t-th code bit c_t. Since every code word must end in zero-node, all paths which do not end in the zero-node are cleared. Every remaining path corresponds precisely to a code word.

Example 2.2

Consider the Hamming code from the Example 2.1. There are at the most $2^{n-k} = 2^3 = 8$ nodes in every stage. Every transfer between two stages corresponds to one bit from $A = I\!F(2)$. Figure 2.1 represents the syndrome trellis for the parity check matrix $H^{'}$ given in example 2.1.

2.2 Convolutional Codes

Convolutional codes were first introduced by Elias [27] in 1955 as an alternative to block codes. This section only gives a brief introduction to the theory of convolutional codes. For more thorough survey refer to, e.g., [42] and [48].

To grasp the content of this thesis, some knowledge about binary convolutional codes is required. Convolutional codes are the second big class of codes of the error correction besides block codes. They differ from block codes in that the encoder contains memory and the n encoder outputs at any given time unit depend not only on the k inputs at that time unit but also on m previous input blocks. An (n, k, m) convolutional code can be implemented with a k-input, n-output linear sequential circuit with input memory m. Typically, n and k are small integers with $k < n$, but the memory order m should be made large to achieve low error probabilities. Indeed, besides redundancy, memory is an important parameter for the error correction capability of a code. In the important special case when $k = 1$, the information sequence is not divided into blocks and can be processed continuously.

In general, a rate $R = k/n$, convolutional encoder input (*information sequence*) is a sequence of binary k-tuples,

$$\mathbf{u} = \cdots \mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \cdots$$

where $\mathbf{u}_t = (u_t^1, \cdots, u_t^k)$. The output (*code sequence*) is a sequence of binary n-tuples

$$\mathbf{c} = \cdots \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \cdots$$

where $\mathbf{c}_t = (c_t^1, \cdots, c_t^n)$. The sequence must start at a finite (positive or negative) time t and may or may not end. The relation between the information sequence and the code sequence is determined by the equation

$$\mathbf{c} = \mathbf{uG} \tag{2.10}$$

where

$$\mathbf{G} = \begin{pmatrix} G_0 & G_1 & \cdots & G_m & & \\ & G_0 & G_1 & \cdots & G_m & \\ & & G_0 & G_1 & \cdots & G_m \\ & & & \cdots & \cdots & \cdots \end{pmatrix} \tag{2.11}$$

is a semi-infinite generator matrix and where the sub-matrices G_i, $0 \leq i \leq m$, are binary $k \times n$ matrices whose entries are

$$G_i = \begin{pmatrix} g_{1,i}^{(1)} & g_{1,i}^{(2)} & \cdots & g_{1,i}^{(n)} \\ g_{2,i}^{(1)} & g_{2,i}^{(2)} & \cdots & g_{2,i}^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,i}^{(1)} & g_{k,i}^{(2)} & \cdots & g_{k,i}^{(n)} \end{pmatrix}.$$

The arithmetic in (2.10) is carried out over the binary field $I\!F_2$, and the parts left blank in the generator matrix $\mathbf{G}$ are assumed to be filled with zeros.

The right hand side of (2.10) defines a discrete-time convolution between $\mathbf{u}$ and $\mathbf{g} = (G_0, G_1, \cdots, G_m)$, hence the name *convolutional codes*. As in many other situations where convolution appears it is convenient to express the sequences in some sort of transformation. In information theory and coding theory it is common to use the delay operator D, the D-transform. The information and code sequences becomes

$$\mathbf{u}(D) = \cdots + u_o + u_1 D + u_2 D^2 + \cdots$$

and

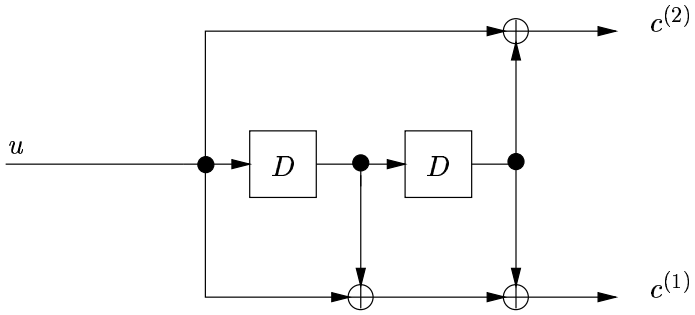

Figure 2.2: An encoder for the generator matrix in 7.1.

$$\mathbf{c}(D) = \cdots + c_0 + c_1 D + c_2 D^2 + \cdots$$

They are related through the equation

$$\mathbf{c}(D) = \mathbf{u}(D)\mathbf{G}(D), \tag{2.12}$$

where

$$\mathbf{G}(D) = \begin{pmatrix} \mathbf{g}_1^{(1)}(D) & \mathbf{g}_1^{(2)}(D) & \cdots & \mathbf{g}_1^{(n)}(D) \\ \mathbf{g}_2^{(1)}(D) & \mathbf{g}_2^{(2)}(D) & \cdots & \mathbf{g}_2^{(n)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{g}_k^{(1)}(D) & \mathbf{g}_k^{(2)}(D) & \cdots & \mathbf{g}_k^{(n)}(D) \end{pmatrix}$$

is the generator matrix. $\mathbf{g}_l^{(o)}(D)$ are the generator polynomials and are given by

$$\mathbf{g}_l^{(o)}(D) = g_{l,0}^{(o)} + g_{l,1}^{(o)} D + g_{l,2}^{(o)} D^2 + \cdots g_{l,m}^{(o)} D^m,$$

for $l = 1, 2, \cdots, k$ and $o = 1, 2, \cdots, n$.

Example 2.3: *Consider the polynomial generator matrix*

$$\boldsymbol{G} = (1 + D + D^2, 1 + D^2) \tag{2.13}$$

An encoder for this generator matrix can be built as in figure 2.2.

An important subclass of convolutional codes is the class of systematic codes. In a systematic code , the first k output sequences are exact replicas of the k input sequences, i.e.,

$$c^i = u^i, \quad i = 1, 2, \cdots, k.$$

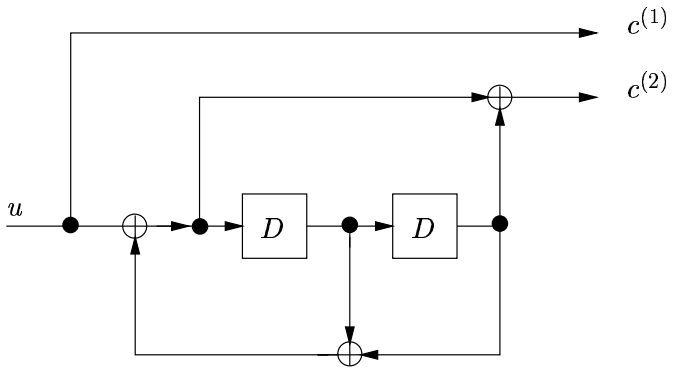

Figure 2.3: realization of a $(2, 1, 2)$ recursive systematic encoder.

Definition 2.2: *The Generator matrix of a rate $R = k/n$ convolutional code is called systematic, if all the k information sequences $\boldsymbol{u}(D)$ appear unchanged in the n code sequences $\boldsymbol{c}(D)$.*

An important class of systematic convolutional codes is the class of *recursive systematic convolutional* (RSC) codes. They play a particular role in the parallel concatenated codes as well as in the convolutional coupled codes. For example, a binary $R = 1/2$ RSC code is obtained from a non systematic convolutional (NSC) code using a feedback loop and setting one of the two outputs c_1 or c_2 equal to the input data bit u_k.

Example 2.4: *Consider the convolutional code of the previous example. The equivalent recursive systematic generator matrix is given by*

$$\boldsymbol{G}_{sys} = \frac{1}{1 + D + D^2}\boldsymbol{G} = (1, \frac{1 + D^2}{1 + D + D^2})$$

An encoder for this generator matrix is presented in figure 2.3.

2.2.1 Finite Code Sequences

In theory, code sequences of convolutional codes are of half infinite length. But for practical application usually finite sequences are used. There are three different methods to obtain finite code sequences:

- **Truncation:** We stop encoding after a certain number of bits without any additional efforts. This leads to high error probabilities for the last bits in a sequence.
- **Termination:** We add some tail bits to the code sequence in order to ensure a predefined end state, which leads to low error probabilities for the last bits in a sequence.

- **Tail biting:** We choose a starting state which ensures that starting and end state are the same. This leads to equal error protection.

In general we prefer termination or tail biting, where tail biting increases the decoding complexity and for termination additional redundancy is required. In this work we will only consider terminated code sequences, where we start encoding in the all-zero encoder state and we ensure that after the encoding process all memory elements contains zeros again. In case of polynomial encoder this can be done by adding km zero bits to the information sequence of length L. As consequence we decrease the code rate

$$R_{ter} = \frac{kL}{n(L+m)} = R\frac{L}{L+m}, \tag{2.14}$$

where $L/(L+m)$ is the so-called *fractional rate loss.*

2.2.2 State Diagram

A convolutional coder has a finite number m of memory and consequently a finite number 2^m of memory states σ. The coder can be realized as a finite state automate. The result c_t of the coder at time t depends from the memory state σ_t and from the information block u_t. At time $t+1$ the coder is in state σ_{t+1}. Each state modification $\sigma_t \rightarrow \sigma_{t+1}$ is associated with a certain input sequence and an output sequence. The graph with all possible states can be presented like in the next example.

Example 2.5: *The $(2,1,2)$ convolutional coder with $m=2$ memory elements of example 2.3 is presented in the figure 2.4. The coder has $2^m = 4$ memory states $\sigma \in \{(00),(01),(10),(11)\}$. For example, the state transition $(10) \rightarrow (01)$, which is labeled with $0/10$, means that the correspondent information block is $u_t = (0)$ and the output code symbol is $c_t = (10)$.*

2.2.3 Trellis

A convolutional code is the set of all possible code sequences encodes by a sequential circuit, the convolutional encoder. A trellis consists of branches and nodes, where each node represents an encoder state and a branch represents a state transition. The branches are labeled with the corresponding input/output block. We order the nodes in rows and each column corresponds to a particular time step. A path in a trellis can be defined as a sequence of branches or nodes which illustrates a valid code word. In figure 2.5 we depict the trellis for the $(2,1,2)$-convolutional code from example 2.3. Here each branch is labeled with the associated output symbols. A solid line represents an input one, a dashed line an input zero. In order to represent code words we only need the labeling of the output symbols. All possible paths in a trellis represent the convolutional code C. But note, the trellis still depends on the encoder. If for a particular encoder among all

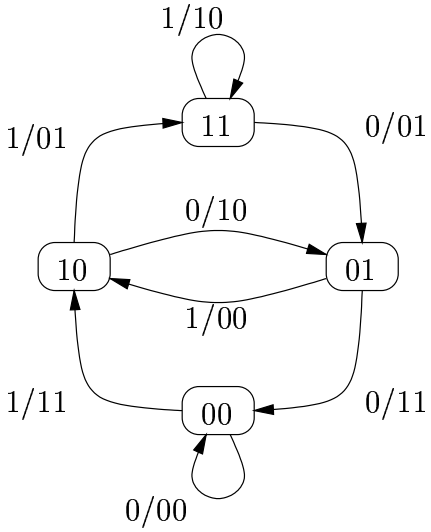

Figure 2.4: State diagram of a $(2, 1, 2)$ convolutional encoder.

possible realizations the number of the physical states is minimal we call the trellis *minimal Viterbi-Forney trellis*.

2.2.4 Distance Properties of Convolutional Codes

The most important distance property of convolutional codes is the free distance [21].

Definition 2.3: *The free distance of a convolutional code C is the minimum Hamming distance between two code sequences,*

$$d_{free} = \min_{\mathbf{c}_1, \mathbf{c}_2 \in C} d_H(\mathbf{c}_1, \mathbf{c}_2). \tag{2.15}$$

Since convolutional codes are linear, all non-zero code sequences can be compared with the all-zero sequence to get the same result

$$d_{free} = \min_{\mathbf{c} \in C, \mathbf{c} \neq \mathbf{0}} w(\mathbf{c}). \tag{2.16}$$

The free distance of a convolutional code is an important parameter for its error correction capability. A minimum distance decoder always corrects an error sequence, **e**, if

$$w(\mathbf{e}) < \frac{d_{free}}{2}. \tag{2.17}$$

Extended row and column distance

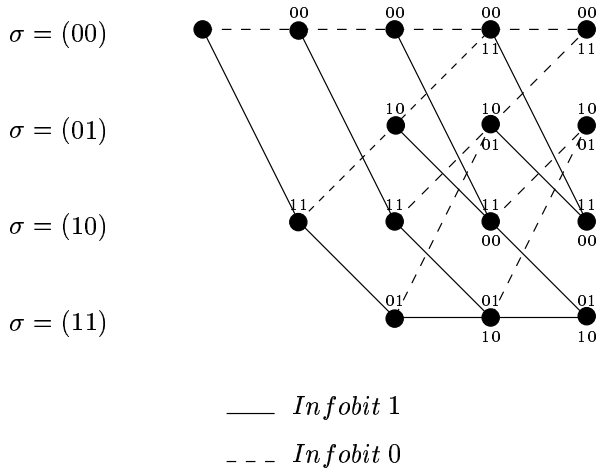

Figure 2.5: Trellis for $(2, 1, 2)$-convolutional code.

The extended row and the extended column distance are important for the description of the properties of convolutional codes. Their definition is introduced in [62] and continued in [25][24].

Definition 2.4: *Let* $\mathbf{0}$ *be the zero state and* σ_i *be the state in stage* i *of the trellis which belongs to the code word* $\mathbf{c} = (c_0, c_1, \cdots) \in C$. c_i *are code blocks of length* n. *Then the extended row distance* $d_r(jn)$ *is defined as*

$$d_r(jn) = \min_{c \in C_j} \sum_{i=0}^{j-1} w(c_i)$$

with

$$C_j = \{c \in C | \sigma_0 = \mathbf{0}, \sigma_i \neq \mathbf{0} \text{ for } 1 \leq i \leq j-1, \sigma_j = \mathbf{0}\}.$$

The extended column distance $d_c(jn)$ *is defined by*

$$d_c(jn) = \min_{c \in C_j} \sum_{i=0}^{j-1} w(c_i)$$

with

$$C_j = \{c \in C | \sigma_0 = \mathbf{0}, \sigma_i \neq \mathbf{0} \text{ für } 1 \leq i \leq j-1\}.$$

$w(c_i)$ *is the Hamming weight of the encoder output* c_i *of length* n.

The extended distances increase with path length j. One property of the extended row distance $d_r(jn) + d_r(jn) \geq d_r(in + jn) \forall i, j$ (triangle inequality). This property is valid for all codes of practical interest [26].

Remark 2.1: *In the definition of the extended distances, the code sequence between the first and the end stage in the trellis must not pass through the zero state. If the code sequence may "touch" the all-zero path in non-consecutive zero states, then the extended distances are called active distances. The triangle inequality for the active distances is still fulfilled [13].*

Chapter 3

Concatenated Codes

Because coupled codes are a kind of concatenated codes, first the basic of concatenated constructions should be understood.

In his goal to find a class of codes whose probability of error decreased exponentially with code length, while decoding complexity increased only algebraically, Forney [28] arrived at a solution consisting of the multilevel coding structure known as concatenated code. It consists of a cascade of an inner and an outer code, which, in Forney's approach, would be a relatively short inner code (typically a convolutional code) admitting simple maximum likelihood decoding, and a long high-rate algebraic non binary Reed-Solomon outer code equipped with a powerful algebraic error-correction algorithm, possibly using reliability information from the inner decoder. Initially motivated only by theoretical research interests, concatenated codes have since then involved as a standard for those applications where very high coding gains are needed, such as (deep-) space applications and many others. Alternative solutions for concatenation have also been studied, such as using a trellis-coded modulation scheme as inner code [23], or two concatenated convolutional codes [36]. An interleaver was also proposed between the two encoders to separate bursts for errors produced by the inner decoder. In this chapter two types of concatenated codes are considered: serial and parallel concatenated codes.

3.1 Serial Concatenated Codes

In a serial concatenated code, the codes are arranged serially one after another. Each code encoded the entire data stream inclusive the already generated redundancy bits. In the case of two component codes, one speaks of an *inner* code and an *outer* code.

Example 3.1: *By means of a very simple example we want to grasp the concatenation of two codes. At first, we consider a simple* $(3, 2, 2)$*-SPC Code (Single Parity Check), which should be linked with a* $(4, 3, 2)$*-SPC code. The code rate of the concatenated code amounts to* $R_c = 2/4 = 1/2$*. The minimum distance will be discussed below.*

u	$\mathbf{c}_1$	$\mathbf{c}_2$	$w(\mathbf{c}_2)$
00	000	0000	0
01	011	0110	2
10	101	1010	2
11	110	1100	2

Table 3.1: Code words of the concatenation of outer $(3, 2, 2)$-SPC code and inner $(4, 3, 2)$-SPC code.

SPC-codes append a single parity bit to the information word ***u****. Thus, they have the minimum distance* $d_{min} = 2$*. Now, which minimum distance has the concatenated code? Intuitively, one suspects a minimum distance greater than* 2*. For this simple example, the weight spectrum is given in Table 3.1. Since the code is linear, it is sufficient to consider the weights of the code words and not the distances among each other. It is apparent, that the minimum distance of the concatenated code still is* 2*. Obviously, a code concatenation does not lead automatically to an improvement of the distance properties.*

Now the following question came up: How must the serial code concatenation generally be constructed, so that the concatenated code has optimum distance properties? A skillful code concatenation are product codes.

3.1.1 Product Codes

Let $C^|$ be a $(n^|, k^|)$ linear code and let C^- be a (n^-, k^-) linear code. Then a $(n^|n^-, k^|k^-)$ linear code can be formed such that each code word is a rectangular array of $n^|$ columns and n^- rows in which every row is a code vector in $C^|$ and every column is a code vector in C^-, as shown in figure3.1. This two-dimensional code is called the direct product (or simply the product) of $C^|$ and C^- [45]. The $k^|k^-$ digits in the upper left corner of the array are information symbols. The digits in the upper right corner of this array are computed from the parity-check rules for $C^|$ on rows, and the digits of the lower left corner are computed from the parity-check rules for C^- on columns. Now, should we compute the check digits in the lower right corner by using the parity-check rules for C^- on columns or the parity-check rules for $C^|$ on rows? It turns out that either way yield the same $(n^| - k^|) \times (n^- - k^-)$ check bits, and it is possible to have all row code vectors in $C^|$ and all column vectors in C^- simultaneously. Therefore, for encoding the product code $C^| \times C^-$, we may first encode the k^- rows of the information array based on the parity-check rules for $C^|$ and then encode the $n^|$ resulting columns based on the rules for C^-, or vice versa.

If the code $C^|$ has minimum weight $d^|_{min}$ and the code C^- has minimum weight d^-_{min}, the minimum weight of the product code is exactly $d^|_{min}d^-_{min}$. A minimum-weight code vector in the product code is formed by (1) choosing a minimum-weight code vector in $C^|$ and a minimum-weight code vector in C^-; and (2) forming an array in which all columns corresponding to zeros

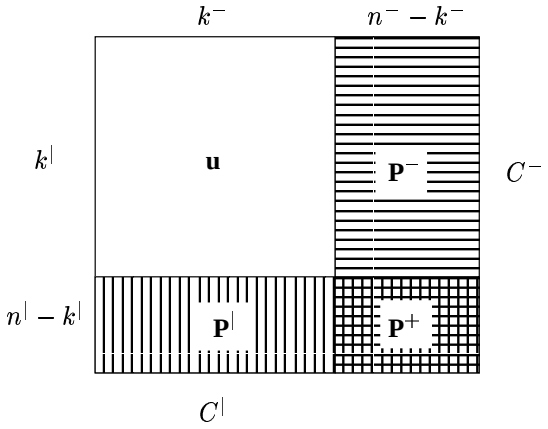

Figure 3.1: Code array for the product code $(C^| \times C^-)$.

x_0	x_4	x_8
x_1	x_5	x_9
x_2	x_6	x_{10}
x_3	x_7	x_{11}

Figure 3.2: $(12, 6, 4)$ product code with component codes $(3, 2, 2)$ and $(4, 3, 2)$-SPC codes.

in the code vector from $C^|$ are zeros and all columns corresponding to ones in the code vector $C^|$ are the minimum-weight code vector chosen from C^-.

The code rate of the product code is given by

$$R_c = \frac{k^| \cdot k^-}{n^| \cdot n^-} = R^| \cdot R^- \tag{3.1}$$

and it is made up of the product of the code rates of $C^|$ and C^-. Assumed that $R^| = R^- = R_k$, then

$$R_c = R^| \cdot R^- = R_k^2.$$

The product code in figure 3.1 holds inherently a block interleaver.

Example 3.2: *We construct a* $(12, 6, 4)$*-Product code from the already known* $(3, 2, 2)$*- and* $(4, 3, 2)$*-SPC codes (see figure 3.2). The code rate of the product code* $R_c = 6/12 = 1/2$ *is not*

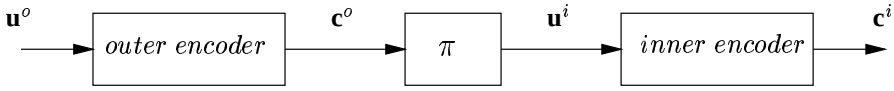

Figure 3.3: Encoder scheme for serial concatenated convolutional codes with interleaving.

different compared with this of the concatenated code without interleaving. In fact, the difference consists rather therein, that through the interleaving, 3 *information words yield a concatenated code word. Thus, the block length of the concatenated code without spreading grows from* $n = 4$ *to* $n = 12$*, which - as we now - is a great advantage for block codes. This leads to the following improvement. The two component codes hold minimum distance of* 2 *and thus can only detect one error, but no error correction. The product code increases the minimum distance to* 4*, so that one error can be corrected or three error can be detected. The error correction is illustrated in figure 3.2. Is the symbol* x_1 *incorrect, then respond both the parity-check rule of the* 2*th row and the parity-check rule of the* 1*st column.*

3.1.2 Choice of the Component Codes

Of course, not only block codes can be linked together to construct serial concatenated codes. The same method can be used with two or more convolutional codes as well as with a combination of block and convolutional codes. But, which code one selects as inner code and which code one selects as outer code? The answer to this question depends on many factors. For example, with respect to the distance spectrum it is advantageous, if the outer code holds a free distance as large as possible. According to this, the stronger code must be deployed as outer code [7].

In the other hand we will show, that due to the technical feasibility concatenated codes can not be decoded with the Maximum Likelihood criterion. Rather, the iterative decoding described in section 3.1.3 performs very close to ML-decoding. For this, it is more favorable to deploy the stronger code as inner code, since this will be decoded at first and thus delivers a better starting basis for the decoding of the outer codes. Usually the inner code is a rate $1/2$ code and is relatively short. The outer code is much longer and has much higher code rate.

3.1.3 Iterative Decoding of Serially Concatenated Codes

We will now consider the decoding of serial concatenated code with interleaving as introduced in [7]. The encoding scheme as depicted in figure 3.3 is quite simple. The encoder consists of the cascade of an outer encoder, an interleaver permuting the outer code word, and an inner encoder encoding the permuted outer code bits.

An iterative decoding algorithm can be used for decoding long codes obtained from concatenated codes. Here decoding is split into several steps and symbol-by-symbol reliability information is repeatedly transferred between the decoding steps. In each decoding step, we have soft input

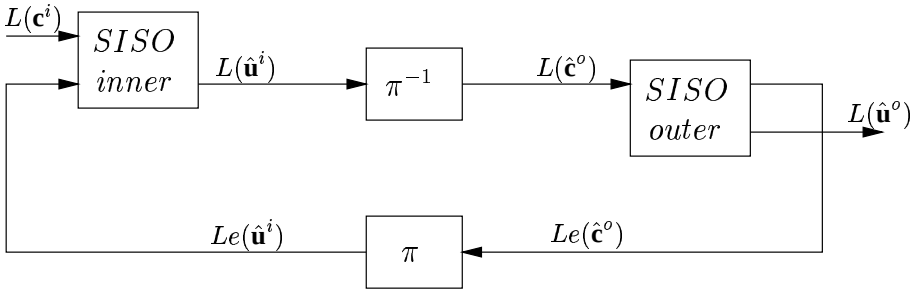

Figure 3.4: Decoding scheme for serial concatenated codes with interleaving.

reliabilities and soft output reliabilities (SISO). The soft output information of each information symbol consists of three parts: Reliabilities from the channel, the a-priori knowledge, and the so-called extrinsic part from code correlations. Only the extrinsic values should be used to gain the new a-priori values for the next iteration [11]. The complexity of an iterative algorithm depends on the complexity of the SISO-decoding of the component codes. The low complexity of SISO-decoding of convolutional codes is a main reason for using convolutional codes in concatenated systems. The decoding scheme for a serial concatenated code is presented in figure 3.4. The symbols L at the input and output ports of the SISO refers to the logarithmic likelihood ratio (LLRs). We define the LLR $L(\hat{u})$ as the real number

$$L(\hat{u}) = ln\frac{P(u=1|observation)}{P(u=0|observation)} \tag{3.2}$$

where u is in $I\!F(2)$ and $P(u|observation),\ i=0,1$ is the posteriori probability of the bit u. The LLR of a vector $\mathbf{u} = (u_1, u_2, \cdots)$ is defined as $L(\mathbf{u}) = (L(u_1), L(u_2), \cdots)$.

The decoder for the inner code generates reliabilities $L(\hat{\mathbf{u}}^i)$ which represent the permuted channel values of the outer decoder. The outer decoder calculates L-values for the information word $L(\hat{\mathbf{u}}^o)$ needed for the final decision and additionally provides extrinsic values $Le(\hat{\mathbf{c}}^o)$ for outer code bits. These extrinsic values are interleaved and treated as a-priori information for the next decoding step.

3.2 Parallel Concatenated codes

In figure 3.5 we see the code word of a parallel concatenation of two block codes. Both component codes $C^|(n^|, k^|)$ and (n^-, k^-) are encoded by a systematic generator matrix $\mathbf{G}^|$ respectively $\mathbf{G}^-$:

$$\mathbf{G}^i = [\mathbf{I}_k \mathbf{A}_i].$$

In a parallel concatenated code, the component codes are arranged parallelly. Each component coder gets only the information bits, not the redundancy bits of the other encoders. The out-

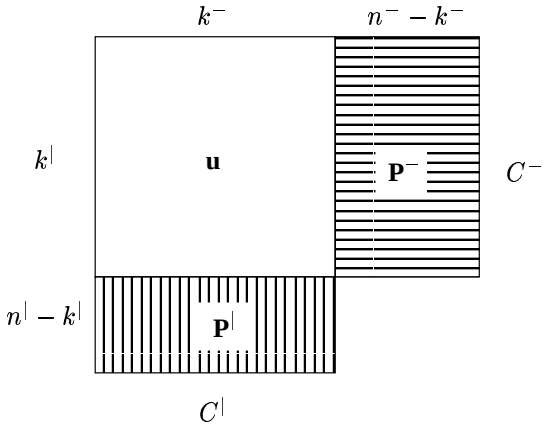

Figure 3.5: Code word of a Parallel concatenated code.

put sequences of the component encoder are jointed to a data stream through a parallel-serial-converter.

The difference to the serial concatenated codes consists in the fact, that exclusively the information bits are encoded by the component codes, not the respective parity bits. We can transform the product code depicted in figure 3.1 in a parallel concatenated code, when we exclude the parity bits $\mathbf{P}^+$. This results the structure of figure 3.5. One speaks in this connection of an uncompleted product code. The parallel concatenated code has the code rate

$$\begin{aligned} R_c^p &= \frac{k^- \cdot k^|}{k^- \cdot k^| + (n^- - k^-) \cdot k^| + (n^| - k^|) \cdot k^-} \\ &= \frac{1}{\frac{1}{R_c^|} + \frac{1}{R_c^-} - 1} = \frac{R_c^| \cdot R_c^-}{R_c^- + R_c^| - R_c^- \cdot R_c^|}. \end{aligned} \tag{3.3}$$

Assume, that the two component codes are identical: $R^- = R^| = R_k$, then the code rate of the over-all code can be given by:

$$R_c^p = \frac{R_k}{2 - R_k}.$$

For the minimum distance of the parallel concatenation we obtain:

$$d_{min} = d_{min}^| + d_{min}^- - 1. \tag{3.4}$$

Equation (3.4) can be explained as follows. Assume that the information array **u** contains only one row, which has exactly one non-zero element and all other rows should contain only zeros.

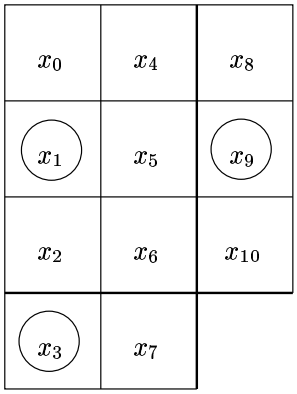

Figure 3.6: $(11, 6, 3)$ parallel concatenated code composed of $(3, 2, 2)$ and $(4, 3, 2)$-SPC code.

Moreover, the corresponding code word of this row should have minimum weight d^{-}_{min}. The vertical coding with $C^{|}$ generates only one code word unequal zero, since the parity bits are no longer encoded. This code word has a minimum weight of $d^{|}_{min}$. The over-all weight results from the sum of the two weights. However, one must take into account that the non-zero information bit, occurs in the two code words but is transmitted only once. The outcome of this is equation (3.4).

Example 3.3: *Now we want to apply the previously considered examples 3.1 and 3.2 for parallel concatenation. From the two SPC-codes we construct a* $(11, 6, 3)$*-Product code (see figure 3.6). The code rate* $R_c = 6/11$ *shows only a little change compared to the complete product code. But, according to equation (3.4), the minimum distance is reduced from* 4 *to* 3*. So still* 1 *error can be corrected, but can detect only* 2 *errors.*

The encircled elements in figure 3.6 indicate the binary digits equal to 1 *for a possible minimum weight code word.*

3.2.1 The Code Rate Gain of Parallel Concatenated Codes

For a coded transmission system, the rate loss resulting from increasing the number of transmitted symbols in dB is given by

$$G^{-}(R) = 10log_{10}\frac{1}{R}[dB], \tag{3.5}$$

where R is the code rate of the system. The rate loss describes the decrease of the emanated energy per symbol, which occurs at a coded transmission compared with an uncoded one. From this it follows, that the rate loss represents a horizontal shift in the bit error curves as function of E_b/N_0.

The code rates of serial and parallel concatenated codes are given in (3.1) and (3.3), respectively. In the case, that all component codes have the same code rate R_k, then we can compare the code

rate of the serial and parallel concatenated codes through the calculation of their ratio

$$\frac{R_c^p}{R_c^s} = \frac{\frac{R_k}{2-R_k}}{R_k^2} = \frac{1}{R_k(2-R_k)}. \tag{3.6}$$

From (3.5) and (3.6) we get the code rate gain of the parallel concatenated codes [50]

$$R_{gain} = -10log_{10}(R_k(2-R_k)). \tag{3.7}$$

From (3.7), we note that the higher the code rate of the component code, the less the gain in code rate of parallel concatenated code, as expected. The major disadvantage of the parallel concatenated code is the loss in the minimum distance. It is only $2d_{min}-1$ compared to d_{min}^2 for the product code.

Chapter 4

Turbo Codes

Turbo codes where introduced by Berrou, Glavieux and Thitimajshima in their famous paper "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes" [11]. The paper presented several epoch-making ideas and results to the field of channel coding - results at first looked upon with scepticism and doubt in the coding community, but today widely accepted. In fact, the paper presented results of communication over the AWGN channel less than 1 dB above the power limit predicted by Shannon. Especially, it was shown that communication is possible at SNRs for which the cutoff rate is lower than the code rate, a limit for a long time considered the practical limit for reliable communication.

Berrou *et al*. made important contributions to both the problem of choosing codes/encoders, and the problem of efficient decoding. The basic principle of the Turbo coding concept is illustrated in figure 4.1. In short, the same message is encoded in two different ways, by encoder 1 and encoder 2. The decoder is correspondingly divided into two separate decoders, where each decoder decodes its part of the concatenated code word. By the use of sophisticated algorithms, the decoder can exchange information on their decoding results and thereby cooperate in finding the correct code word. The term "Turbo" actually reflects the iterative decoding associated with Turbo codes. The authors compared the process of using the output from one unit as input in the next, over and over again, with the functionality of a Turbo combustion engine. Hence, the term "Turbo" is indicative on the decoding method rather than the code selection and the term "Turbo decoding" is sometimes used as a synonym for iterative decoding.

Apart from the important work Berrou *et al*. made on the concept of iterative decoding, they

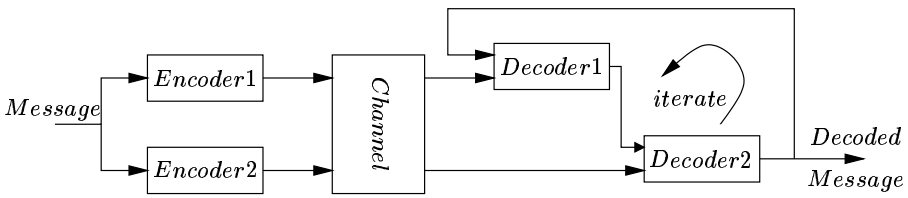

Figure 4.1: The Turbo coding/decoding principle.

proposed a new code construction: parallel concatenation of recursive convolutional encoders. Moreover, there is no clean-cut definition of what Turbo codes are. For example, it is possible to exchange the constituent codes with any other type of code, for example block codes. Such construction is called Block Turbo Codes [53]. Further, it is possible to concatenate several constituent codes by arranging additional encoders in parallel [41]. Such construction are still denoted Turbo codes, or sometimes *multiple Turbo codes*. In the investigations reported in this thesis, we are referring to the original encoder structure presented by Berrou *et al*. that is, parallel concatenation of two recursive systematic convolutional encoders.

4.1 Encoder Structure

Figure 4.2 presents a general version of a binary parallel concatenated Turbo encoder. The encoder is composed of three primary components: the constituent encoders, the interleaver, and a puncturing and multiplexing unit. The constituent encoders are terminated *recursive systematic convolutional* (RSC) encoders. The interleaver (denoted π) is a device that permutes the data sequence in some predetermined manner.

4.1.1 The Constituent Encoders

The constituent encoders are RSC encoders, i.e. systematic convolutional encoders with feedback. RSC encoders are used, instead of the more traditional feed-forward (FF) convolutional encoders, for many reasons. The first reason, that RSC encoders are systematic encoders is the following consideration. When all of the outputs of the constituent encoders are considered, the systematic outputs of the encoders are the same within a permutation. Since sending all of these outputs is nothing more than a repetition code, we may puncture the systematic outputs of all of the constituent encoders and send the information packet as indicated in figure 4.2 without loosing significant performance or suffering from the bandwidth expansion that would result from sending all the systematic outputs.

An other reason that RSC encoders are used in the Turbo encoder deals with the recursive nature of the encoders and the desire to increase the free distance of the over-all system. Since a non-zero code word of a terminated recursive systematic code has at least two ones [8], it is hoped that it is possible to select an interleaver that causes short merges in one trellis to become longer merges in another trellis and cause the free distance, and hence performance of the system to increase. This means that a packet with a single '1' is no longer a limiting case on the free distance of the turbo encoder.

In [8], it was shown that the error probability of a parallel concatenated convolutional code with large interleaving length is proportional to $K^{1-w_{min}}$ where K is the interleaver length and w_{min} is the minimum number of information bits in a non-zero code word. All terminated recursive convolutional encoders have $w_{min} = 2$, so that the interleaving gain goes as $1/K$. On the other hand non recursive convolutional encoders and block codes have $w_{min} = 1$, so such codes are

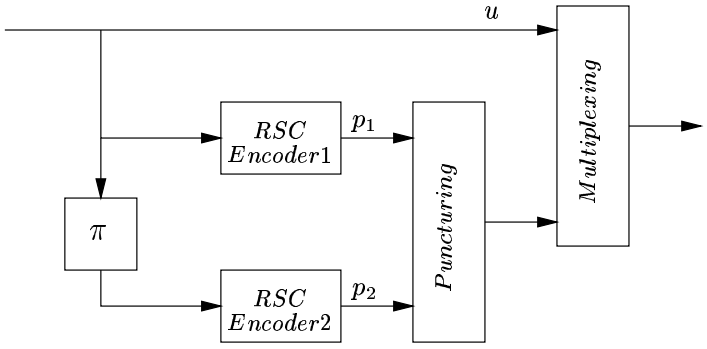

Figure 4.2: Block diagram of a Turbo code encoder.

not useful in parallel concatenated codes.

Moreover, for a large range of SNRs, the behavior of the parallel concatenated convolutional code is determined by the effective free distance

$$d_{free,eff} = w_{min} + 2c_{min}. \tag{4.1}$$

where c_{min} is the lowest weight of the parity-check bits in a code word of the CC generated by information sequences of weight w_{min}. Since w_{min} is equal to 2, it follows, that the constituent encoders must maximize the weight c_{min} of the parity-check bits for $w_{min} = 2$.

4.1.2 Interleaving

The interleaving performed on the information sequence before it is fed to the second constituent encoder constitutes a re-ordering of the information symbols. The combination of two recursive encoders and the interleaver provides a solution to two important issues associated with coding: (1) the creation of codes with good distance properties which (2) can be efficiently decoded, through iterative decoding. The influence on the distance spectra is discussed in [52]. It was shown that the use of systematic feedback encoders and certain interleavers results in *spectral thinning*, in which information sequences which generate a low weight parity sequence from the first constituent encoder are interleaved with high probability to information sequences that generate high-weight parity sequences in the second constituent encoder. The reason that the interleaver enables the use of iterative decoding is that it de-correlates nearby[1] decoder inputs. This is essential, since nearby decoder inputs that are correlated have a detrimental influence on the decoding performance. In this context, the interleaver has often been explained as reducing the correlation between the parity bits corresponding to the original and interleaved data frames.

In the original paper introducing Turbo codes, Berrou *et al.* already showed an exceptional understanding of some of the most important parameters making a good interleaver. In particular:

[1]With nearby is meant position that are close to each other in the input sequence.

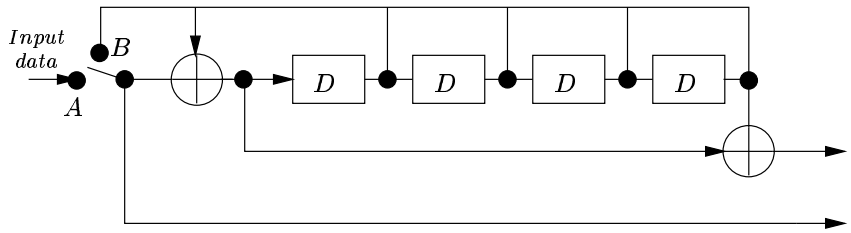

Figure 4.3: Trellis termination.

- The performance of a Turbo code is improved when the interleaver size is increased, which has a positive influence on both the code properties and the iterative decoding performance.
- The interleaver should randomize the input sequence in order to avoid particular low-weight patterns mapping onto themselves, reducing the effective free distance of the resulting Turbo code.

In general, the interleaver can process a continuous input stream. However, throughout this thesis we view the input to the interleaver as divided into blocks. The re-ordering is then performed within each such block.

There are many possibilities of representing the interleaver rule. Among the most common is the use of a vector π, containing the positions that each input symbol holds after interleaving. Thus $\pi(i)$ is the position that input bit i is interleaved to. Equivalently, the deinterleaving rule can be represented by the vector π^{-1}, where $\pi^{-1}(i)$ holds the input position that is interleaved to position i.

4.1.3 Trellis Termination

As mentioned in section 2.2.1, there are three methods to make a code sequence of a convolutional code finite: truncation, termination and tail biting. In the majority of cases, trellis termination is applied in Turbo codes. Since the component encoders are recursive, it is not sufficient to set the last m information bits to zero in order to drive the encoder to the all-zero state, i.e., to terminate the trellis. Thereby m is the number of memory elements in the convolutional encoder. The termination (tail) sequence depends on the state of each component encoder after K information bits, which makes it impossible to terminate both component encoders with the same m bits. Fortunately, the simple stratagem illustrated in figure 4.3 is sufficient to terminate the trellis. Here the switch is in position "A" for the first K clock cycles and is in position "B" for m additional cycles, which flush the encoders with zeros. The decoder is not assumed to have knowledge of the m tail bits. The same termination method can be used for unequal rate and unequal memory encoders.

Additional results regarding trellis termination and tail biting for Turbo codes can be found in [3][12] and [1], respectively.

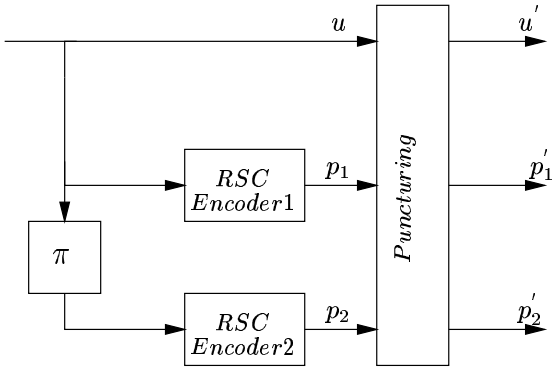

Figure 4.4: Encoder of the PSTC

4.1.4 Puncturing

Puncturing is the process of removing certain symbols/positions from the code word, thereby reducing the code word length and increasing the over-all code rate. In the original Turbo code proposal, Berrou *et al*. punctured half the bits from each rate $1/2$ constituent encoder. However, the systematic bits are the same for both constituent encoders, except for the re-ordering caused by the interleaver. Thus, puncturing half the systematic bits from each constituent encoder corresponds to sending all the systematic bits once, if the puncturing is properly performed.

In the original Turbo code proposal, the over-all code rate is $R = 1/2$. The most common alternative to this is not to puncture the parity bits of either constituent encoder, which result in a Turbo code with rate $1/3$. This is an appealing approach for investigations regarding interleaver design and the choice of constituent encoders, since it removes the ambiguousness resulting from the various possibilities of performing the puncturing. Further, puncturing may have different effects on different choices of interleavers, and on different constituent encoders.

An other method to generate rate $1/2$ Turbo codes was proposed in [47]. Here, the puncturing may be optimized with respect to the parity and systematic bits resulting in the *partially systematic Turbo codes* (PSTC). The PSTC outperform the conventional systematic Turbo codes in the error-floor region. It was proven by analysis of the weight distribution that the PSTCs become the stronger, the more systematic bits are punctured. The encoder of the PSTC is depicted in figure 4.4. It consists of the interleaver, two RSC encoders, and the puncturer. Since convolutional coupled codes are non-systematic codes, the PSTC are of special interest in this work.

4.2 Distance Spectra and Union Bound

The computational complexity of calculating the distance spectrum of a Turbo code is considerable, also for small interleavers. Therefore, one is often confined to computing only the lower

part of the distance spectrum. Fortunately, this is still useful for performance estimation since, at moderate to high SNRs, a large part of the decoding errors are made to code words at distances corresponding to the lower part of the distance spectrum.

A major step forward in the understanding of the Turbo code concept, both for analysis and design, was taken by Benedetto *et al.* in [9][10]. There a method was derived with which it is feasible to calculate the average distance spectrum of Turbo codes using a specific constituent encoder and a certain interleaver length K, where the average is taken over the ensemble of all interleavers. The presented method made possible several important observations regarding the unexplained and unmatched performance of Turbo codes. In the following we will summarize the method as proposed in [10]. Let

$$T^C(X,Y) = \sum_{w=0}^{K} \sum_{j=0}^{N-K} A_{w,j} X^w Y^j$$

be the *input-redundancy weight enumerating function* (IRWEF) of a constituent code (N, K), where $A_{w,j}$ denotes the number of code words generated by an input information word of Hamming weight w whose parity check bits have Hamming weight j, so that the over-all Hamming weight is $w + j$. Consider now the parallel concatenated code obtained as follows. Two linear systematic codes C_1 with parameters (N_1, K) and C_2 with parameters (N_2, K), the constituent codes (CC), having in common the number K of the input information bits, are linked through an interleaver so that the information part of the second code word is just a permuted version of the first one. The parallel concatenated code, that we note as C_p, is then a $(N_1 + N_2 - K, K)$ linear code as the interleaver performs a linear operation on the input bits.

If w is the Hamming weight of the input word, and z_1 and z_2 are the weights of the parity check bits introduced by the first and the second encoders, respectively, the weight of the corresponding code word of C_p will be $w + z_1 + z_2$.

We want now to obtain the IRWEF $T^{C_p}(X,Y)$ of C_p starting from the knowledge of the constituent codes. For a given interleaver, this operation is exceedingly complicated, as the redundant bits generated by the second encoder will not only depend on the weight of the input word, but also on how its bits have been permuted by the interleaver. The only feasible solution, in theory, would be an exhaustive enumeration for all possible cases; in practice this is impossible for large K, and this was precisely the reason for lengthy computer simulations. To overcome this difficulty, an abstract interleaver was introduced called *uniform interleaver*, defined as follows [10].

Definition 4.1: *A uniform interleaver of length K is a probabilistic device which maps a given input word of weight w into all distinct* $\binom{K}{w}$ *permutations of it with equal probability* $1/\binom{K}{w}$.

Consider now the conditional weight enumerating function $T_w^C(Y)$ of the parity-check bits generated by code C corresponding to the input words of weight w. It can be obtained from the

IRWEF as

$$T_w^C(Y) = \sum_j A_{w,j}^C Y^j = \frac{1}{w!} \cdot \frac{d^w T^C(X,Y)}{dX^w}|_{X=0}$$

so that we can also write the inverse relationship

$$T^C(X,Y) = \sum_w X^w T_w^C(Y). \tag{4.2}$$

From the definition 4.1, it is apparent that the conditional weight enumerating function $T_w^{C_2}(Y)$ of the second code is independent from that of the first code thanks to the uniform randomization produced by the interleaver. As a nice consequence of this, we can easily evaluate the conditional weight enumerating function of the parallel concatenated code which uses the uniform interleaver as the product, suitably normalized, of the two conditional weight enumerating function of the constituent codes

$$T_w^{C_p}(Y) = \frac{T_w^{C_1}(Y) \cdot T_w^{C_2}(Y)}{\left(\begin{array}{c} K \\ w \end{array} \right)}. \tag{4.3}$$

$A_{w,j}^{C_p}$ are then given by

$$A_{w,j}^{C_p} = \sum_{j=j_1+j_2} \frac{A_{w,j_1}^{C_1} \cdot A_{w,j_2}^{C_2}}{\left(\begin{array}{c} K \\ w \end{array} \right)}.$$

Also, we obtain the IRWEF of the code C_p as

$$T^{C_p}(X,Y) = \sum_{w=1}^{K} X^w T_w^{C_p}(Y). \tag{4.4}$$

Finally, the union bound on the bit error rate (BER) of the $(N_1 + N_2 - K, K)$ concatenated code C_p is given by

$$P_b \leq \frac{1}{2} \sum_{w=0}^{K} \sum_{j=0}^{N_1+N_2-K} \frac{w}{K} A_{w,j}^{C_p} \cdot erfc(\sqrt{(w+j)\frac{R_c E_b}{N_0}}), \tag{4.5}$$

where R_c is the rate for the concatenated code C_p and E_B/N_0 is the bit signal-to-noise ratio.

$erfc(x)$ is the complementary error function defined as

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-\mu^2} d\mu.$$

In (4.5), we have assumed Binary Phase Shift Keying (BPSK) modulation, and AWGN channel and soft-decision ML-decoding.

4.3 Iterative Decoding

One of the novel attributes of Turbo codes is their ability to compose large codes that can be decoded with reasonably low complexity. As mentioned earlier, this is achieved by iteratively decoding the two constituent codes that together compose the Turbo code. A block diagram of an iterative decoder is shown in figure 4.5. It consists of two constituent decoders, one for each constituent code, and the interleaving/deinterleaving blocks required to convert the sequences between the code spaces.

Each block processes input blocks of size K, i.e. the size of the interleaver. After the first decoder has performed its decoding using the received channel symbols associated with the first code, it passes a block of length K of soft informations to the second constituent decoder. Next, the second decoder uses the information from the first decoder together with the received channel symbols associated with the second code. Hopefully, the second decoder performs better than the first, since it has access to more information. Further, if the first decoder is presented with the result from the second decoder one can imagine that it might improve its performance, compared to its first decoding attempt. Thus, in the second decoding round of the first decoder, it uses the same channel information as in the first round, together with the information passed from the second decoder.

One decoding iteration is completed after one pass of both the first and the second constituent decoders. The decoding performed by one constituent decoder is referred to as a half iteration.

Ideally, the information passed between the constituent decoders should consist of prior knowledge of the probability distribution of each bit in the information sequence. As such, the decoder inputs used as a priori information should depend only on the transmitted information sequence, and not on the noise on the other decoder inputs. By using decoders producing a posteriori probabilities, so called APP- or soft-output decoders, the a posteriori probabilities after decoding the first constituent code can be used as a priori input when decoding the second constituent code.

In a first description, consider an isolated APP decoder whose input a priori information is based on the a priori probabilities $P(u_i = 1)$ and $P(u_i = 0)$, where u_i is the ith information symbol. For binary information symbols, it is convenient to represent the a priori information as log-likelihood ratio (LLR). Thus, denoting the ith a priori input $La(u_i)$, we have

$$La(u_i) = ln \frac{P(u_i = 1)}{P(u_i = 0)}. \tag{4.6}$$

The input to each decoder consists of three sequences, for the rate-$1/2$ constituent encoders considered. These inputs are: the received systematic sequence **x**, the received parity sequence, **y**

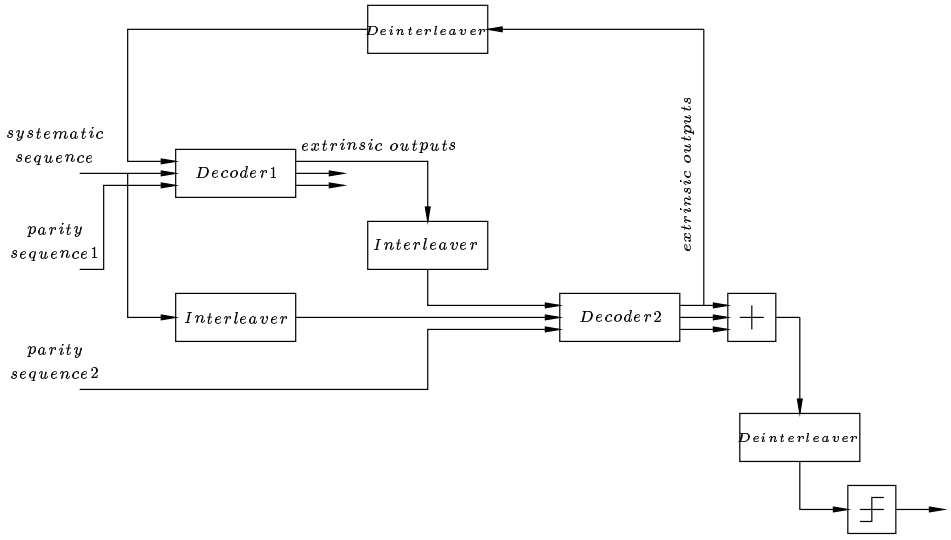

Figure 4.5: Block diagram of an iterative decoder.

and the a priori input sequence, $La(\mathbf{u})$. These sequences are in short referred to as **R**, that is,

$$\mathbf{R} = (\mathbf{x}, \mathbf{y}, La(\mathbf{u})). \tag{4.7}$$

The task of an APP decoder is to derive the a posteriori probabilities $P(u_i = 1|\mathbf{R})$ and $P(u_i = 0|\mathbf{R})$. Similar to the a priori input, the decoder soft-output, denoted $L(\hat{u}_i)$, is represented as an LLR

$$L(\hat{u}_i) = ln\frac{P(u_i = 1|\mathbf{R})}{P(u_i = 0|\mathbf{R})} \quad i = 1, 2, \cdots K. \tag{4.8}$$

The decoder hard decision is achieved by using the sign of the APP soft-output; if $L(\hat{u}_i) > 0$, the decoder hard decision is $\hat{u}_i = 1$, and vice versa.

Let us now return to the iterative decoding environment. It is very important that the information passed between the constituent decoders is properly composed. In particular, since the two decoders are linked in a loop, care must be taken so that instability is avoided. Therefore, it is only a specific portion of the decoder soft-output that should be passed to the next constituent decoder. Assuming that the encoder inputs are independent, the LLR (4.8) can be partitioned into three parts:

$$\begin{aligned} L(\hat{u}_i) &= ln\frac{P(u_i = 1|La(u_i))}{P(u_i = 0|La(u_i))} + ln\frac{P(x_i|u_i = 1)}{P(x_i|u_i = 0)} + \\ & ln\frac{P(u_i = 1|\mathbf{x}_1^{i-1}, \mathbf{x}_{i+1}^K, \mathbf{y}_1^K, La(\mathbf{u}_1^{i-1}), La(\mathbf{u}_{i+1}^K))}{P(u_i = 0|\mathbf{x}_1^{i-1}, \mathbf{x}_{i+1}^K, \mathbf{y}_1^K, La(\mathbf{u}_1^{i-1}), La(\mathbf{u}_{i+1}^K))}, \end{aligned} \tag{4.9}$$

where $\mathbf{x}_l^n$ means the sequence $(x_l, x_{l+1}, \cdots, x_n)$.

Thus the a posteriori LLR $L(\hat{u}_i)$ is composed of three LLRs. The first LLR contains the contribution from the input a priori information $La(u_i)$. The second LLR denoted $Li(x_i)$ and called the *intrinsic information*, since it contains the contribution directly from systematic channel observation x_i. The third LLR is denoted $Le(\hat{u}_i)$ and called the *extrinsic information*, since its contribution to $L(\hat{u}_i)$ stems from the other sources than the a priori information $La(u_i)$ and the systematic channel observation x_i. Noting that $P(u_i = k|La(u_i)) = e^{kLa(u_i)/(1+e^{La(u_i)})}$, $k \in \{0, 1\}$, the output LLR $L(\hat{u}_i)$ can be expressed as

$$L(\hat{u}_i) = La(u_i) + Li(x_i) + Le(\hat{u}_i). \tag{4.10}$$

In figure 4.5, each decoder have three outputs corresponding to the three addends in (4.10).

In an iterative decoder, the a priori information originates from the other constituent decoder; thus, it should not be included in the information that is passed to that decoder in the next decoding step. Likewise, the intrinsic information $Li(x_i)$ is directly available to the next decoder through the channel observation x_i. Consequently, the only portion of $L(\hat{u}_i)$ that is passed to the next decoder is the extrinsic information $Le(\hat{u}_i)$. Thus, it is the extrinsic information derived in one half-iteration that becomes the input a priori information in the next. By indicating the number of performed half-iteration with a superscript (l), we have

$$L(\hat{u}_i) = Le(\hat{u}_{i'})^{(l-1)} + Li(x_i) + Le(\hat{u}_i)^{(l)}, \tag{4.11}$$

where the subscript i' indicates the re-ordering performed by the interleaver and deinterleaver; output i' from half iteration $l-1$ is interleaved (if l even), or deinterleaved (if l odd) to position i before being passed to the lth half-iteration. In the following section, algorithms that can be used to derive the APPs are discussed.

4.3.1 APP Decoding

A device capable of calculating a posteriori probabilities of each information symbol based on the channel observation and a priori probabilities is called an *APP decoder*. An algorithm for estimating state and state transition probabilities of a Markov source was presented by Bahl *et al.* in 1974 [2]. The algorithm is suitable for decoding of convolutional codes, whose code words can be viewed as the output from a Markov source. The algorithm, often referred to as the BCJR-algorithm[2], is optimal in the sense of minimizing the symbol error rate. It was modified by Berrou *et al.* to account for a priori information [11]. In the Turbo coding literature, this algorithm is referred to as both the MAP- and the BCJR-algorithm.

The complexity of the BCJR-algorithm is higher than that of the Viterbi algorithm [29][63]. However the Viterbi algorithm does not produce a posteriori probabilities. Therefore, modifications to the Viterbi algorithm have been proposed. In [6], Battail presented a method to estimate

[2] After the initials of the authors: Bahl, Cocke, Jelinek and Raviv.

the reliability of symbols decoded with the Viterbi algorithm. A similar approach was taken by Hagenauer and Hoeher in [37], proposing the Soft Output Viterbi Algorithm (SOVA) which was later used for iterative decoding of Turbo codes in [38]. The Max-Log-MAP algorithm is a simplified version of the MAP- or BCJR-algorithm, with slightly inferior performance. However, with a low-complexity correction procedure, the performance of the Max-Log-MAP algorithm is very close to that of the original BCJR algorithm [56]. Additional low-complexity variants of the BCJR-algorithm, based on reducing the trellis search-space are reported in [33].

The BCJR Algorithm:

Here, we give a brief review of the BCJR-algorithm, a full derivation is given in Appendix A. The encoder input is, as above represented by $\mathbf{R} = (\mathbf{x}, \mathbf{y}, \mathbf{La})$. However, in the iterative decoding environment the true a priori information **La** is replaced by the extrinsic output from the previous decoding stage. Hence, the encoder input at time i is $R_i = (x_i, y_i, Le(\hat{u}_{i'})^{(l-1)})$, where $Le(\hat{u}_{i'})^{(l-1)}$ is the extrinsic output from the previous decoder that is the a priori input at bit position i in this decoding stage.

From the derivation in Appendix A, the decoder soft output decision variable after the lth half-iteration, i.e. the log-likelihood ratio $L^{(l)}(\hat{u}_i)$ can be expressed as

$$L(\hat{u}_i)^{(l)} = ln\frac{P(u_i = 1|\mathbf{R})}{P(u_i = 0|\mathbf{R})} = ln\frac{\sum_{s'}\sum_{s}\alpha_{i-1}(s^{'})\gamma_i^1(R_i, s^{'}, s)\beta_i(s)}{\sum_{s'}\sum_{s}\alpha_{i-1}(s^{'})\gamma_i^0(R_i, s^{'}, s)\beta_i(s)}, \tag{4.12}$$

where $s^{'}$ and s are the possible encoder states at time $i-1$ and i, respectively. The α,-, β-, and γ-quantities are related to state- and state-transition probability densities, defined in Appendix A. For BPSK modulation ($0 \rightarrow -1$ and $1 \rightarrow +1$ over an AWGN channel with noise variance σ^2, the trellis transition related probability density function $\gamma_i^k(R_i, s^{'}, s),\ k \in \{0, 1\}$, is (see Appendix A)

$$\gamma_i^k(R_i, s^{'}, s) = e^{(x_i(2k-1)+y_i(2c_i-1))/\sigma^2 + kLe(\hat{u}_{i'})^{(l-1)}} P(S_i = s|u_i = k, S_{i-1} = s^{'}), \tag{4.13}$$

where c_i is the parity bit associated with a transition between the states $s^{'}$ and s, caused by the information symbol $u_i = k$. The probability $P(S_i = s|u_i = k, S_{i-1} = s^{'})$ is either 0 or 1, depending on the existence of a trellis transition between the states $s^{'}$ and s caused by an information symbol $u_i = k$. The α- and β-values are calculated recursively, starting from the beginning and the end of the received block, respectively:

$$\alpha_i(s) = \sum_{s'=0}^{2^m-1}\sum_{k=0}^{1}\alpha_{i-1}(s^{'})\gamma_i^k(R_i, s^{'}, s) \tag{4.14}$$

$$\beta_i(s^{'}) = \sum_{s'}\sum_{k=0}^{1}\beta_{i+1}(s)\gamma_{i+1}^k(R_{i+1}, s^{'}, s), \tag{4.15}$$

where m is the number of memory elements in the encoder. With encoders initialized and terminated in their zero states, the recursions are initialized with

$$\alpha_0(s) = \begin{cases} 1 & s = 0 \\ 0 & otherwise \end{cases} \tag{4.16}$$

and

$$\beta_K(s) = \begin{cases} 1 & s = 0 \\ 0 & otherwise. \end{cases} \tag{4.17}$$

For encoders not terminated in the zero state at the end of each block, the initialization of the backward recursion is instead

$$\beta_K(s) = \frac{1}{2^m}, \quad s = 0, \cdots, 2^m - 1. \tag{4.18}$$

Finally, after calculating the decoder soft-output $L(\hat{u}_i)^{(l)}$, the extrinsic information is found as

$$Le(\hat{u}_i)^{(l)} = L(\hat{u}_i)^{(l)} - Le(\hat{u}_{i'})^{(l-1)} - Li(x_i), \tag{4.19}$$

where $Le(\hat{u}_{i'})^{(l-1)}$ is the a priori input and $Li(x_i)$ is calculated as

$$Li(x_i) = \frac{2}{\sigma^2} x_i. \tag{4.20}$$

4.4 Conclusion

The fundamental principle behind Turbo coding has been introduced, including the encoder structure and the principle of iterative decoding. The central components of a Turbo code encoder are the terminated *recursive systematic convolutional* (RSC) encoders and the interleaver that link them in parallel by re-ordering the bits in the information sequence before they enter the second constituent encoder.

A method to evaluate the bit error probability of a parallel concatenated coding scheme independently of the interleaver was discussed. Crucial was the introduction of a probabilistic interleaver called uniform interleaver which permits an easy derivation of the weight enumerating function of the parallel concatenated code starting from the weight enumerating function of the constituent codes.

The concept of iterative decoding relies on the use of soft-input/soft-output decoders, which calculates *a posteriori probabilities* (APPs) based on the received channel sequence and *a priori* information. An optimal algorithm for computing APPs is the BCJR-algorithm, also called the MAP-algorithm.

Chapter 5

Convolutional Coupled Codes

In the previous chapter the parallel concatenation of two convolutional encoder was presented. The error performance of such construction is very promising. Inspired from this construction which is near the Shannon limit error correction performance, we will consider a further development of concatenated convolutional codes. Instead of letting two constituent encoders, we put a set of parallel encoders in the place of the constituent encoders. In the construction systematic convolutional codes and systematic block codes are linked together such that only the systematic part of the convolutional codes is encoded with the block encoders. The bits of each information vector of the convolutional codes are scrambled by a given interleaver before entering to the block encoders. We call this construction *convolutional coupled codes* and *code coupling* is the new scheme of parallel code concatenation. Then, differently from the Turbo codes, in which information symbols and the redundancy from the constituent codes are transmitted [11], we transmit only the redundancy from the convolutional and block codes. By Turbo Codes, it was shown, that the observed flattening of the error correcting performance is a consequence of the relatively low free distance. The performance at this region may be improved by using elaborate interleavers. Nevertheless, one couldn't master the appearance of this error floor analytically. With convolutional coupled codes, we achieve a constructive controlled error floor and hence, the distance properties of the code is analytically estimated.

5.1 Encoder Structure

The class of the so called coupled codes was introduced in [15]. The coupled code is formed by ν identical systematic binary codes of rate k/n and minimum distance d_o and k identical systematic block codes with parameters $(2\nu, \nu, d_i)$. In order to distinguish between the constituent codes we use the terms *outer* and *inner* codes, thereby the k block codes will be named as inner codes and the first mentioned codes as outer codes. The indexes o and i refer to the outer and inner codes, respectively. Figure 5.1 shows the encoding scheme of a coupled code. According to it we will explain how the encoder works. Each coupled code word stems from $K = k \cdot \nu$ information bits.

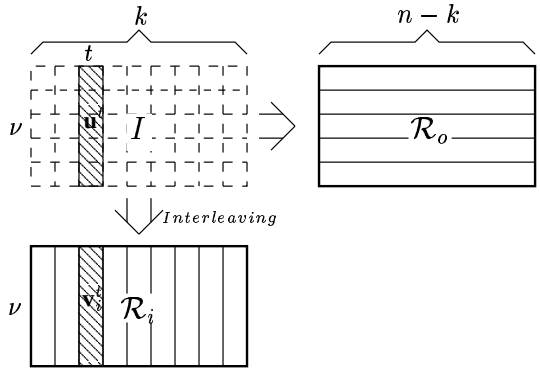

Figure 5.1: Encoding scheme of a rate k/n coupled code.

The ensemble of information bits are written in a $(\nu \times k)$ rectangular matrix I. Each row of the matrix represents an info word for a rate k/n outer encoder. The k information bits in the i-th row in the matrix I are fed into the corresponding outer encoder and the resulting $n-k$ parity bits are written into the i-th row of the matrix $\mathcal{R}_o$, $i = 1, 2, \cdots, \nu$. After encoding with the ν outer codes intra-row permutations are applied to the bit in matrix I according to the interleaving scheme described in section 5.4.

The number of inner block codes is given by k. Let $\mathbf{u}^t = (u_1^t, u_2^t, \cdots, u_\nu^t)$ denote the t-th column of I, $t = 1, 2, \cdots, k$. The word $\mathbf{u}^t$ is fed into the inner block encoder. The parity part of the resulting code word, $\mathbf{v}_i^t = (v_{i.1}^t, v_{i,2}^t, \cdots, v_{i,\nu}^t)$, is written into the t-th column of $\mathcal{R}_i$, $t = 1, 2, \cdots, k$.

Since only the two parts of redundancy $\mathcal{R}_o$ and $\mathcal{R}_i$ (see figure 5.1) produced by the outer codes and the inner block codes, respectively, are transmitted, the coupled code is *non-systematic*. The over-all code rate of the resulting code is equal to the rate of the outer codes and remains k/n.

Now, the binary convolutional coupled code (CCC) is a coupled code where the systematic outer codes are rate b/c recursive systematic convolutional codes. In this work we consider only terminated outer code sequences, where we start encoding in the all-zero encoder state and ensure that, after the encoding process, all memory elements contain zero again. Due to the termination bits of each outer RSC encoder, the over-all rate of the code is no longer b/c but is given as

$$R_c = \frac{b}{c}\frac{L}{L+m}, \tag{5.1}$$

where L and m are the length of the information sequence and the number of memory elements of an outer RSC encoder, respectively. The number of the inner encoders k is then given by $L + m$.

Let G be the $(2\nu \times \nu)$ encoding matrix of the systematic inner block codes. Let I_ν denote the $(\nu \times \nu)$ identity matrix. Then

$$G = [I_\nu P], \tag{5.2}$$

where P is a $(\nu \times \nu)$ matrix and is called *coupling matrix.*

The number of ones in each row of the coupling matrix P has to be constant and is called *coupling factor* ($\mathcal{K}$).

In the following sections we will show that the performance in terms of error rate for a wide range of SNR and the distance properties of the coupled codes depend on the coupling factor. Thus, for a given length ν the coupling factor has to be as large as possible and has to be odd, so that the weight of the inner parity sequence of the over-all code word can not be zero. A $(\nu \times \nu)$ *circulant matrix* with $\mathcal{K}$ ones in each row can be used as coupling matrix. In this case the inner block code is a *quasi cyclic code* of minimum distance $d_i \geq 4$ if $\nu \geq 4$.

As an example we give the following generator matrix

$$G = [I_\nu P] = \begin{bmatrix} \mathbf{1} & 0 & \cdots & 0 & \mathbf{0} & 1 & \cdots & 1 \\ 0 & \mathbf{1} & \cdots & 0 & 1 & \mathbf{0} & \cdots & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 \\ \vdots & & & & & & & \\ 0 & 0 & \cdots & \mathbf{1} & 1 & 1 & \cdots & \mathbf{0} \end{bmatrix}. \tag{5.3}$$

The entries of the coupling matrix are 0 in the positions (i, j), where $i = j$ and 1 otherwise.

5.1.1 Basic Idea

The basic idea behind this code construction is based on the cognizance, that coding not only redundancy, but also diversity (smearing of the information over many digits). Thereby, it is generally known that the information digits contributes relatively little to the minimum distance of a code. The inner block encoding is in fact simply a unique reversible mapping of the information word with the objective, that in the case where the parity weight is relatively little, the weight of the over-all code word will be larger. The part of the (new) information word on the minimum distance is described by the so called *coupling factor*. According to this, the codes are called *coupled codes*.

In section 5.1.3 we will show that the coupled codes are linear, that the encoding can be described by multiplying the input vector with a generator matrix, consequently the minimum distance can be attributed to the minimum weight code word. By means of some basic encoding scenarios, we will explain the basic idea behind this construction.

Let $\mathbf{c}_j$ be a code word of an outer component code C_j. The vectors $\mathbf{u}_j$ and $\mathbf{v}_j$ correspond to the information sequence and the parity-check sequence of the systematic encoder, respectively. Assume first that all other $\nu - 1$ systematic codes are zero. The weight of the resulting code word $\mathbf{c}_j$, after encoding the information sequences with the inner encoders, can hence be calculated as the sum of the parity weight $w(\mathbf{v}_j)$ plus the product term of the input weight $w(\mathbf{u}_j)$ and the *coupling factor* $\mathcal{K}$, i.e.,

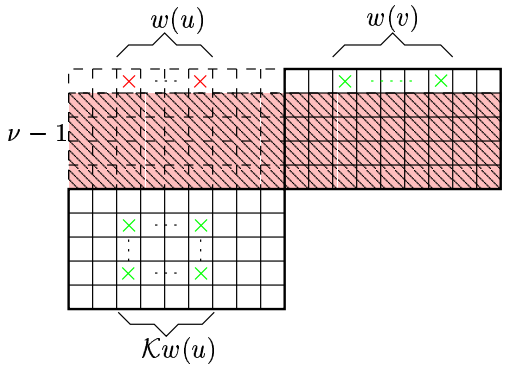

Figure 5.2: Encoding scheme in case of only one outer constituent code is different from zero.

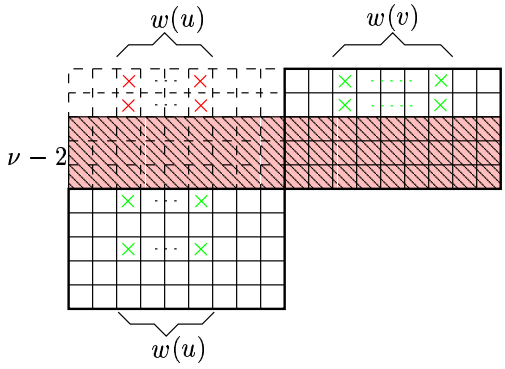

Figure 5.3: The worst case of two outer constituent codes different from zero.

$$w(\mathbf{c}_j) = \mathcal{K}w(\mathbf{u}_j) + w(\mathbf{v}_j). \tag{5.4}$$

Figure 5.2 shows the encoding scheme of the coupled code in case that only one outer constituent code is different from zero. Thus, the minimum weight code word in this case is given by the minimization of (5.4).

Now, assume that two outer codes are non-zero. Then, the worst case occurs if both outer code words are the same code word of minimum weight (see figure 5.3). I.e. the weight of each non-zero column in the information matrix I is two and hence produces an inner redundancy weight of at least 2. That means, that the expected increasing of the information weight through the inner encoding does not occur in this case. This has to be avoided. By an appropriate interleaving, the case of two low weight outer information vectors that leads to mostly weight-two inner information vectors may be avoided. Consequently, the number of low weight code words which dominates the performance expressed in terms of bit error rate, may be reduced.

If there are at least three non-zero outer code words, then we will obtain a large parity weight from the outer codes so that the inner parity weight is no longer important for the minimum weight over-all code word.

In other words, with this code construction one can achieve, that if the information weight is small, then we obtain a large inner parity weight. In contrast, if the information weight is large

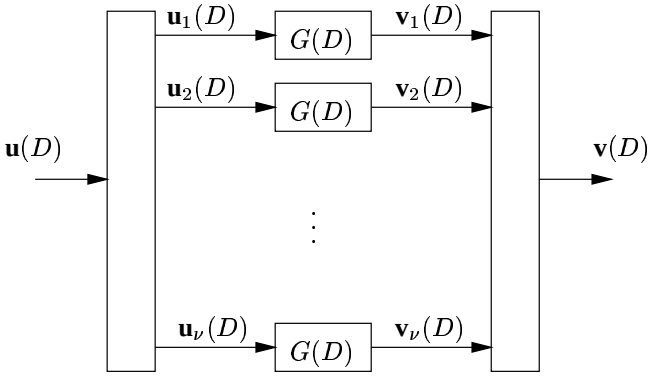

Figure 5.4: A parallel composition of ν identical encoders

enough, then is the outer parity weight large and thus we obtain a large over-all weight. Since the over-all code word contains no information sequence, which is simply mapped through the coupling matrix, we conclude, that an improvement on the distance properties of a code without additional redundancy can be achieved.

5.1.2 Generator Matrix of a Parallel Composition of Encoders

The parallel composition of codes can be considered as the main building block of a coupled code. Assume that all outer encoders are identical. In figure 5.4 we show a parallel composition with ν outer rate b/c encoders, each with generator matrix G.

The information sequence of the structure is divided into sub-blocks of $b\nu$ information symbols. Then we have

$$\mathbf{u}(D) = (\mathbf{u}^{(1)}(D) \cdots \mathbf{u}^{(b)}(D)), \tag{5.5}$$

where

$$\mathbf{u}^{(i)}(D) = (u_1^{(i)}(D), u_2^{(i)}(D), \cdots, u_\nu^{(i)}(D)). \tag{5.6}$$

The indeterminate D can be interpreted as a delay operator.

The information sequence of the l-th constituent outer encoder is

$$\mathbf{u}_l(D) = (u_l^{(1)}(D), u_l^{(2)}(D), \cdots, u_l^{(b)}(D)) \tag{5.7}$$

and the corresponding code sequence

$$\begin{aligned} \mathbf{v}_l(D) &= \mathbf{u}_l(D)G(D) \\ &= (\mathbf{u}_l(D)\mathbf{g}_1(D), \cdots, \mathbf{u}_l(D)\mathbf{g}_c(D)) \\ &= (v_l^{(1)}(D), \cdots, v_l^{(c)}(D)), \end{aligned} \tag{5.8}$$

where $\mathbf{g}_j(D)$ is the j-th column of the generator matrix $G(D)$, i.e.,

$$G(D) = \begin{pmatrix} g_{11}(D) & \cdots & g_{1c}(D) \\ \vdots & \vdots & \vdots \\ g_{b1}(D) & \cdots & g_{bc}(D) \end{pmatrix} = (\mathbf{g}_1(D)\mathbf{g}_2(D)\cdots\mathbf{g}_c(D)). \tag{5.9}$$

Each of these ν code sequences are serialized and written row-wise into a buffer. We obtain

$$\mathbf{v}(D) = (\mathbf{v}^{(1)}(D)\cdots\mathbf{v}^{(c)}(D)), \tag{5.10}$$

where

$$\begin{aligned} \mathbf{v}^{(i)}(D) &= (v_1^{(i)}(D), v_2^{(i)}(D), \cdots, v_\nu^{(i)}(D)) \\ &= (\mathbf{u}_1(D)\mathbf{g}_i(D)\cdots\mathbf{u}_\nu(D)\mathbf{g}_i(D)). \end{aligned} \tag{5.11}$$

To obtain a compact notation for the generator matrix of the parallel composition of the codes we use the matrix (Kronecker) tensor product. This is a matrix consisting of all possible products with one elements taken from each matrix.

Definition 5.1: *Let A and B be two matrices of size $m \times n$ and $p \times q$, respectively. The tensor product $A \otimes B$ is the $mp \times nq$ matrix*

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}. \tag{5.12}$$

If the matrix product are defined, then

$$(A \otimes B)(C \otimes D) = AC \otimes BD. \tag{5.13}$$

This immediately gives that the inverse of the tensor product is the tensor product of the inverse,

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \tag{5.14}$$

Equation (5.10) can now be rewritten as

$$\mathbf{v}(D) = (u_1^{(1)}(D)\cdots u_\nu^{(1)}(D) \ \cdots \ u_1^{(b)}(D)\cdots u_\nu^{(b)}(D))$$

$$\times \begin{pmatrix} g_{11}(D) & & & & g_{1c}(D) & & \\ & \ddots & & \cdots & & \ddots & \\ & & g_{11}(D) & & & & g_{1c}(D) \\ & \vdots & & & & \vdots & \\ g_{b1}(D) & & & & g_{bc}(D) & & \\ & \ddots & & \cdots & & \ddots & \\ & & g_{b1}(D) & & & & g_{bc}(D) \end{pmatrix}$$

$$= \mathbf{u}(D)(G(D) \otimes I_\nu), \tag{5.15}$$

where $\mathbf{u}(D)$ is given by (5.5) and I_ν is the $\nu \times \nu$ identity matrix. From (5.15) we conclude that the $b\nu \times c\nu$ generator matrix $G^p(D)$ for a parallel composition with ν identical constituent encoders with a rational generator $G(D)$ is

$$G^p(D) = G(D) \otimes I_\nu. \tag{5.16}$$

Clearly, we also have

$$G^p = G \otimes I_\nu \tag{5.17}$$

where the generator matrix G is

$$G = \begin{pmatrix} G_0 & G_1 & \cdots & G_m & & \\ & G_0 & G_1 & \cdots & G_m & \\ & & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{5.18}$$

where m is the memory length of a constituent encoder.

The right inverse of $G^p(D)$ is

$$G^p(D)^{-1} = (G(D) \otimes I_\nu)^{-1} = G^{-1}(D) \otimes I_\nu \tag{5.19}$$

5.1.3 Generator Matrix of the Convolutional Coupled Code

From the generator matrix of the parallel composition of the outer codes it is easy to obtain the generator matrices for the convolutional coupled code. It consists of a cascade of ν outer identical RSC codes of rate b/c and a number k of inner block code with generator matrix as given in (5.2). Assume that the rate of the RSC codes is $1/2$ and has the generator matrix $G_o = (1, \frac{n(D)}{d(D)})$. The generator matrix of the parallel composition of the ν outer codes is given by

$$\begin{aligned} G_o^p(D) &= G_o(D) \otimes I_\nu = \begin{pmatrix} 1 & & & \frac{n(D)}{d(D)} & & \\ & 1 & & & \frac{n(D)}{d(D)} & \\ & & \ddots & & & \ddots \\ & & & 1 & & \frac{n(D)}{d(D)} \end{pmatrix} \\ &= [I_\nu, \frac{n(D)}{d(D)} I_\nu]. \end{aligned} \tag{5.20}$$

After encoding the information sequence

$$\mathbf{u}(D) = (u_1(D), u_2(D), \cdots, u_\nu(D)) \tag{5.21}$$

with (5.20), where $u_t(D)$ are polynomials of maximum degree $k-1$, we obtain an output sequence whose parity part is given by

$$\mathbf{v}_o(D) = (v_{o,1}(D), v_{o,2}(D), \cdots, v_{o,\nu}(D)). \tag{5.22}$$

The generator matrix of the parallel composition of the k inner block codes can be given by

$$\begin{aligned} G_i^p(D) &= G_i(D) \otimes I_k = \begin{pmatrix} 1 & & & P(D) & & \\ & 1 & & & P(D) & \\ & & \ddots & & & \ddots \\ & & & 1 & & P(D) \end{pmatrix} \\ &= [I_k, P(D) I_k], \end{aligned} \tag{5.23}$$

where $P(D)$ is the generator polynomial of the circulant coupling matrix as given in the example of (5.3). The information sequence is now defined as

$$\mathbf{u}(D) = (u_1(D), u_2(D), \cdots, u_k(D)), \tag{5.24}$$

where $u_t(D)$ are polynomials of maximum degree $\nu - 1$. The parity part of the obtained sequence is

$$\mathbf{v}_i(D) = (v_{i,1}(D), v_{i,2}(D), \cdots, v_{i,k}(D)). \tag{5.25}$$

Since the information sequences in (5.21) and (5.24) are equivalent and only the parity parts (5.22) and (5.25) are transmitted, we conclude that the generator matrix of the coupled code consists of the right part of the matrix in (5.20) and the right part of the matrix in (5.23), i.e.,

$$G_c(D) = [P(D) I_k, \frac{n(D)}{d(D)} I_\nu] \tag{5.26}$$

From this it follows that it is always possible to find a parity-check matrix H_c of the form

$$H_c(D) = \begin{pmatrix} P(D)^{-1}I_k & \mathbf{0}_\nu \\ \mathbf{0}_k & I_\nu \end{pmatrix} \cdot H_o(D)I_\nu \tag{5.27}$$

where $H_o(D)$ is the parity-check matrix of an outer RSC code, and $\mathbf{0}_l$ is the $l \times l$ zero matrix. This can be explained as follow: The received code word consists of the inner parity sequence $\mathbf{v}_i(D)$ and the outer parity sequence $\mathbf{v}_o(D)$ as given in (5.25) and (5.22), respectively. Multiplying the over-all code word with $[P(D)^{-1}I_k, \mathbf{0}_\nu]^T$ results the information word $\mathbf{u}(D)$ and with $[\mathbf{0}, I_\nu]^T$ results the outer parity sequence $\mathbf{v}_o(D)$. This two sequences together present a code word for the ν outer codes. Thus, multiplying this with the parity check matrix of the parallel composition of the ν outer codes, must be zero.

5.1.4 Distance Properties of the Coupled Codes

In this section we will estimate the minimum distance of the coupled code.

The over-all minimum distance of the coupled code can be lower bounded by

$$d_{min} \geq \min_{\substack{x \in 1, \cdots, \nu \\ y \in 1, \cdots, k}} (y \max\{1, d_i - x\}) + x \max\{1, d_o - y\}, \tag{5.28}$$

where d_o and d_i are the minimum distances of the outer and inner codes, respectively.

Proof: Suppose only one outer code is non-zero, that is $x = 1$. Let y be the number of non-zero information bits. By the coupling matrix, the weight of each of this single information bits produces at least (the minimum inner weight is d_i) an inner parity weight of $d_i - 1$. As the outer code has the minimum weight d_o, the outer parity weight is at least $d_o - y$. Then suppose two outer codes are non-zero, etc. Since the inner parity weight and the outer parity weight, when increasing x and y, respectively, are never zero then they are lower bounded by 1. □

Example 5.1:

We will give a simple example for code coupling to demonstrate the unequation (5.28).

We consider 4 *binary extended Hamming codes* $(8, 4, 4)$ *as outer component codes. The same component code with parameters* $(8, 4, 4)$ *is used as inner codes. The encoding scheme is obtained as in figure 5.5. The code word is formed by the parity-check bits generated by the outer and the inner encoders. From (5.28), it is apparent, that the minimum distance of this coupled code is* 6*. The encircled elements in figure 5.5 indicate the binary digits equal to* 1 *for a possible minimum weight code word. The parameters of the resulting code are* $(32, 16, 6)$.

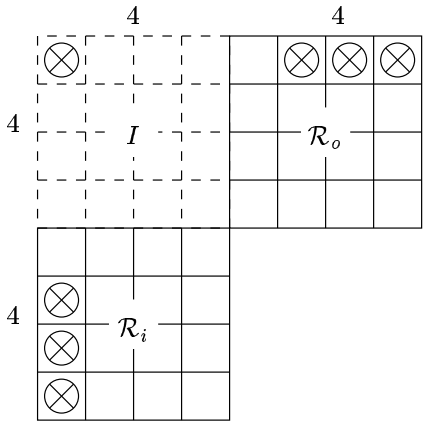

Figure 5.5: Encoding scheme of the $(32, 16, 6)$ coupled code.

On the other hand the minimum distance of the convolutional coupled code can be upper bounded by

$$d_{min} \leq 2\mathcal{K} + d_o - 2. \tag{5.29}$$

where d_o is the free distance of an outer RSC code and $\mathcal{K}$ is the coupling factor.

Proof: This is simply obtained if we assume that only one outer RSC code is different from the zero code word. The minimum input weight of the terminated RSC encoder is 2 [8]. Thus, each of this two non zero information bits will generate the weight $\mathcal{K}$ in the inner parity part of the over-all code word. Moreover, the weight of the outer parity part is at least $d_o - 2$. □

5.2 Decoding of Convolutional Coupled Codes

The encoder of a convolutional coupled code can be regarded as a two step encoder. It is therefore suitable to apply an iterative decoding scheme. Iterative decoding became popular when Turbo codes were introduced. It uses soft decision decoding and information from one constituent decoder is passed on to another. Thereby, the BCJR-algorithm is used for calculating the a posteriori probabilities. It was modified by Berrou *et al.* to account for a priori information. With this modification, a maximum a posteriori (MAP) decoding algorithm is obtained. The only requirement for the MAP-algorithm is a binary trellis. In [2] it was shown that MAP decoding can be applied to any code for which a trellis, especially a binary trellis, can be drawn. A convolutional coupled code consists of outer convolutional codes and inner block codes. By decoding of this codes the BCJR algorithm is considered not only for the convolutional decoder but also for the block decoder. It is well known that the code words of a linear binary (b, c) block code C can be presented as paths through a trellis of depth b with a most 2^{b-c} states [5][64][2]. For the construction of the trellis, the systematic matrix H of the code is used and this results in

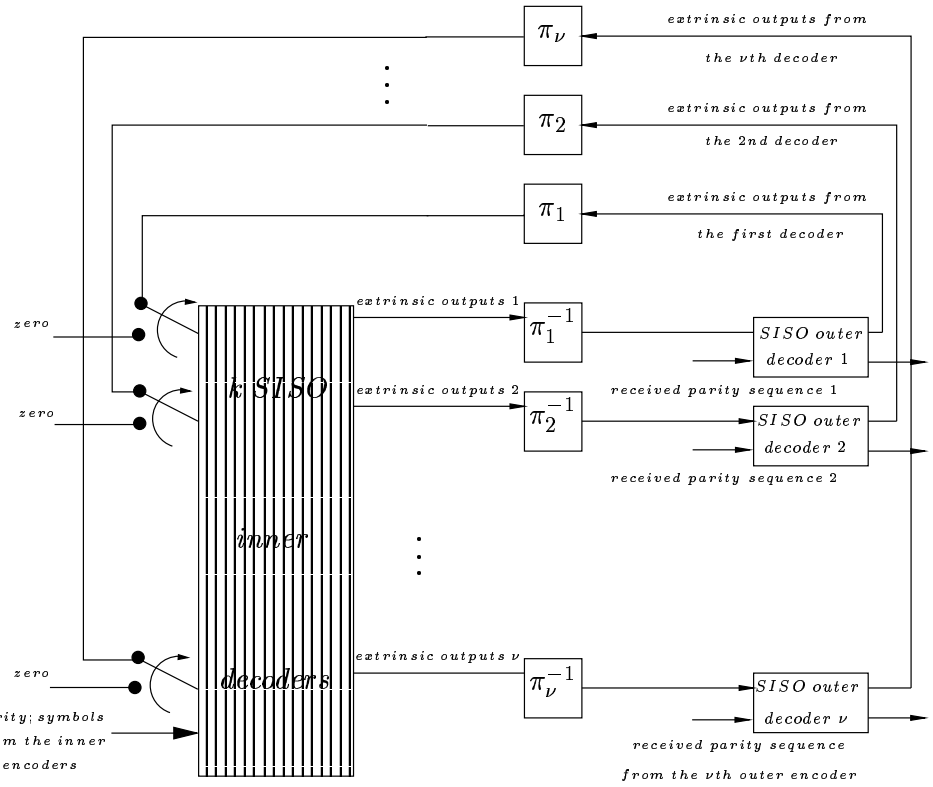

Figure 5.6: Block diagram of the iterative decoder for the convolutional coupled code.

a trellis with an irregular structure as opposed to the regular trellis of convolutional codes (see section 2.1.2).

In this section, we present the iterative algorithm for decoding convolutional coupled codes with outer rate $1/2$ convolutional codes. The iterative Turbo decoder structure is shown in figure 5.6. It consists of two stages: a block of k soft-input soft-output (SISO) inner decoders, followed by ν parallel SISO outer decoders. The two stages are separated by deinterleavers and interleavers. The concept of the soft-input soft-output component decoder shall be briefly illustrated in this section. At first we consider the inner SISO decoders. The parity sequence from the inner encoders are divided into sub-blocks of ν symbols, thereby at each time instant $t = 1, 2, \cdots, k$ there are two different types of soft input sequences, namely

- the redundant information block sequence

$$\mathbf{y}_i^t = (y_{i,1}^t, y_{i,2}^t, \cdots, y_{i,\nu}^t),$$

- the a-priori information block sequence

$$La_i(\mathbf{u}^t) = (La_i(u_1^t), La_i(u_2^t), \cdots, La_i(u_\nu^t)).$$

In contrast to the Turbo component decoder, there is no systematic input sequence. Thus, the input blocks of the t-th inner SISO decoder are in short referred to as $\mathbf{R}_i^t$, that is

$$\mathbf{R}_i^t = (\mathbf{y}_i^t, La_i(\mathbf{u}^t)). \tag{5.30}$$

It is now the task of the soft-input soft-output decoder to appropriately combine the two kinds of information to generate a soft output which can be beneficially processed by the other soft-input soft-output component decoder. The soft-output at the stage l contains

- a weighted version of the a-priori information sequence $La_i(u_l^t)$, and
- a newly generated extrinsic information sequence $Le_i(\hat{u}_l^t)$, which is a combination of the influences of all soft inputs except for the weighted version of $La_i(u_l^t)$.

The a posteriori LLR $L(\hat{u}_l^t)$ can be expressed as

$$\begin{aligned} L(\hat{u}_l^t) &= ln\frac{P(u_l^t = 1|\mathbf{R}_i^t)}{P(u_t^l = 0|\mathbf{R}_i^t)} \\ &= \frac{P(u_l^t = 1|La_i(u_l^t))}{P(u_l^t = 0|La_i(u_l^t))} + ln\frac{P(u_l^t = 1|\mathbf{y}_i^t, La_i(\mathbf{u}_{1,l-1}^t), La_i(\mathbf{u}_{l+1,\nu}^t))}{P(u_l^t = 0|\mathbf{y}_l^t, La_i(\mathbf{u}_{1,l-1}^t), La_i(\mathbf{u}_{l+1,\nu}^t))} \\ &= La_i(u_l^t) + Le_i(\hat{u}_l^t), \end{aligned} \tag{5.31}$$

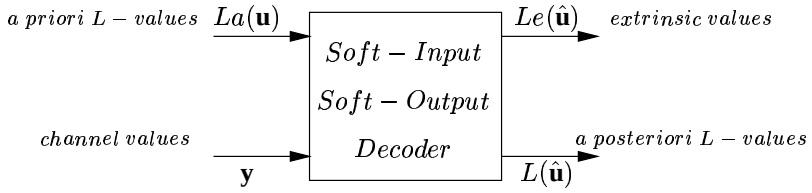

Figure 5.7: Soft-Input Soft-Output decoder.

where $\mathbf{u}_{p,q}^t$ is defined as $(u_p^t, u_{p+1}^t, \cdots, u_q^t)$.

Figure 5.7 shows a soft-input soft-output decoder, with input and output ports corresponding to the described iterative decoding situation.

During the first iteration, the a priori informations $La_i(u_l^t),\ t = 1, 2, \cdots, k$ and $l = 1, 2, \cdots, \nu$ are set to zero.

The extrinsic LLRs $Le_i(\hat{u}_l^t)\ t = 1, 2, \cdots, k$ are passed through the l-th inverse interleaver (a block labeled π_l^{-1}), whose outputs correspond to the a priori information LLRs of the l-th outer code. The LLR sequence

$$La_o(\mathbf{u}^l) = (La_o(u_1^l), La_o(u_2^l), \cdots, La_o(u_k^l))$$

is then sent to the l-th block "outer SISO". Together with the received parity sequence

$$\mathbf{y}_o^l = (y_{o,1}^l, y_{o,2}^l, \cdots, y_{o,k}^l)$$

from the l-th outer encoder, they correspond to the L-values of the code symbols of the l-th outer code.

The input sequence of the l-th soft-input soft-output outer decoder is given by

$$\mathbf{R}_o^l = (La_o(\mathbf{u}^l), \mathbf{y}_o^l).$$

Similar to the inner decoders, the soft-output of the l-th decoder at time instant t contains two parts, (1) a weighted version of the a-priori information sequence $La_o(u_t^l)$, and (2) a generated extrinsic information sequence $Le_o(\hat{u}_t^l)$ which after interleaving is used as a priori information of the inner decoder, i.e.,

$$L(\hat{u}_t^l) = La_o(u_t^l) + Le_o(\hat{u}_t^l) \tag{5.32}$$

The discussed decoding procedure is repeated several times before the final decision is taken. The E_b/N_0 which is required to obtain a certain error performance decreases with each iteration step of the iterative decoder.

5.2.1 Decoding Complexity

Besides the error correcting performance, the decoding complexity is important for every channel coding scheme. Like Turbo codes, the complexity of the convolutional coupled code is dominated by the complexity of the component decoders and the complexity of the data exchange between the outer and inner decoders. Differently from the Turbo codes, convolutional coupled codes use RSC and block codes. Consequently, the difference in the complexity to Turbo codes can only emerge from the inner SISO block decoders. Since the inner block decoder performs the MAP algorithm, it makes sense to use the minimal trellis of the block code to minimize the complexity. The notion of minimal trellis was introduced by Forney [30] and is defined as follows [51]:

Definition 5.2: *Let T and $\bar{T}$ be two trellises of a code C. $|V_j|$ indicates the number of nodes of the j-th level in T and $|\bar{V}_j|$ the number of nodes of the j-th level in $\bar{T}$. A trellis T is minimal, if and only if for any trellis $\bar{T}$*

$$|V_j| \leq |\bar{V}_j| \quad for\ all\ j.$$

Hence, a minimal trellis is then a trellis, that holds for every time instant a minimum number of nodes.

We will now show that the trellis complexity of the inner codes by using a certain coupling matrix can be significantly reduced. Here we will consider a (4×4) coupling matrix with only one zero in each row. Now, the t-th SISO inner decoder, $1 \leq t \leq k$, uses the a priori values $(La_i(u_1^t), \cdots, La_i(u_4^t))$ and the channel values $(y_{i,1}^t, \cdots, y_{i,4}^t)$ as input sequences. It is useful for the evaluation of a trellis with low state-complexity to combine the a priori values and the channel values in such a form that we obtain a sequence

$$(La_i(u_1^t), y_{i,1}^t, La_i(u_2^t), y_{i,2}^t La_i(u_3^t), y_{i,3}^t, La_i(u_4^t), y_{i,4}^t)$$

as input of the t-th SISO inner decoder. This input sequence can be assigned to a code word $\mathbf{c}$ of a code $C^{'}$ whose encoding matrix is

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

The columns in bold face correspond to the columns of the identity matrix and the columns between correspond to the coupling matrix. This code can be represented by the trellis T given in figure 5.8 (for convenience, in this figure, a thick edge is labeled by 1, and a thin edge, labeled by 0). Thereby the encoding matrix $G^{'}$ was transformed in the following matrix $G_m^{'}$ equivalent to $G^{'}$:

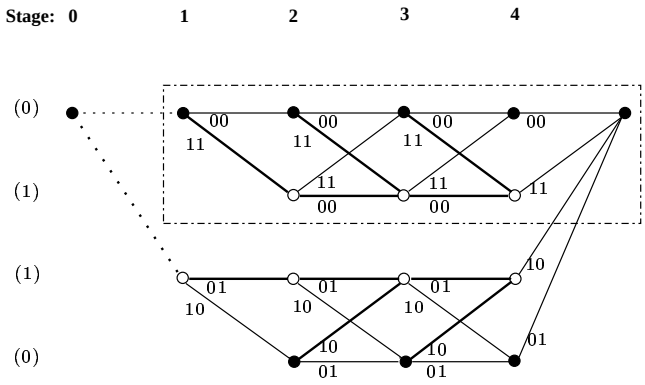

Figure 5.8: Trellis representing the code C'.

$$G'_m = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The trellis is built up by considering the three last rows of the matrix G'_m as generator matrix of a terminated convolutional code C_c of memory 1 and transfer function matrix

$$G_c(D) = [1 + D, 1 + D].$$

The trellis diagram representing C_c is shown in figure 5.8 (see the marked frame in the upper part of the figure). Note that the trellis state-complexity of the code C_c is 2. The upper part of the trellis T represents all the code words in C' if the first info bit is zero. The lower part of the trellis gives all the code words in C' if the first info bit is set to one. Thus the trellis T represents C'. Moreover we point out that the trellis state-complexity is independent on the length ν of the coupling matrix, i.e., the number of outer codes, and is alway equal to 4.

Generally, we can not say, whether the convolutional code in Turbo codes or the block code in coupled codes features a higher complexity. This depends always on the used coupling matrix P.

5.3 Effective Free Distance

Due to the interleavers between the inner and outer codes, it is not possible to evaluate the free distance of the convolutional coupled codes analytically. “Effective” signifies in this case that this is only an estimation on the free distance of the code. The effective free distance of convolutional coupled codes is defined in [16]. Thereby, we assume that only one of the ν outer convolutional codes is different from zero. Let $\mathbf{c}_j$ be a code word of this non zero outer component encoder. $\mathbf{u}_j$ and $\mathbf{v}_j$ correspond to the information sequence and the parity-check sequence

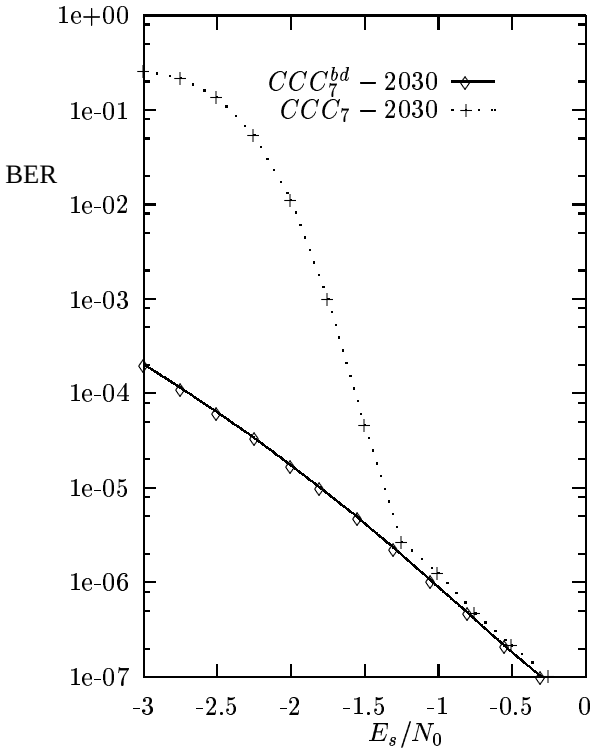

Figure 5.9: Performance of rate $\simeq 1/2$ convolutional code CCC_7 of code length 2030 symbols and comparison with the performance of the sub-code CCC_7^{bd}

of this RSC encoder. The weight of the resulting over-all code word $\mathbf{c}_j$, after encoding the information sequence with the inner encoders, can hence be given by

$$w(\mathbf{c}_j) = \mathcal{K}w(\mathbf{u}_j) + w(\mathbf{v}_j).$$

In this context we propose the following definition

Definition 5.3: *(Modified active row distance)*

*Consider a rate $1/2$ recursive systematic convolutional code C. Let **0** be the zero state and σ_t be the state at time instance t of the path which belongs to the code word $\boldsymbol{c} = (c_1, c_2, \cdots)$. $c_t = (u_t, v_t)$ is a code block of length 2. u_t is the input bit and v_t is the parity bit. We assume that the parity bits are weighted with one and the input bits are weighted with the coupling factor $\mathcal{K}$. The modified active row distance at a position j is defined as*

$$d_j^{r,m} = \min_{\boldsymbol{C} \in \gamma_j} \sum_{t=0}^{j-1} (\mathcal{K}w(u_t) + w(v_t)) \tag{5.33}$$

with

$$\gamma_j = \{\boldsymbol{v} \in C | \sigma_0 = \sigma_j = \boldsymbol{0}, \sigma_t \neq \boldsymbol{0} \; if \; \sigma_{t-1} = \boldsymbol{0} \; \forall \; t < j\} \, . \tag{5.34}$$

By means of the following example, we will discuss the term of effective free distance.

The simulated bit error rate performance of a rate $1/2$ convolutional coupled code with $\nu = 7$ outer RSC codes of memory 3 (CCC_7) is shown in figure 5.9. The coupling factor here takes the value 5, i.e., the coupling matrix contains two zeros in each row. The generator polynomial of the outer RSC codes is $g(D) = (1 + D + D^3 / 1 + D + D^2 + D^3)$. The simulation was done for the binary-input additive white Gaussian noise channel. The bit error rate curve of the convolutional coupled code is similar to that of the Turbo codes and can be divided into three regions. For the first region at low SNR, the bit error rate is very poor. This is due to the fact that the iterative decoder, even after a great number of iterations, does not converge. Increasing the SNR, there is a region, within which the BER drops rapidly several decades. This region is called "waterfall region". The reason for this behavior is that the iterative decoder often converges and thus a better error correction is achieved. Finally, in the "error floor region" which appears at high SNR the BER is dominated by code words of small weight. In this region it is assured that the decoder always almost converges.

As reference, we investigate the performance of the sub-code CCC_7^{bd} which is defined as a convolutional coupled code with only one of the 7 outer codes being different from zero. Consequently the minimum distance of the sub-code CCC_7^{bd} can be given by the minimum of the modified active row distances according to (5.33), i.e.,

$$d_{min}^{bd} = \min_{j=1,2,\cdots} \left\{ d_j^{r,m} \right\} . \tag{5.35}$$

The minimum distance of CCC_7^{bd} was found by computer search and is equal to 16. Decoding the sub-code CCC_7^{bd} can be interpreted as decoding the convolutional coupled code CCC_7 with the additional a priori knowledge that $\nu - 1$ outer codes are zero. Therefore we can consider the performance of CCC_7^{bd} as a lower bound for performance of CCC_7.

Figure 5.9 shows that for higher SNR the performance curve of CCC_7 gets very close to that obtained for CCC_7^{bd}. This convergence gives a heuristic evidence for the fact that the minimum distance properties of CCC_7 and CCC_7^{bd} are similar. Thus the minimum distance of the convolutional coupled code can be estimated by equation (5.35). We call d_{min}^{bd} the *effective free distance* and CCC_ν^{bd} the *effective boundary coupled code*. Hence, the effective free distance will be denoted by $d_{free,eff}$.

5.4 Interleaving

Interleavers have been used in communication systems for a long time; the classical use is to randomize the location of errors, enabling the use of random-error-correcting codes on channels with bursts error patterns. Typically, bursty channels including fading channels, often found

in wireless transmission. Another use of interleaving is in concatenated coding scheme, where the output of the inner-stage decoder exhibits burst error patterns (as happens with a viterbi decoder). Hence, the most important parameter of the interleaver in this case is its ability to spread error bursts such that they appear as isolated errors to the outer-stage decoder. Naturally, the optimum interleaver would achieve this with the minimum memory [54]. It should be clear that the required parameters of the interleaver in this case depend uniquely on the inner and outer codes and decoders used.

Turbo coding, however, also introduced a further dimension to what is required from the interleaver - this involves the effects of the iterative algorithm and the passing of intrinsic information between successive decoder stages. In this context, the interleaver has often been explained as reducing the correlation between the parity bits corresponding to the original and interleaved data frames. Already in the original paper introducing Turbo codes, Berrou *et al*. showed an understanding of some of the most important parameters making a good interleaver. In particular, increasing the block size results in improved performance, and the interleaver should randomize the input sequence in order to avoid particular low-weight patterns mapping onto themselves, reducing the effective free distance of the Turbo codes.

Like Turbo codes, interleavers should obviate the becoming of low-weight code words. In section 5.1.1 we have shown that by an appropriate interleaving, the case of two low weight outer information vectors that lead to mostly weight-two inner information vectors may be avoided. Consequently, the number of low-weight code words which dominates the performance expressed in terms of bit error rate, can be reduced.

An interleaver π is a permutation $i \rightarrow \pi(i)$ that maps a data sequence of k input symbols $(u_1, u_2, \cdots, u_k)$ onto the same sequence in a new order. If the input sequence is $\mathbf{u} = (u_1, u_2, \cdots, u_k)$, then the permuted data sequence is $\mathbf{u}M_p$, where M_p is an interleaving matrix with a single 1 in each row and column, all other entries being 0. Every interleaver has a corresponding deinterleaver π^{-1} that acts on the interleaved data sequence and restores it to the original. The deinterleaving matrix is simply the transpose of the interleaving matrix M_p^T. In the following we present 3 interleaver principles, which are used in this work.

5.4.1 The Modulo Interleaver

Let $\mathbf{u} = (u_0, u_1, \cdots, u_{k-1})$ be a vector of length k and let I, $0 < I < k$, be a number which is relatively prime to k. The the interleaved sequence $\mathbf{u}_g$ is defined by

$$\mathbf{u}_g = (u_{g,0}, u_{g,1}, \cdots, u_{g,k-1}), \quad u_{g,l} = u_{i \cdot I \ mod \ k}.$$

The deinterleaving is given by I^{-1} with $I \cdot I^{-1} = 1 \ mod \ k$.

The interleaving has to be selected such that the permuted information sequences of several low weight outer code words have ones in as different as possible places. Here, we have considered

modulo interleavers with different values $I_1, I_2, \cdots, I_\nu$ such that each outer code with own interleaver. They fulfill the following condition

$$I_j \cdot I_i^{-1} \neq 1 \; mod \; k, \quad 2 \leq i \leq \nu, \; 1 \leq j \leq \nu.$$

In the remainder of this work, we assume that the information vector of the first outer code is not interleaved, i.e. $I_1 = I_1^{-1} = 1$.

5.4.2 The Random Interleaver

A random interleaver is simply a random permutation π. In order to be able to deinterleave the sequence, the random must be deterministic, such that the random table can be generated at the interleaver and the deinterleaver.

5.4.3 The s-Random Interleaver

For the large block size interleavers, most random interleavers perform well. However, for short interleavers, the performance of the code with a random interleaver degrades substantially. For short block length interleavers, selection of the interleaver can have a significant effect on the performance of the coupled codes.

The semi-random interleaver is based on the random selection without replacement of k integers from 1 to k, where k is the interleaver length under constraints. This kind of interleavers was discussed in [32].

Constraint 1: The i-th randomly selected integer $\pi(i)$ must be rejected if there exists j such that:

$$0 < i - j \leq S_1, \; and \; |\pi(i) - \pi(j)| \leq S_2.$$

This constraint guarantees that if two symbols i, j, are within distance S_1 in the sequence without interleaving, they can not be mapped to distance less than S_2 in the interleaved sequence.

We can also add other constraints to improve again the performance of the interleaver. An extension of this procedure is to consider multiple error events, in the sequence without interleaving. As an example, the figure 5.10 depicts two error events that interchange their component symbols. To avoid this situation we define two more parameters T_1 and T_2 and impose on the construction of the s-interleaver an additional constraint. Again randomly select without replacement integer from 1 to k, and if the i-th selection $\pi(i)$ satisfies the constraint 1 described previously, check if the following condition are also satisfied.

Constarint 2: The i-th randomly selected integer $\pi(i)$ must be rejected if there exist any triple j, t, l, $j, t, l < i$ for which the conditions:

$$0 < i - j \leq T_1, \quad |\pi(i) - \pi(t)| \leq T_2$$

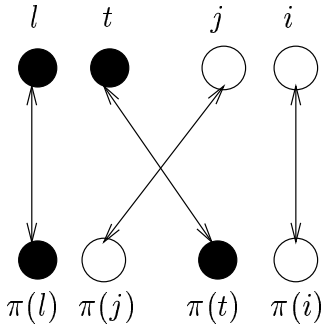

Figure 5.10: Error event.

$$0 < |t - l| \leq T_1, \quad |\pi(j) - \pi(l)| \leq T_2$$

are fulfilled.

This constraint guarantees that the two relatively close component symbols i and j, do not have $\pi(t)$ near $\pi(i)$ and $\pi(l)$ near $\pi(j)$ in the interleaved sequence, with t and l near each other.

An interleaver according to constraint 1, converges in a reasonable time if $S_1 S_2 \leq k/2$.

5.4.4 Simulation Results

The simulated performances of a rate $1/2$ convolutional coupled code with $\nu = 7$ outer RSC codes of memory 3 for different interleaving schemes are shown in figure 5.11. The over-all code length is 2016 code bits. This corresponds to an interleaver length of $k = 144$. The simulated interleavers are modulo, random, and s-random. For the modulo interleaving scheme we use different values which fulfill the condition defined above. For the s-random interleaving scheme we use different $S = S_1 = S_2$ for each outer information sequence and we consider just the constraint 1, previously defined. Figure 5.11 shows that in the "waterfall region", all the interleavers have the same comportment. In the "error floor region" the random interleaver gives almost the same performance as the s-random interleaver; it is just a little less good. This very small gain of performance is brought by the constraint of the s-random interleaver.

However, it is allowed to think that the performance of the modulo interleaver would be worse than those of the s-random interleaver. Figure 5.11 shows that the performance of the two interleaving schemes are very close to each other. Moreover, the figure shows, that no curve falls below the performance of the effective boundary coupled code (CCC_7^{bd}). This confirms our expectation and shows that in contrast to Turbo codes, the structure of the convolutional coupled codes allows to give a lower bound on the performance of the error rates even for medium code lengths. Only the BER of long Turbo codes employing uniform interleaving can be approximated for high SNR. In addition, we note that by means of simple interleaving scheme such as modulo- or random interleavers, we achieve good performance and have potential of being alternative to other designed interleavers. We conclude that for convolutional coupled codes, interleaver-design is not of same importance as for Turbo codes.

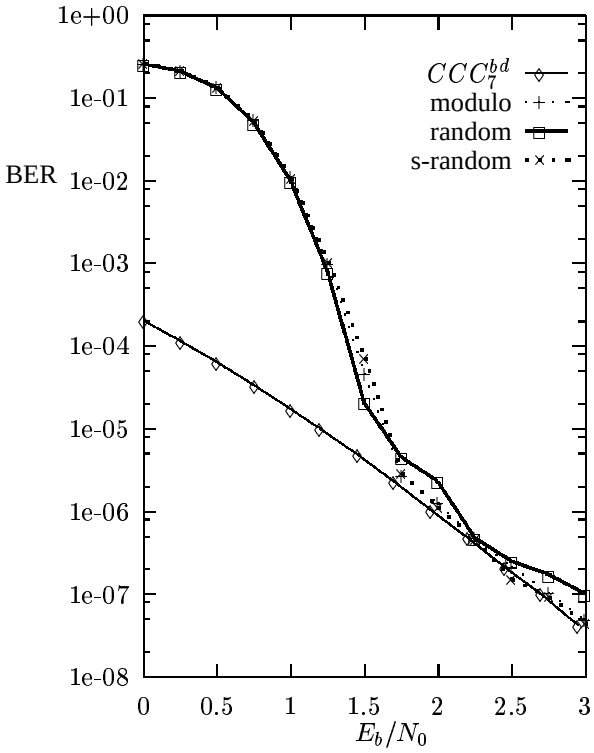

Figure 5.11: BER of coupled codes for different interleaving scheme.

5.5 Stop Criteria

Convolutional coupled codes achieve better performance as the number of iterations and interleavers size increases. However increasing the number of iteration needs much delay and computation for decoding. As the error rate approaches the performance limit of a given convolutional coupled code, any further iteration results in very little improvement. Therefore, it is important to devise an efficient criterion to stop the iteration process and prevent unnecessary computation and decoding delay. Figure 5.12 shows a typical variation of the bit error rate with the number of iterations, for the rate $1/2$ convolutional coupled code with a code length of 2030, operating at a SNR of $E_b/N_0 = 2$dB.

We see that a BER of about $1 \cdot 10^{-6}$ at this E_b/N_0 with 7 iterations can be achieved, and further iterations do not reduce this error rate dramatically. Conversely, significant performance penalties result from reducing the number of iterations below 7. For example, the BER rises about hundredfold to $1 \cdot 10^{-4}$ if only 4 iterations are used, and thus an $N = 4$ fixed stopping condition would not produce a very reasonable trade-off between performance and decoding speed for this code.

It is possible to improve the average decoding speed of the iterative decoder if a stopping rule with a variable number of iterations per frame is used instead [39] [60] [61]. For each decoded frame, the number of iterations performed is determined by the number of passes before a certain

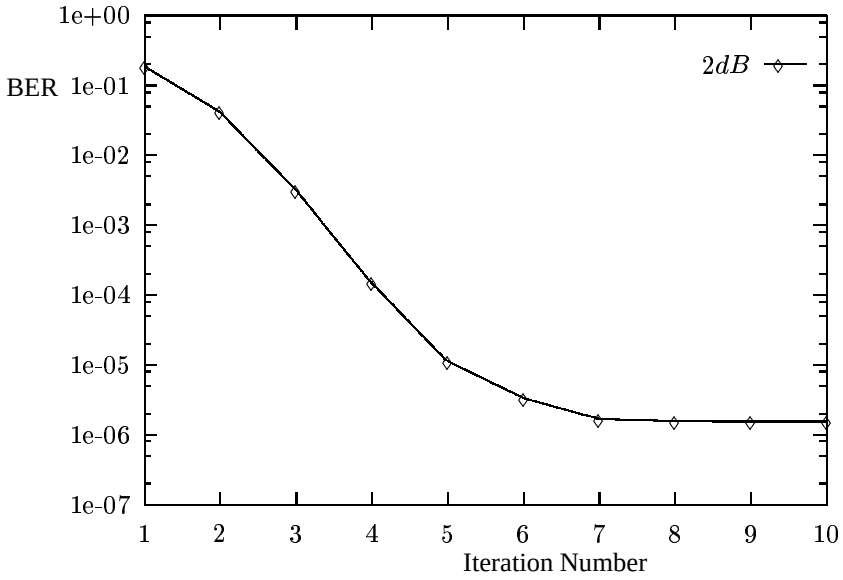

Figure 5.12: Bit error rate according to the iteration number and SNR.

condition or rule for stopping is satisfied. The stopping condition attempts to determine when a frame can be reliably decoded with no further iterations, and it is computed based on data available to the decoder during the decoding of each specific code word. More explicitly, at the end of each iteration, the decoder performs a check on the condition for stopping. If the condition is true, the iterative process on the frame is terminated, and the decoded sequence from the current iteration is sent to the output; otherwise, the iterative process continues to the next iteration. To prevent an endless loop should the stopping rule never be satisfied, we require that the decoder cease after a maximum number of iterations, N_{max}.

In the following we will present an adapted stop criteria on the convolutional coupled code introduced by Hagenauer *et al.* One such stopping criterion has been devised based on the *cross entropy* (CE) between the distributions of the estimates at the outputs of the inner and outer decoders at each iteration. This criterion is known as CE criterion. It effectively stops the iteration process with very little performance degradation.

For simplicity, we consider the ν outer SISO decoders as one single outer decoder and the k inner SISO decoders as one single inner decoder. Let $\mathbf{u} = (u_1, u_2, \cdots, u_K)$ be an information block of length $K = \nu k$ and $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,K})$ and $\mathbf{v}_o = (v_{o,1}, v_{o,2}, \cdots, v_{o,K})$ be the parity-check sequences of the inner and outer codes, respectively. Assuming BPSK transmission over an AWGN channel, $v_{i,t}$ and $v_{o,t}$ all take values from $+1, -1$ for $1 \leq t \leq K$. Let $\mathbf{y}_i = (y_{i,1}, y_{i,2}, \cdots, y_{i,K})$ and $\mathbf{y}_o = (y_{o,1}, y_{o,2}, \cdots, y_{o,K})$ be the received sequences from the inner and outer encoders, respectively. Then

$$\begin{aligned} y_{i,t} &= v_{i,t} + n_{i,t} \\ y_{o,t} &= v_{o,t} + n_{o,t}, \end{aligned}$$

where $n_{i,t}$ and $n_{o,t}$ are Gaussian random variables with zero mean and variance $\sigma^2 = \frac{N_0}{2E_s}$. In the following we use $\hat{\mathbf{u}} = (\hat{u}_1, \hat{u}_2, \cdots, \hat{u}_K)$ to denote the estimate of $\mathbf{u}$.

At the l-th iteration, let $L(\hat{u}_t)^{(l)}$ and $Le(\hat{u}_t)^{(l)}$ denote the a posteriori and the extrinsic LLR value of the estimated information $\hat{u}_t$. From (5.31) and (5.32) it is shown that

$$L_i(\hat{u}_t)^{(l)} = Le_o(\hat{u}_t)^{(l-1)} + Le_i(\hat{u}_t)^{(l)} \tag{5.36}$$

$$L_o(\hat{u}_t)^{(l)} = Le_i(\hat{u}_t)^{(l)} + Le_o(\hat{u}_t)^{(l)}. \tag{5.37}$$

The probability distribution $P(\hat{u}_t)^{(l)}$ at the output of each SISO decoder is given by

$$P(\hat{u}_t = \pm 1)^{(l)} = \frac{e^{\pm L(\hat{u}_t)^{(l)}}}{1 + e^{\pm L(\hat{u}_t)^{(l)}}}. \tag{5.38}$$

At iteration l, the CE between the distributions $P_i(\hat{\mathbf{u}})^{(l)}$ and $P_o(\hat{\mathbf{u}})^{(l)}$ of the outputs of the inner and outer decoders for an independently, identically distributed source $\mathbf{u}$ is defined as [39]

$$T(l) = E_{P_o^{(l)}}\left\{ log\frac{P_o(\hat{u})^{(l)}}{P_i(\hat{u})^{(l)}} \right\} = \sum_{t=1}^{K} E_{P_o^{(l)}}\left\{ log\frac{P_o(\hat{u}_t)^{(l)}}{P_i(\hat{u}_t)^{(l)}} \right\}, \tag{5.39}$$

where $E\{X\}$ denotes the expectation of the random variable X. The CE can be used to stop the iteration process in iterative decoding.

Let

$$\triangle Le_o(\hat{u}_t)^{(l)} = L_o(\hat{u}_t)^{(l)} - L_i(\hat{u}_t)^{(l)} = Le_o(\hat{u}_t)^{(l)} - Le_o(\hat{u}_t)^{(l-1)}.$$

Suppose that the decoding iteration converges, and at iteration l the decoding process can be terminated.

Based on some assumptions given in [39], the CE of (5.39) can be approximated as follows:

$$T(l) = \sum_t \frac{|\triangle Le_o(\hat{u}_t)^{(l)}|^2}{e^{|L_i(\hat{u}_t)^{(l)}|}}. \tag{5.40}$$

In [39], it is shown that when $T(l)$ drops to a value of $10^{-2} \sim 10^{-4}T(1)$, the distributions $P_i(\hat{\mathbf{u}})^{(l)}$ and $P_o(\hat{\mathbf{u}})^{(l)}$ are close enough to terminate the iterative process with very little performance degradation. The corresponding average numbers of iterations required by this criteria is shown in figure 5.13. The threshold $T(l)$ is set to $10^{-4}T(1)$.

We conclude that the average number of iterations for decoding convolutional coupled codes is comparable with those required for Turbo codes (see [60]).

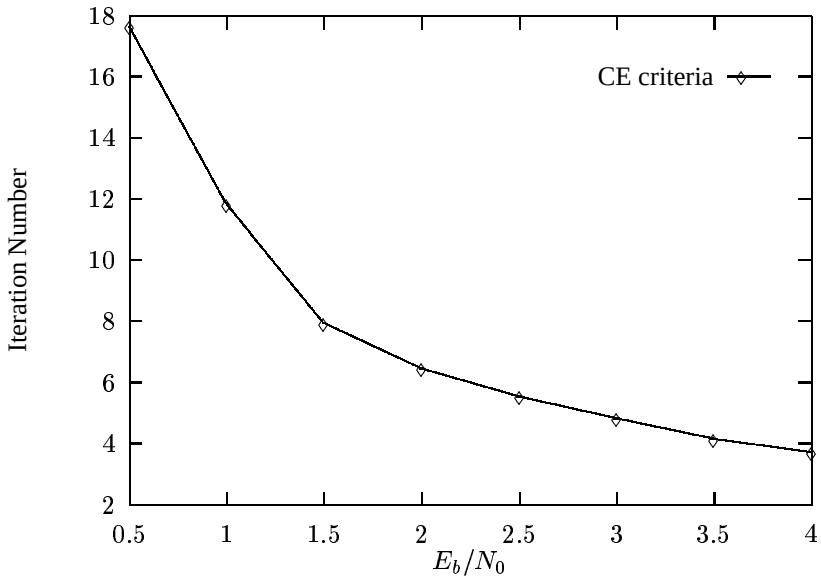

Figure 5.13: Average number of iteration of the rate $1/2$ convolutional coupled code of code length 2030 with 7 outer codes, using CE.

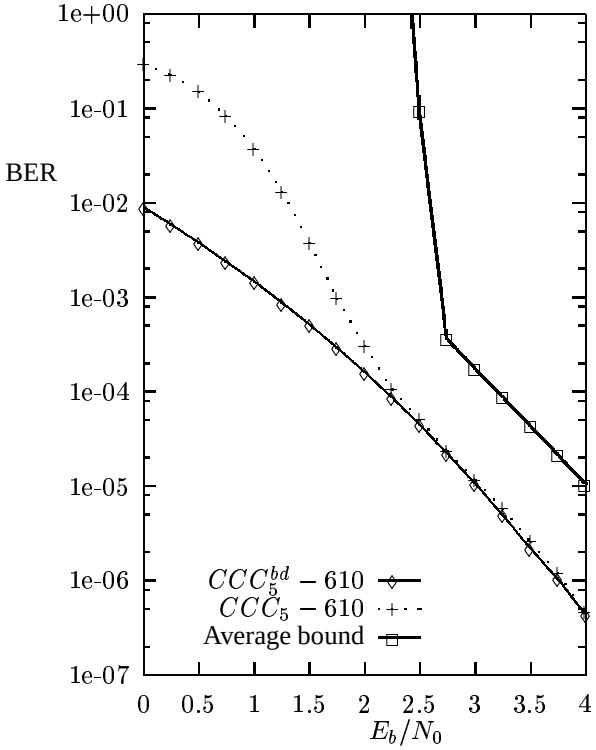

Figure 5.14: Simulation results for the code CCC_5 and CCC_5^{bd} compared to ML average bound.

5.6 Performance Bound

Since the introduction of Turbo codes, there has been a growing interest and demand on calculating tight performance bounds of concatenated coding systems with interleaving in order to estimate the quality of the (de)coding scheme besides large-scale simulation. A simple and well known upper bound on the word and bit error probability of ML-decoding is the so called union bound. This bound is tight above the SNR, corresponding to the channel cutoff rate R_0, but loose for lower SNRs and hence allows accurate prediction of the performance only for higher SNRs. The computation of the union bound requires knowledge of the weight distribution of the code. The presence of the interleavers in coupled code structures makes it very difficult to calculate the exact weight distribution of the code. The main obstacle is to assess the influence of the interleavers when computing the weight distribution of the over-all code. The idea of the averaging the performance of the codes over the structure of the interleavers was presented in [10] by introducing the concept of the "Uniform Interleaver (UI)" (see definition 4.1). The input-redundancy weight enumerating function (IRWEF) of a code C is given by

$$T^C(X,Y) = \sum_{w,h} A^C_{w,h} X^w Y^h, \tag{5.41}$$

where $A_{w,h}$ denotes the number of code words generated by an input information word of Hamming weight w whose parity-check bits have Hamming weight h, so that the over-all Hamming weight is $w+h$.

Consider now a convolutional coupled code obtained as described above, from the concatenation of ν outer terminated $1/2$ rate RSC codes and k inner block codes. Outer and inner encoder are linked through interleavers so that the information vectors of the outer codes are permuted before entering to the inner encoders. The parity-check bits generated by the inner encoder together with the parity-check bit from the outer encoders compose the over-all code word.

Now we want to obtain the IRWEF $T^{C_c}(X,Y)$ of the convolutional coupled code, starting from the knowledge of those of the outer RSC codes and the inner block codes. Let $T^C(X,Y)$ and $T^B(X,Y)$ be the IRWEF of an outer RSC code and an inner block code, respectively. For simplicity, we consider the ν outer RSC codes of code length $2k$ as one single code C_o of code length $\nu \cdot 2k$. The IRWEF of the codes C_o is given by

$$T^{C_o}(X,Y) = [T^C(X,Y)]^\nu, \tag{5.42}$$

and similar for the inner block codes

$$T^{C_i}(X,Y) = [T^B(X,Y)]^k, \tag{5.43}$$

where C_i is the code which results from k inner block codes. An additional difficulty is the presence of more than one interleaver in the code since we use for each outer code a different interleaver. To overcome this, we assume that only one over-all interleaver is employed. Thus the

uniform interleaver can be considered. If w is the Hamming weight of all the input word, and h_i, h_o are the weights of the parity-check bits introduced by C_i and C_o, respectively, the weight of the corresponding over-all code word is $h_i + h_o$. As a consequence we can evaluate the average weight enumerating function of the coupled code C_c using the uniform interleaver:

$$A_{w,j}^{C_c} = \sum_{j=h_i+h_o} \frac{A_{w,h_i}^{C_i} \cdot A_{w,h_o}^{C_o}}{\begin{pmatrix} \nu k \\ w \end{pmatrix}}. \tag{5.44}$$

Using the union bound, an upper bound to the bit error probability for ML-soft decoding of the code for transmission over an AWGN channel can be computed in form:

$$P_b(e) \leq \frac{1}{2\nu k} \sum_{d=d_{min}}^{2\nu k} B_d^{C_c} erfc(\sqrt{R_c d \frac{E_b}{N_0}}), \tag{5.45}$$

where $B_d^{C_c}$ is the total number of non-zero information bits on all weight d paths, i.e.,

$$B_d^{C_c} = \sum_{w=1}^{\nu k} w A_{w,d-w}^{C_c}, \tag{5.46}$$

and R_c is the over-all code rate. By comparing this upper bound with the simulation results with a chosen modulo interleaving, we note that the deterministic interleaving offers performance far below the average one (see figure 5.14). This is due to the fact that the use of one over-all interleaver for all information vectors of the outer codes deteriorates the performance of the coupled code. Therefore, the performance obtained with the uniform interleaver is worse than the simulated performance using a separate interleavers for each row. This confirms the fact that the choice of an appropriate interleaving plays a decisive role in the performance of coupled codes.

5.7 Error Performance Simulations

From equations (5.33) and (5.35), increasing of the coupling factor $\mathcal{K}$ leads to a higher effective free distance. In order to investigate this, we consider the rate $\simeq 1/2$ convolutional coupled codes CCC_5, CCC_7 and CCC_9 obtained from the constructions with 5, 7 and 9 outer RSC codes, respectively. We use likewise a coupled matrix with two zeros in each row. The generator polynomials of the outer RSC codes are identical for all three codes and are $g_1 = 15_8$ and $g_2 = 17_8$. To make a fair comparison, the code lengths of the different codes should be about the same size (approximately 2030 symbols).

The simulation results are presented in figure 5.15. We observe that increasing the number of outer codes ν, which corresponds to the increasing of the coupling factor, leads to lower bit

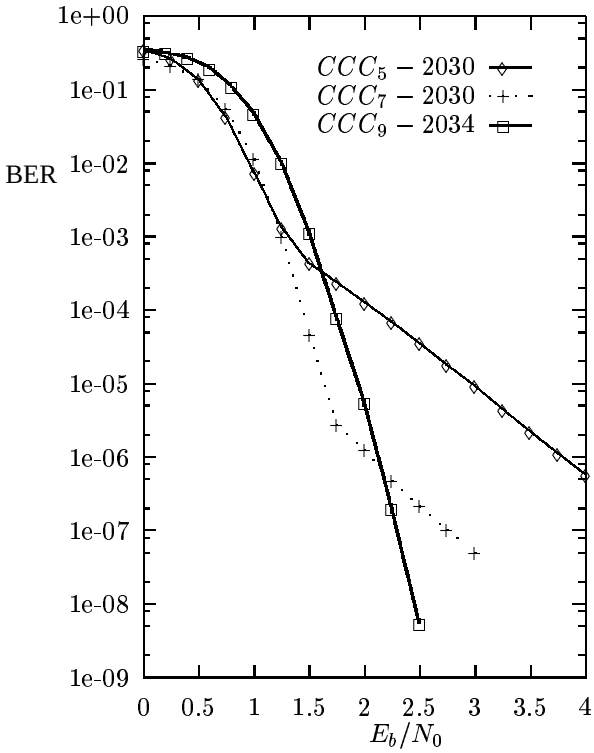

Figure 5.15: Performance comparison of rate $\simeq 1/2$ convolutional coupled codes, CCC_5, CCC_7 and CCC_9 (code length $\simeq$ 2030 symbols).

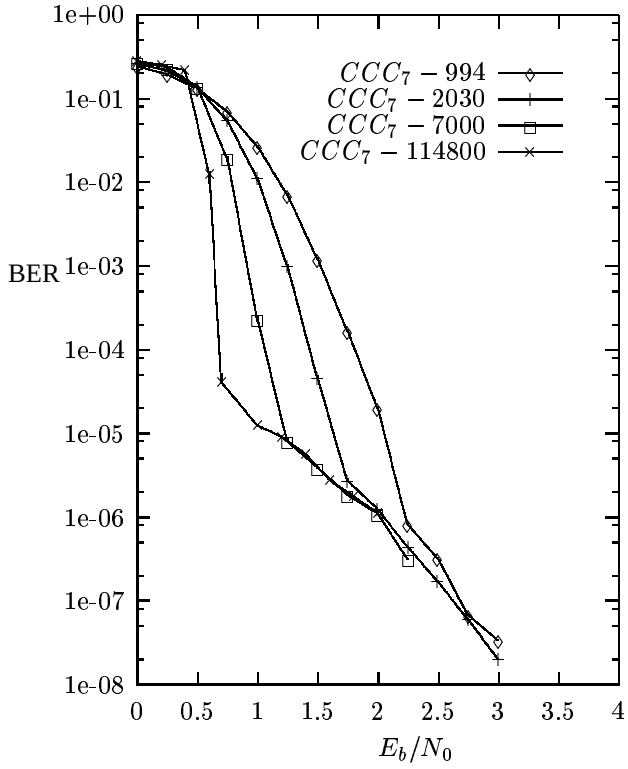

Figure 5.16: Performance of rate $\simeq 1/2$ convolutional coupled codes CCC_7 for different code lengths.

error rates for moderate and high SNRs. Since codes with large minimum distance perform asymptotically better than codes with small one, we conclude that increasing the coupling factor improves the distance properties. Moreover the results indicate that a small number of outer codes performs well at low SNRs. That means that the convergence behavior of codes with small coupling factor may be better that those with large one. This will be more discussed in section 6.1.

Let us now consider the performance of different convolutional coupled codes when varying the code length. In figure 5.16 we present the bit error probability of convolutional coupled codes employing the same number of outer codes with different interleaver lengths. Enlarging the code length leads to an improvement in performance for a wide range of bit error probability. This confirms the fact that increasing the interleaver length for a given concatenated code leads to better performance. Moreover, we note that the "error floor" of all codes with different code lengths is close by the lower bound obtained from the effective boundary coupled code.

The figure 5.17 indicates that as the number of iterations increases, the performance of the convolutional coupled code improves. This is expected based on the results of Turbo codes. However, after a certain number of iterations (in this case 13), the performance appears to saturate and does not improve with iterations which is also expected from section 5.5.

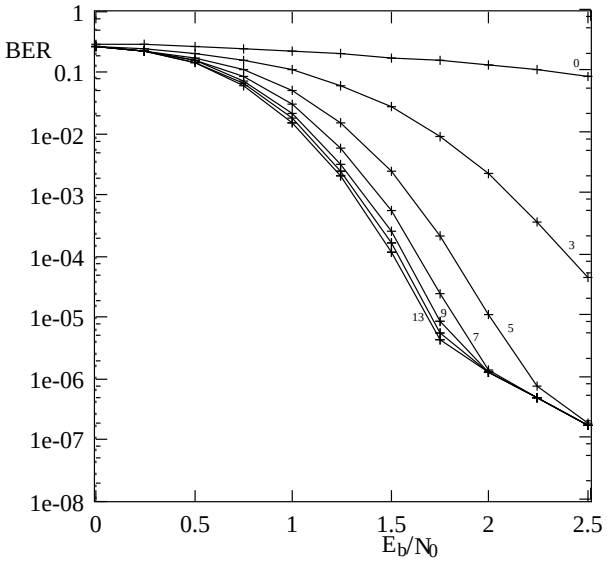

Figure 5.17: BER performance of the iterative decoder after different number of iterations.

5.7.1 Comparison between the Performance of Convolutional Coupled Codes and Turbo Codes

As the construction of convolutional coupled codes is very similar to that of Turbo codes, we present a comparison of these two code classes. In order to provide a fair comparison, the convolutional coupled codes and Turbo codes are constructed with the same memory $m = 3$ RSC codes. The Turbo code considered in this comparison is specified for the UMTS standard [65]. Both codes are iteratively decoded by exchanging extrinsic information between the decoders of the constituent codes. All of these codes are of rate $R_c \simeq 1/2$ and code length $N \simeq 2000$.

Although the encoders and the decoders of these two code classes look similar, there are some remarkable differences:

- The RSC codes of the convolutional coupled codes are coupled by the inner block codes, whereas those of the Turbo codes are coupled by encoding the same (interleaved) info bits.

- The structure of the convolutional coupled codes allows to give a lower bound on the error performance, but no bounds are known for Turbo codes at present[1] .

- The delay caused by the interleaver length in the convolutional coupled codes amounts only $1/\nu$ of the delay generally caused in Turbo codes, since the ν outer codes may be simultaneously decoded.

[1]Only the BER of long Turbo codes employing uniform interleaving can be approximated for high SNR [10].

- For a fixed code length and a fixed generator polynomial of the RSC code, the performance of the Turbo codes is influenced by the interleaver structure. The performance of the convolutional coupled code is subject to the number of outer codes and to the (inner) block code.

- The convolutional coupled codes does not contain any systematic bits; nevertheless, iterative decoding works. In the contrary, by Turbo codes a certain number of systematic bits are necessary for the iterative decoder to converge at low SNR [47].

In the following, the codes are compared with respect to their error rate performances (see figure 5.18 and 5.19). Figure 5.18 shows that convolutional coupled codes need a certain number of outer codes ($\nu = 7, 9 \cdots$) to achieve good performances. Compared with the Turbo code, convolutional coupled codes with $\nu = 7$ outer codes offer nearly the same performance at small and moderate SNR. In addition, we observe that the flattening of the performance curve caused by the free distance asymptote, as in the Turbo code case [52], does not appear to be very strong for the convolutional coupled codes. Moreover, figure 5.18 shows that increasing the number ν of outer codes from 5 to 7 shifts down the "flattening region" by a factor of 10^{-2}, similar improvements can be expected if ν further increased. This makes the convolutional coupled codes with a high number of outer codes preferable for low BER at high SNR. In this case, for small and moderate SNRs, Turbo codes outperforms slightly convolutional coupled code with 9 outer codes. Selection of the outer RSC codes in the convolutional coupled codes in chapter 7 will show, that there is a code CCC_9 with a specific outer codes generator polynomials, which achieves a similar performance as Turbo codes in the "waterfall region" and keep the same behavior in the "error floor region" as the simulated convolutional coupled code CCC_9 in this comparison. Consequently, the distance properties of the convolutional coupled codes for large number of outer codes are better than those of Turbo codes and this without loss in the convergence behavior of the iterative decoder. That means, that convolutional coupled codes with a certain coupling factor and specific outer code polynomials have the potential of being a realistic alternative to Turbo codes.

Comparing the word error rates, figure 5.19 shows that the convolutional coupled code outperform the Turbo codes in the "error floor region", while the performance in the "waterfall region" is only slightly worse.

5.8 Conclusion

In this chapter we have investigated several aspects of the so called convolutional coupled code. The structural properties of the generator matrix of the coupled code has been investigated. The minimum distance of the convolutional coupled code is lower and upper bounded and we have introduced the term of the effective free distance. We have shown that the performance of the convolutional coupled code is strictly related to the defined effective free distance, which seems to be the most important parameter for the construction of this codes. For a large range of SNR,

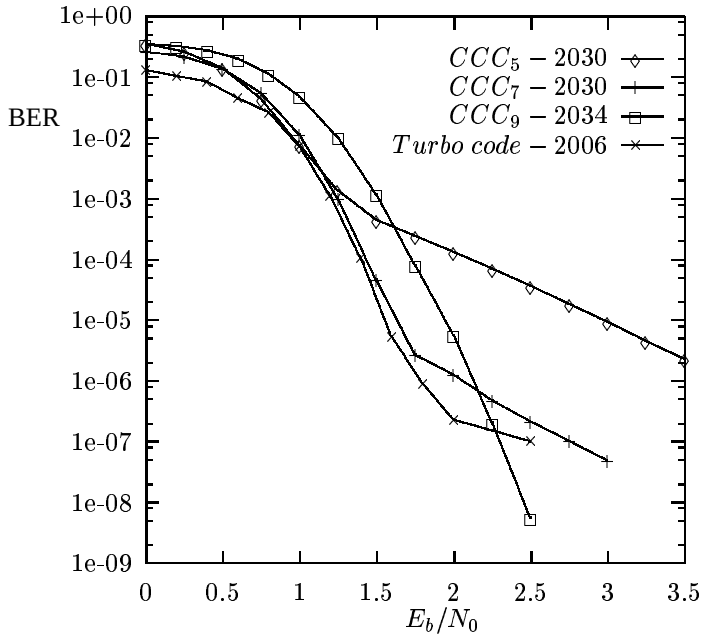

Figure 5.18: Comparison of bit error rate performances of Turbo codes and convolutional coupled code with different number of outer codes (code length $N \simeq 2000$).

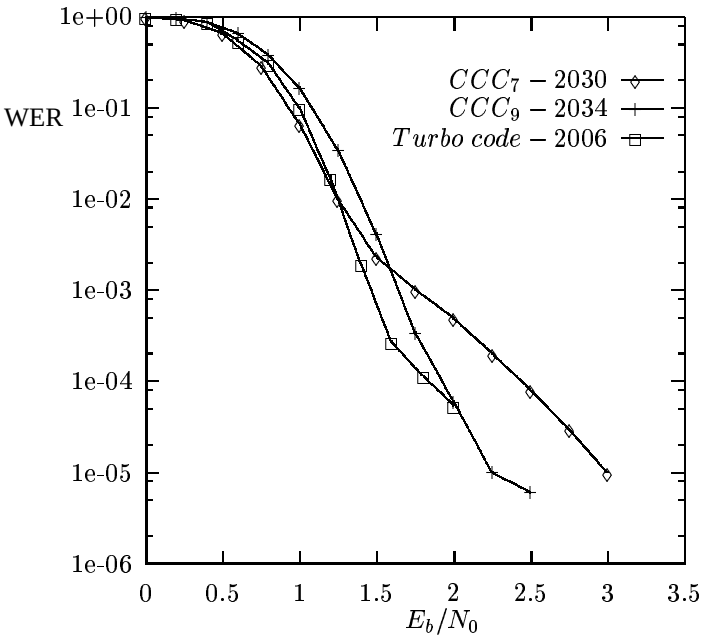

Figure 5.19: Comparison of word error rate performances of Turbo codes and convolutional coupled code with different number of outer codes (code length $N \simeq 2000$).

the behavior of the convolutional coupled code is determined by this parameter. Using soft-input soft-output iterative decoders and only extrinsic information as a priori information in the iteration steps, this construction yields high coding gain at bit error in the range $10^{-5} - 10^{-8}$. It was shown that the iterative decoding algorithm works not only for systematic, but also for non-systematic codes. Indeed, convolutional coupled codes offer competitive error rates especially those with a high number of outer codes. Also, it was shown that it is possible to improve the average decoding speed of the iterative decoder if stopping rule with a variable number of iterations per frame is used. Thereby, we have shown that the average number of iterations for decoding convolutional coupled codes is comparable with those required for Turbo codes; consequently the difference in the complexity to the Turbo codes can only emerge from the inner SISO block decoders. It was also shown that the average bit error probability using the uniform interleaver which permit an easy derivation of the weight enumerating function of the coupled code, is not a significant bound for coupled codes with deterministic interleavers.

Chapter 6

Iterative Decoding Trajectories

As the iterative decoding proceeds, the statistical dependencies increase between decoder inputs and outputs. In [14], Brink presents a method for investigating the convergence of iterative decoding based on the mutual information between the transmitted information symbols and the decoder a priori inputs/extrinsic outputs, for different number of decoding iterations. With this methodology, comprising the technique of using *extrinsic information transfer charts*, Brink succeeds in predicting the position of what is referred to as the "Turbo-cliff", that is, the E_b/N_0 for which the iterative decoding starts to converge. The methodology of extrinsic information transfer charts is applied to the convolutional coupled codes and is presented in the following paragraphs. For simplicity, we consider the ν outer SISO decoders as one single outer decoder and the k inner SISO decoders as one single inner decoder. The same applies to the ν interleavers, which are combined to one over-all interleaver.

6.1 Extrinsic Transfer Characteristics

A simplified iterative decoder for convolutional coupled codes is shown in figure 6.1. The variables P_i, P_o, D_i, A_i, E_i, A_o, E_o and D_o denote Log-likelihood ratios. For each iteration, the SISO inner decoders take channel observation P_i and a priori knowledge A_i on the information bits and outputs a posteriori soft values D_i. The extrinsic information $E_i = D_i - A_i$ is passed through the bit deinterleavers to become the a priori input A_o for the SISO outer decoders. The outer decoders feed back extrinsic information $E_o = D_o - A_o$ which becomes the a priori knowledge A_i for the SISO inner decoders.

The idea is to predict the behavior of the iterative decoder by solely looking at the input/output relations of the individual constituent decoders [14]. For this, we make use of the observation obtained by simulation that the probability density function of the extrinsic output values E (a priori values A for the next decoder respectively) approaches a Gaussian distributions with increasing number of iterations. Hence, it seems appropriate to model the a priori input A to the constituent decoder by applying an independent Gaussian random variable n_A with zero mean

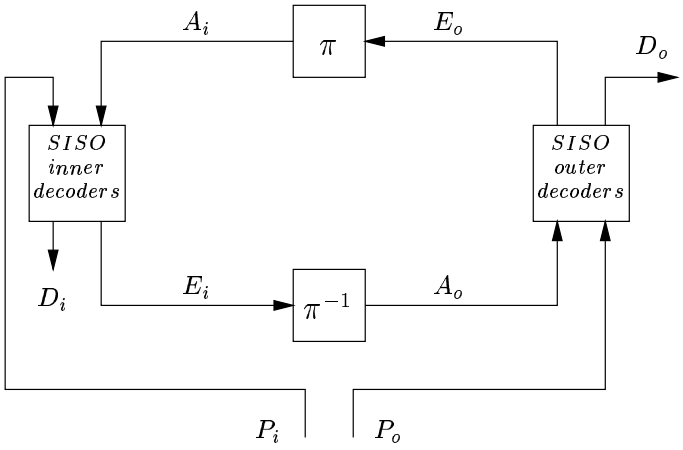

Figure 6.1: A simplified iterative decoder for convolutional coupled codes.

and variance σ_A^2. In conjunction with the known transmitted inner information bits $u \in \{\pm 1\}$ we write

$$A = \mu_A \cdot u + n_A. \tag{6.1}$$

Since A is an L-value based on a Gaussian distribution it can be shown [39] that μ_A is given by

$$\mu_A = \frac{\sigma_A^2}{2}. \tag{6.2}$$

With (6.2) the conditional probability density function belonging to the L-value A is

$$P_A(\xi|U=u) = \frac{e^{-\frac{(\xi - \frac{\sigma_A^2}{2}u)^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A} \tag{6.3}$$

with $u \in \{\pm 1\}$.

To measure the information contents of the a priori knowledge, the mutual information $I_A = I(U;A)$ [40] between the information bits U and the L-values A is used.

$$I_A = \frac{1}{2} \cdot \sum_{u=-1,1} \int_{-\infty}^{+\infty} P_A(\xi|U=u) \cdot ld \frac{2P_A(\xi|U=u)}{P_A(\xi|U=-1) + P_A(\xi|U=1)} d\xi \tag{6.4}$$

$$0 \leq I_A \leq 1.$$

With (6.3), equation (6.4) becomes

$$I_A(\sigma_A) = 1 - \int_{-\infty}^{+\infty} \frac{e^{-\frac{(\xi - \frac{\sigma_A^2}{2})^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A} \cdot ld[1 + e^{-\xi}]d\xi. \tag{6.5}$$

In order to simplify the notation [14]

$$J(\sigma) := I_A(\sigma_A = \sigma) \tag{6.6}$$

with

$$\lim_{\sigma \to 0} J(\sigma) = 0, \quad \lim_{\sigma \to \infty} J(\sigma) = 1, \sigma > 0. \tag{6.7}$$

The function $J(\sigma)$ can not be expressed in closed form. As it turns out from the numerical integration, $J(\sigma)$ is monotonically increasing and thus the inverse function exists.

$$\sigma_A = J^{-1}(I_A). \tag{6.8}$$

The mutual information is also used to quantify the extrinsic output $I_E = I(U, E)$.

$$I_E = \frac{1}{2} \sum_{x=1,-1} \int_{-\infty}^{+\infty} P_E(\xi|U = u) \cdot ld \frac{2P_E(\xi|U = u)}{P_E(\xi|U = -1) + P_E(\xi|U = 1)} d\xi \tag{6.9}$$

$$0 \leq I_E \leq 1$$

Viewing I_E as a function of I_A and the E_b/N_0-value, the extrinsic information transfer characteristic is defined as

$$I_E = T(I_A, E_b/N_0) \tag{6.10}$$

or, for fixed E_b/N_0, just

$$I_E = T(I_A). \tag{6.11}$$

To compute $T(I_A, E_b/N_0)$ for the desired $I_A, E_b/N_0$-input combination, the distribution P_E of (6.9) are most conveniently determined by Monte Carlo simulation (histogram measurements). To this purpose, the independent Gaussian random variable of (6.1) is applied in conjunction with the known transmitted bits u. Note that a certain value of I_A is obtained by appropriately choosing the parameter σ_A according to (6.8).

The transfer characteristics based on the mutual information prove to be quite robust against changes in the shape of input distributions of A, owing to the robustness of the entropy measure. In an iterative decoder the actual distributions P_A differ significantly from Gaussian for the very first few iterations. The extrinsic distributions at the output of the outer decoder after zero, 5, 10 and 20 iterations, are depicted in figure 6.2. Obviously, the shape of P_E is very different from the Gaussian at the beginning of the iteration process; after 5 iterations it approaches the shape of

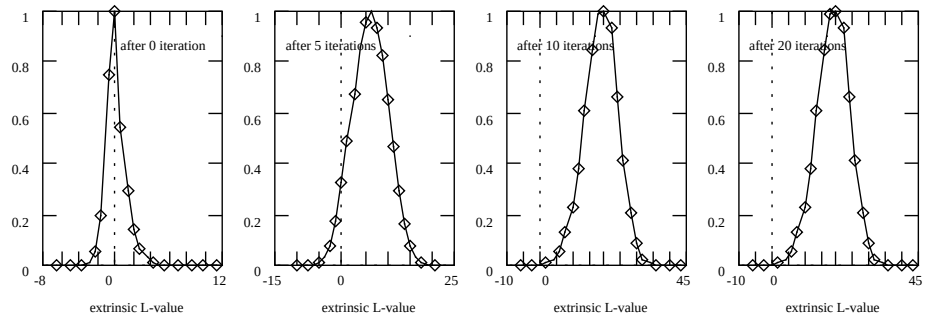

Figure 6.2: Measured shape of the extrinsic L-value distributions at the output of the outer decoder after different number of iterations (ordinate is normalized to one).

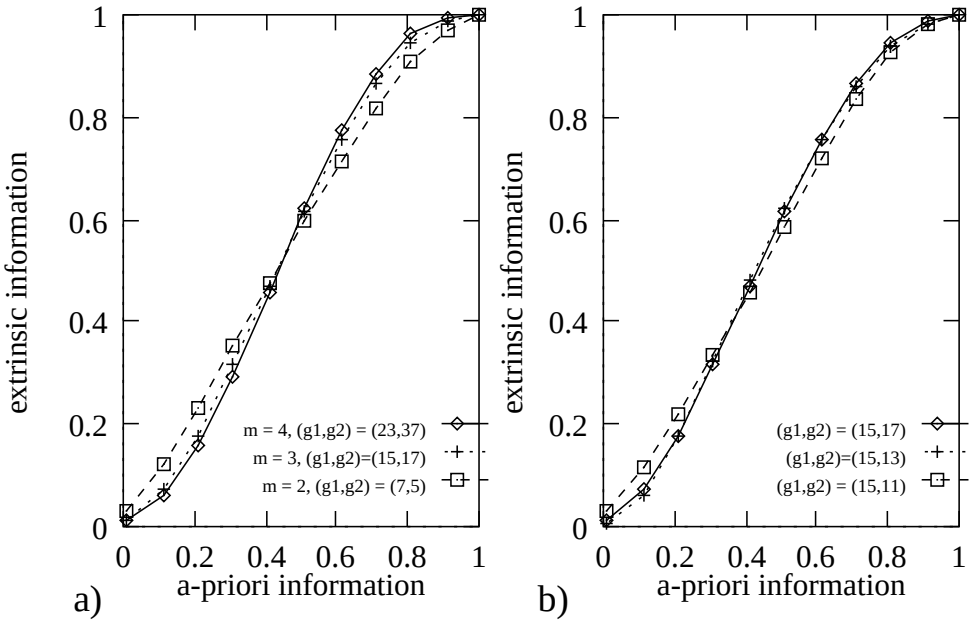

Figure 6.3: Exit information transfer characteristic of soft input/soft output outer decoder for rate $1/2$ convolutional codes, $E_b/N_0 = 1$dB, for a) different code memory b) memory 3, different generator polynomials.

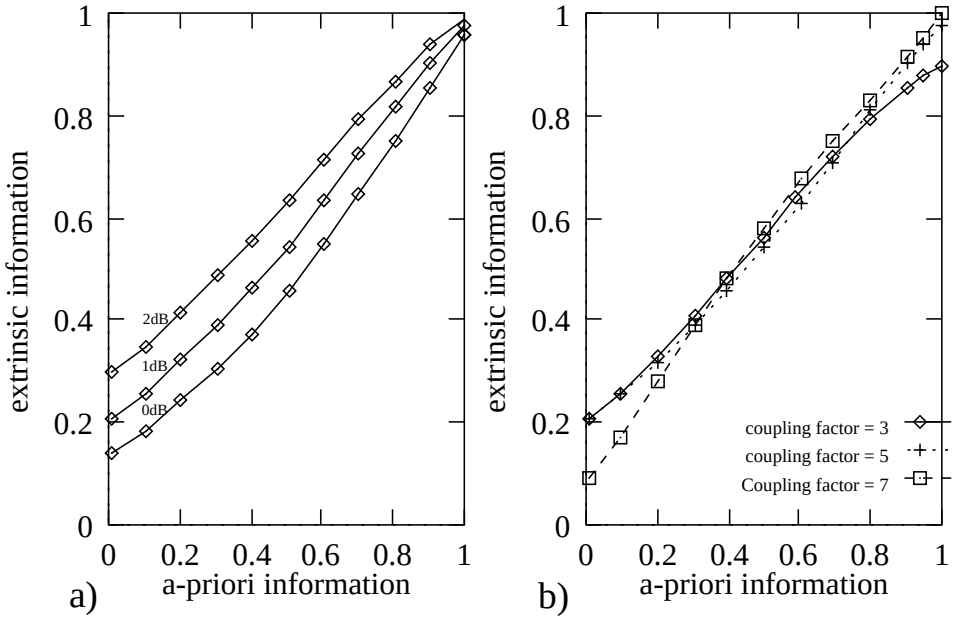

Figure 6.4: Extrinsic information transfer characteristics of soft input/soft output inner decoder, a) E_b/N_0 of channel observation as parameter to curves b) with different coupling factor ($E_b/N_0 = 1$dB).

a Gaussian distribution very closely; for more iterations, the maximum value is shifted towards higher L-values, and the shape becomes asymmetric again.

In figure 6.3a), the influence of using convolutional encoders with different number of memory elements is shown. For a priori inputs with small mutual information with the transmitted sequence (i.e. the lower part of the abscissa), the best decoding progress is obtained with the codes corresponding to a small number of memory elements. On the other hand, with a priori inputs having high mutual information to the transmitted sequence, the situation is the opposite. Then, the best decoding progress is obtained for the codes that correspond to a larger number of memory elements.

For a given number of memory elements in the encoder, the extrinsic information transfer characteristic depends on the generator polynomials. Figure 6.3b shows the characteristics for three memory-3 constituent encoders using generator polynomials $(15, 17)_8$, $(15, 13)_8$ and $(15, 11)_8$. It is noted that at early decoding stages, that is for mutual information $I_A < 0.5$, the largest decoding progress for each half iteration is achieved for the $(15, 11)_8$-code. Equivalently, it is observed that if E_b/N_0 is decreased, the characteristic of $(15, 17)_8$-code reaches the diagonal first, which corresponds to a cease in the iterative decoding performance. In figure 6.4a, the influence of different SNRs is shown. An increase in E_b/N_0 results in increased mutual information at the inner decoders output, for a given mutual information at the decoder input[1]. This correspond to improved decoding performance. In figure 6.4b, transfer characteristics of inner block codes

[1]The extrinsic outputs from the previous half-iteration act as a priori inputs.

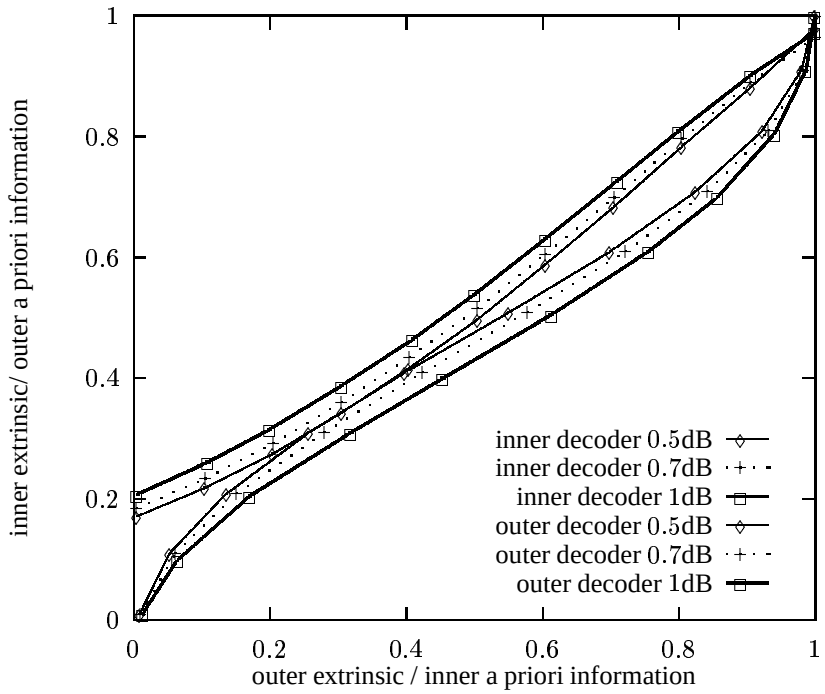

Figure 6.5: EXIT charts of rate $\simeq 1/2$ convolutional coupled code for $E_b/N_0 = 0.5, 0.7$ and 1dB ($\mathcal{K} = 5$, $(g_1, g_2) = (15, 17)_8$).

with different coupling factors at fixed $E_b/N_0 = 1$dB are depicted. We consider in this cases coupling matrices which contain two zeros in each row. We observe that the extrinsic output at the beginning (A-priori information $I_A \simeq 0$) for coupling factor $\mathcal{K} = 7$ is smaller than those for $\mathcal{K} = 3$ and 5. Obviously the one with the best performance at the beginning is the worst at the end (A-priori information $\simeq 1$). This indicates, that the convergence properties in the "turbo cliff region" of convolutional coupled codes with coupling factor 3 and 5 is better than those with 7.

6.2 Extrinsic Information Transfer Chart

If the previously determined transfer characteristics of inner and outer component codes are plotted in a single figure, the so-called extrinsic information transfer (EXIT) chart is obtained. EXIT charts provide a visualization of the exchange of extrinsic information between the two component decoders. Convergence of iterative decoding is possible if the transfer characteristics do not intersect. Hence, the signal to noise ratio E_b/N_0 at which a turbo cliff occurs, i.e. a rapidly decreasing bit error rate, can be estimated very precisely from the EXIT chart.

Exemplary we want to construct a rate $\simeq 1/2$ convolutional coupled code with 7 outer codes and $\mathcal{K} = 5$. The generator polynomials of the outer codes are $(15, 17)_8$. With this parameters the iterative decoder leads to a turbo cliff at about 0.7dB. In the EXIT chart of figure 6.5 it can be observed that the transfer characteristics of the inner and outer codes do not intersect at signal

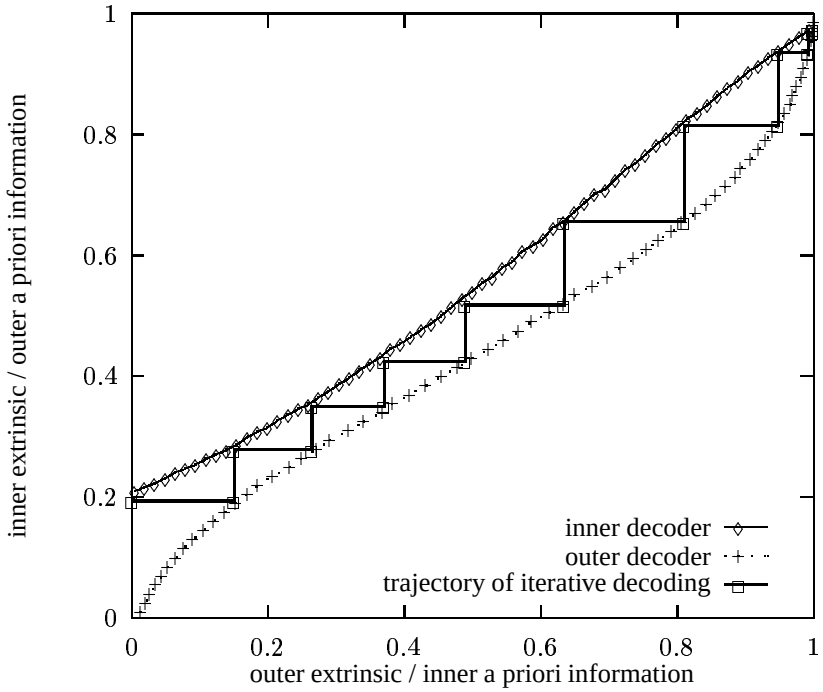

Figure 6.6: Simulated trajectories of iterative decoding at $E_b/N_0 = 1$dB.

to noise ratio of $E_b/N_0 \geq 0.7$dB, whereas no convergence of iterative decoding is possible for lower signal to noise ratios.

6.2.1 Trajectories of Iterative Decoding

Supposed that the independence and Gaussian assumption are valid for modeling extrinsic and a priori information, the transfer characteristics should approximate the true behavior of the iterative decoder. The decoding trajectory can be graphically obtained by a drawing of the zigzag path bounded by the decoder transfer characteristics into the EXIT chart.

For a fixed E_b/N_0, let l be the iteration index. For $l = 0$ the iteration starts at the origin with zero a priori knowledge of the inner decoder $I_{A_{i,0}} = 0$. At iteration l, the extrinsic output of the inner decoder is $I_{E_{i,l}} = T_i(I_{A_{i,l}})$. $I_{E_{i,l}}$ is delivered to the outer decoder as $I_{A_{o,l}} = I_{E_{i,l}}$. The extrinsic information of the outer decoder is $I_{E_{o,l}} = T_o(I_{A_{o,l}})$, which is fed back to the inner decoder as a priori knowledge $I_{A_{i,l+1}} = I_{E_{o,l}}$ for the next iteration. The iteration stops is $T_i(I_{E_{o,l}}) = T_o^{-1}(I_{E_{o,l}})$, which corresponds to an intersection of both characteristics in the EXIT chart. We note that the interleaving does not change the mutual information.

Figure 6.6 simulated for a convolutional coupled code of code length 112000, $\nu = 7$, $\mathcal{K} = 5$, and generator polynomials of outer RSC codes with 3 memory elements, shows the trajectory of the iterative decoding at $E_b/N_0 = 1$dB. The trajectory converges after about 9 iterations. This figure shows that the EXIT chart is accurate and can predict the trajectory of the iterative decoder.

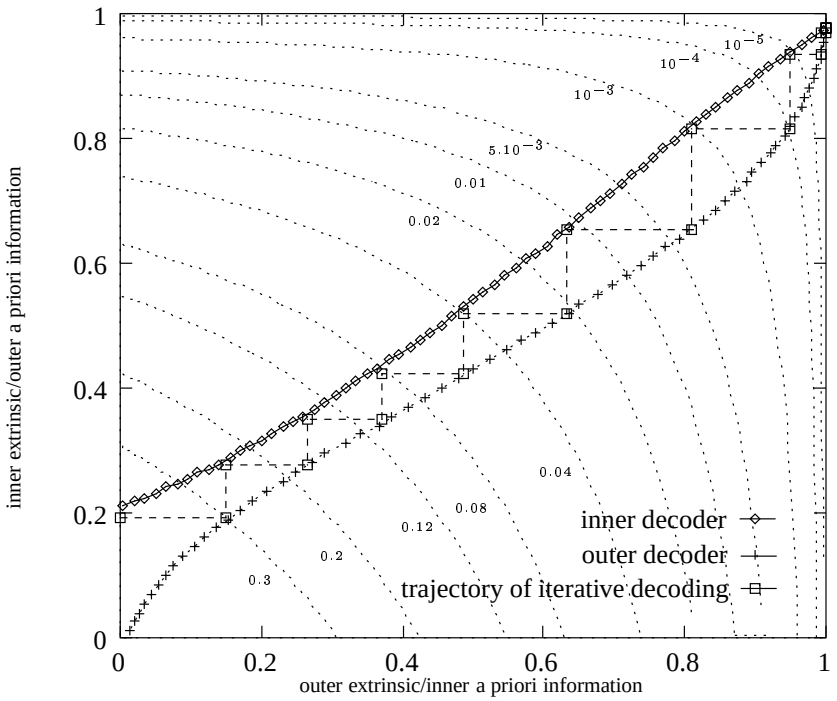

Figure 6.7: Simulated trajectories of iterative decoding at $E_b/N_0 = 1$dB with BER scaling as contour plot.

6.2.2 Estimation of the BER from the EXIT chart

The EXIT chart can be used to obtain an estimate of the BER after an arbitrary number of iteration. For both decoder, the soft output on the information bits can be written as $D = A + E$. The soft output D is assumed to be Gaussian distributed with variance σ_D^2 and mean value $\mu_D = \sigma_D^2/2$, compare to (6.1), (6.2) then the bit error probability is

$$P_b \simeq \frac{1}{2} erfc(\frac{\mu_D}{\sqrt{2}\sigma_D}) = \frac{1}{2} erfc(\frac{\sigma_D}{2\sqrt{2}}). \tag{6.12}$$

Assuming independence it is

$$\sigma_D^2 = \sigma_A^2 + \sigma_E^2. \tag{6.13}$$

Applying equation (6.8) the variances σ_A^2 and σ_E^2 are calculated as

$$\sigma_A^2 = J^{-1}(I_A)^2, \sigma_E^2 = J^{-1}(I_E)^2. \tag{6.14}$$

Finally, with equations (6.12), (6.13) and (6.14) the result is

$$P_b \simeq \frac{1}{2} erfc(\frac{\sqrt{J^{-1}(I_A)^2 + J^{-1}(I_E)^2}}{2\sqrt{2}}). \tag{6.15}$$

With (6.15) an estimate on the bit error probability P_b can be calculated from the EXIT chart.

The figure 6.7 shows the transfer characteristics and the corresponding simulated decoding trajectory at 1dB for the coupled code. Additionally, the BER scaling according to equation (6.15) is given as contour plot.

6.3 Conclusion

The mutual information between systematic bits and extrinsic output of constituent decoders was found to be useful measure for gaining insight into the convergence behavior of iterative decoding. The EXIT analysis is made precise by introducing a model of the decoding process and specifying the desired information-theoretic quantities. The model applies to iterative decoding of parallel concatenated (Turbo), serially concatenated, and in this work of convolutional coupled codes. The influence of code memory, code polynomials as well as different inner codes on the convergence behavior is studied. Furthermore, the model lets one drive properties of EXIT charts that are useful for guiding the code design.

Chapter 7

Choosing Constituent Codes

So far, the optimization of concatenated coding schemes are done with respect to asymptotic slopes of the error probability curve for moderate to high signal to noise ratios or with respect to the distance properties of the codes.

In the previous chapters, two fundamental properties that govern the performance of a convolutional coupled code were exploited: the effective free distance and the performance of iterative decoding. In this chapter, the same approach is used to investigate selection criteria for the choice of constituent codes. This review is not exhaustive, but indicative of properties that should be considered in the process of designing convolutional coupled codes. We have seen previously that increasing the coupling factor improves the distance properties of the coupled code and thus leads to noticeable improvement in performance for a wide range of error probabilities. Since changing the RSC outer codes effects generally the distance properties of the convolutional coupled code, the choice of these is significant. In the previous chapter we have shown that the EXIT chart can be used as a tool to provide design guidelines of convolutional coupled codes. In this chapter we intend to find the best RSC outer codes for a given memory with respect to the effective free distance of the over-all code. Thereafter we search constituent codes based on the EXIT chart technique to yield the best coupled code under certain specifications.

7.1 Choice of the RSC Outer Codes

7.1.1 Code search based on the effective free distance

In the following we will make some considerations on the choice of RSC outer component codes.

Although the behavior of convolutional coupled codes for very low values of the SNR is not well understood yet, we believe that in this region, below the cutoff rate, the effects induced by changing the RSC outer component codes will generally not be very great.

Nonetheless, for SNRs above $1.5-2$dB, the choice of the encoder is important. The performance of convolutional codes depends highly on the choice of generator polynomials. The reason why

good convolutional codes are not necessarily good constituent codes in the convolutional coupled codes is due to the presence of the coupling matrix whose coupling factor influences the distance properties of the code. Therefore a large amount of effort has been spent on finding convenient outer RSC codes. A good selection criteria to improve the performance in the "error floor region" is to maximize the effective free distance.

We will limit ourselves to the case of RSC codes with rate $1/2$. Generalization to rate $1/n$ RSC outer codes is straightforward. The generator matrix of a rate $1/2$ RSC encoder with memory m has the form

$$G(D) = [1, \frac{n(D)}{d(D)}], \tag{7.1}$$

where $d(D)$ is a polynomial of degree m. We define w_{min} as the weight of the information sequence and h_{min} as the weight of the parity sequence of an RSC code word such that

$$d_{free,eff} = \mathcal{K} w_{min} + h_{min}. \tag{7.2}$$

One of the particularities of terminated RSC encoders is that the minimum input weight is equal to two [8].

The design objective here is to obtain a $d_{free,eff}$ as large as possible. For a given memory, we assume that the coupling factor $\mathcal{K}$ is big enough that the addend $\mathcal{K} w_{min}$ in (7.2) does not change when varying the generator polynomials and w_{min} is mostly 2. Thus, in order to maximize $d_{free,eff}$, the weight h_{min} should be as large as possible. Let $1 + D^L$, for some finite L, be the shortest input sequence of weight 2 that generates a finite-length code word. The resultant parity sequence is

$$y(D) = (1 + D^L)\frac{n(D)}{d(D)}.$$

Since $y(D)$ is of finite length, $d(D)$ must be periodic with period L, where the period of a polynomial $P(D)$ is defined as the minimum exponent r, which fulfills the following condition:

$$1 + D^r = 0 \; mod \; P(D).$$

Increasing the period L increases the length of the shortest weight 2 input sequence that generates a finite-length code word. Intuitively, one would expect that increasing its length would result in the code word gaining weight. That is on average, half of the added bits would be ones. The period is maximized, that is, $K = 2^m - 1$, when $d(D)$ is a primitive polynomial. Thus we suggest the following procedure for choosing the generator matrix $G(D)$ of a recursive systematic code with memory m:

1. Choose $d(D)$ as primitive polynomial of degree m.

m	$d(D)$	$n(D)$	w_{min}	h_{min}	$d_{free,eff}$
2	7	5	2	4	10
3	15	17,13	2	6	12
4	31	37	3	5	14
	23	37	3	5	14

Table 7.1: Best rate $1/2$ RSC outer codes with primitive feedback polynomials for coupling factor 3 with respect to the effective free distance.

m	$d(D)$	$n(D)$	w_{min}	h_{min}	$d_{free,eff}$
2	7	5	2	4	14
3	15	17,13,11	2	6	16
4	31	37	2	10	20
	23	37	2	10	20

Table 7.2: Best rate $1/2$ RSC outer codes with primitive feedback polynomials for coupling factor 5 with respect to the effective free distance.

2. Choose $n(D)$ that maximize the effective free distance of the convolutional coupled code.

The results for $\mathcal{K} = 3, 5, 7$ and $m = 2, 3, 4$ are reported in tables 7.1-7.3, where we show the generator matrix through the polynomials $n(D)$ and $d(D)$, the weights w_{min} and h_{min} and the effective free distance $d_{free,eff}$ of the convolutional coupled code. We note that in some cases, there is more than one polynomial $n(D)$ yielding the largest $d_{free,eff}$ for a chosen $d(D)$. Simulation results have shown that the performances of the effective boundary coupled codes (CCC_{ν}^{bd}) do not differ for all candidate codes. We have applied this procedure to select rate $1/3$ and $2/3$ convolutional coupled codes using rate $1/3$ and $2/3$ RSC outer codes, respectively. The results are given in Appendix B. In the next section we will try to choose the best RSC outer code in terms of the convergence properties of the convolutional coupled code.

m	$d(D)$	$n(D)$	w_{min}	h_{min}	$d_{free,eff}$
2	7	5	2	4	18
3	15	17,13,11	2	6	20
4	31	37,35,33, 27,25,23	2	10	24
	23	37,35,33, 31,27,25	2	10	24

Table 7.3: Best rate $1/2$ RSC outer codes with primitive feedback polynomials for coupling factor 7 with respect to the effective free distance.

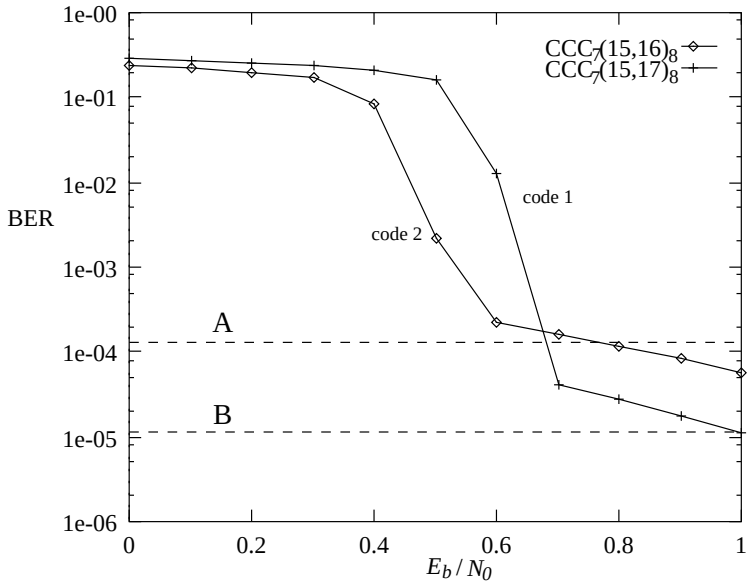

Figure 7.1: BER of rate $1/2$ convolutional coupled codes with two different outer code polynomials.

7.1.2 Code Search Based on the EXIT Chart Technique

When we compare coding schemes with different component codes it seems to be a typical behavior that codes with good convergence properties have inferior performance with respect to the error floor. The performance curve and the typical relation between schemes employing different component codes are illustrated in figure 7.1. The choice between the code 1 with outer code polynomials $(15, 17)_8$ and code 2 with outer code polynomials $(15, 16)_8$ in this example depends on the required BER. As can be seen from the curves, the code 2 outperforms the code 1 for high values of the bit error probabilities. Below 10^{-4}, the code 1 behaves significantly better. This is due to the fact that code 1 holds the larger effective free distance. If we just require a BER of A, code 2 is obviously the best choice. If we require a BER of B, code 1 is the best choice.

The same result can be obtained from the EXIT charts of the two codes (figure 7.2). The inner transfer characteristic is the same for the two codes. Obviously the one with the best performance at the beginning (a priori information $I_A \simeq 0$) is the worst at the end (a priori information $I_A \simeq 1$). The tunnel of the code 2 with outer RSC polynomials $(15, 16)_8$ is much wider open than the one of the code 1 with outer RSC polynomials $(15, 17)_8$, which indicates that convergence of the iterative decoding procedure of code 2 is possible for lower signal to noise ratios.

In the previous section we have chosen the RSC outer codes with respect to the distance properties of the convolutional coupled codes. We note that in same cases more than one generator polynomial yields the best distance properties. Now we try to find under this outer code polynomials the best one, which provides the best convergence properties. Consider the code polynomials of the rate $1/2$ convolutional coupled code with memory $m = 3$ and coupling factor $\mathcal{K} = 7$

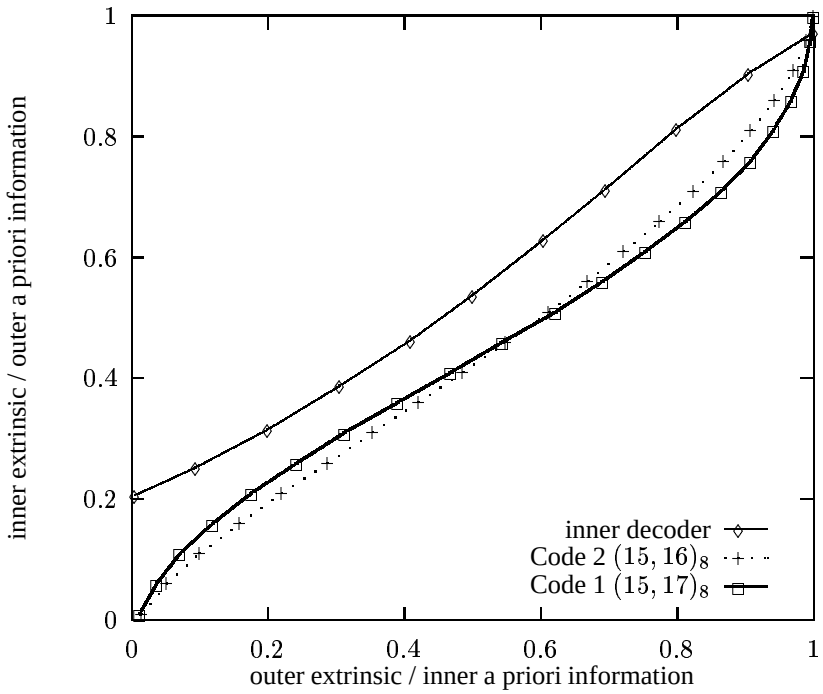

Figure 7.2: EXIT charts of rate $1/2$ convolutional coupled codes with two different RSC outer codes of memory $m = 3$, $\mathcal{K} = 5$ and $E_b/N_0 = 1$dB.

of table 7.3. There are 3 polynomials $n(D)$ yielding the same effective free distance of 20, which are $13_8, 11_8$ and 17_8 for the chosen polynomial $d(D) = 15_8$.

Figure 7.3 shows the EXIT charts of the convolutional coupled codes with the three different outer codes at fixed $E_b/N_0 = 1$dB. We observe that the transfer characteristics of different polynomials differ from each other. If we compare this transfer characteristics to the transfer characteristic of the inner block codes, the outer code with polynomials $(15, 11)_8$ provides the best extrinsic output at the beginning. Moreover, we observe, that the tunnel between the transfer characteristics of inner and outer codes with polynomials $(15, 11)_8$ is much wider open than those of the other polynomial candidates. This indicate that convergence of the iterative decoding with outer polynomials $(15, 11)_8$ is possible for lower SNR. Figure 7.4 confirms this observation and shows that the waterfall of the convolutional coupled code with outer polynomials $(15, 11)_8$ is sooner than the one with $(15, 17)_8$. Thus, the power efficiency of the memory $m = 3$ convolutional coupled code is optimized without introducing additional decoding complexity and without loss of performance at high SNRs. Indeed, figure 7.4 shows that the performance of the two codes in the error floor region is similar. This is due to the fact that the two codes provide the same effective free distance. The same procedure can be applied for any memory m and coupling factor $\mathcal{K}$.

The comparison with the same Turbo code as in section 5.7.1 is illustrated in figure 7.5. The figure shows the performance of the non- and selected convolutional coupled codes with 9 outer codes and code length $\simeq 2000$ with outer code polynomials $(15, 17)_8$ and $(15, 11)_8$, respectively.

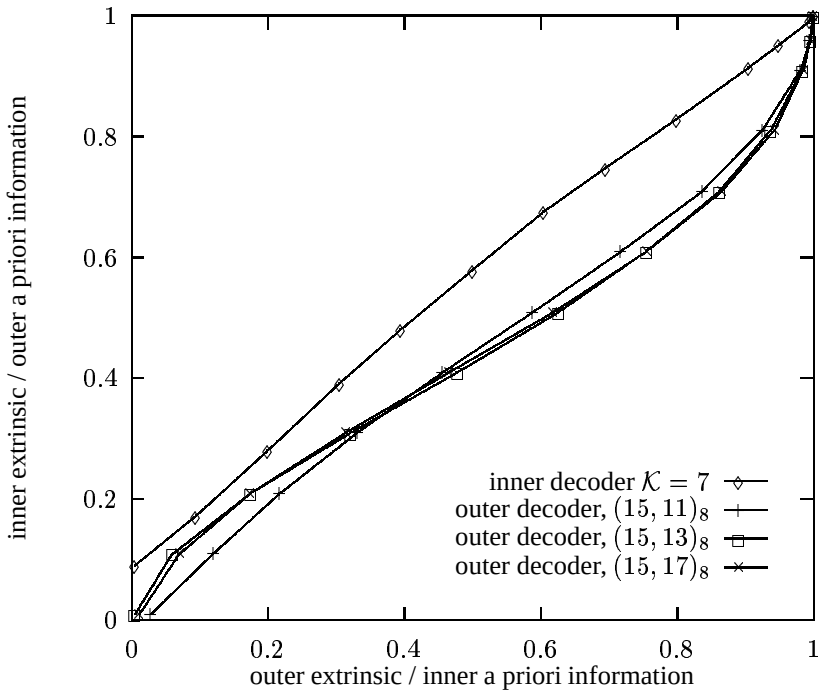

Figure 7.3: EXIT charts of rate $1/2$ convolutional coupled codes for coupling factor $\mathcal{K} = 7$, memory $m = 3$, different outer code polynomials from table 7.3, $E_b/N_0 = 1$dB.

The first one was used in the comparison with Turbo codes in section 5. It can be observed, that the expected gain in the waterfall-region occurs even for this range of code lengths. On the other hand, compared with the Turbo code, the performance of the selected convolutional coupled code with outer code polynomials $(15, 11)_8$ is accurately the same in the waterfall region.

7.2 Conclusion

In chapter 5, it was shown that the asymptotic performance of the convolutional coupled code is strictly related to the effective free distance. The results are then applied in this chapter to find the "best" RSC code for a given memory m and coupling factor $\mathcal{K}$. With respect to this distance we give a table of the best rate $1/2$. Thereby the choice criterion is the maximization of the effective free distance which is equivalent to the minimization of the bit error probabilities at high SNRs. Then we have given some code design examples with component codes chosen according to optimizing the convergence behavior. This code search is based on the EXIT chart technique and yields the best constituent codes with respect to the turbo cliff position.

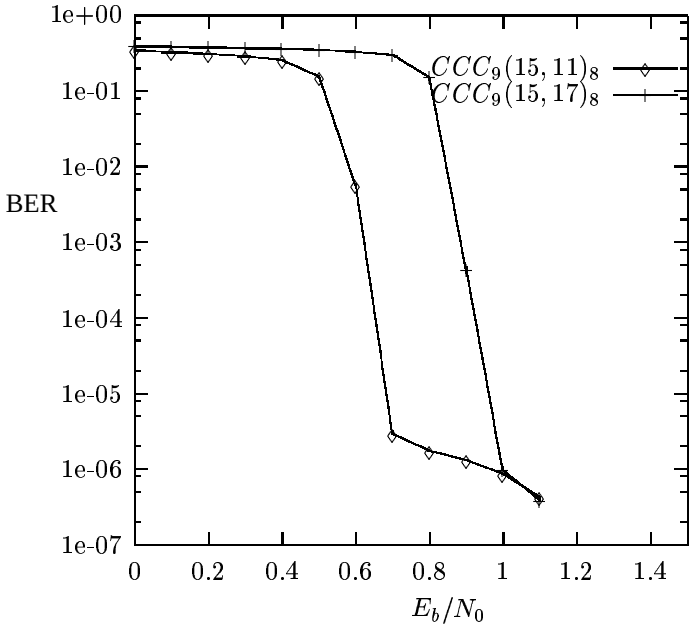

Figure 7.4: Comparison of BER performance of rate $1/2$ convolutional coupled codes for different outer code polynomials ($\mathcal{K} = 7$, code length $N \simeq 112000$).

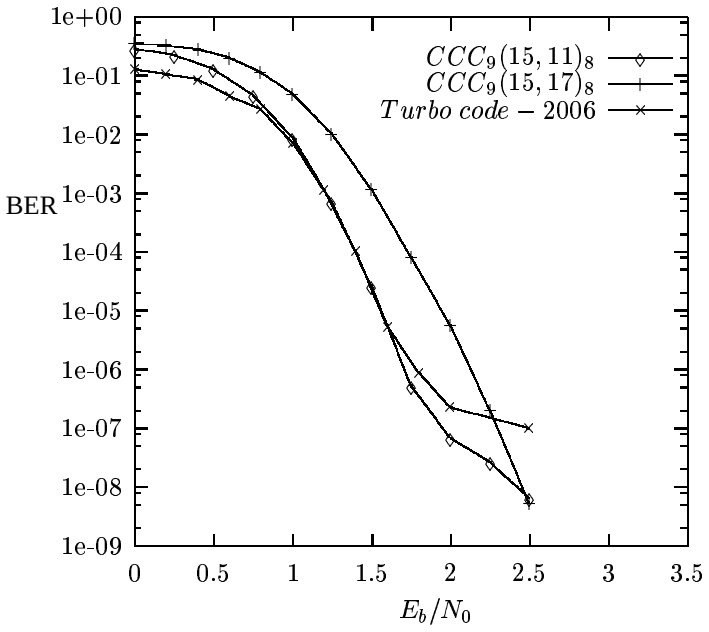

Figure 7.5: Comparison between rate $1/2$ convolutional coupled codes with non- and selected outer RSC codes and Turbo code ($\mathcal{K} = 7$, code length $N \simeq 2000$).

Chapter 8

CCC Extended with Outer BCH Code

At high SNRs the performance of convolutional coupled codes is dominated by a few words with low weight. We refer this part of the curve as the "error floor", although it is not a floor in a true mathematical sense. At low SNRs the main problem is lack of convergence in the iterated decoding process, giving a large number of erroneous bit in the lost frames. This is due to the small free distance of the code. In chapter 5, it was shown that the "error floor" can be estimated by the performance of the sub-code defined as a convolutional coupled code with only one of the ν outer codes being different from zero. This was called "effective boundary coupled code". Thereby it was shown that the free distance of the sub-code and the effective free distance of the convolutional coupled code are similar. In this chapter, an extended coding scheme is proposed including an outer BCH code correcting a few bit errors. The minimum distance of BCH codes is well known, i.e. for a given minimum distance, a corresponding BCH code can be easy developed. If the code coupling scheme is extended with an outer BCH code correcting a few bit errors, the error floor can be lowered significantly, but, this has the disadvantage, that the code rate is reduced and thus, the energy pro symbol is reduced. This is illustrated with an example.

8.1 BER Estimation for High SNRs

Using the union bound, an upper bound to the bit error probability for ML soft decoding of the code over an AWGN channel can be computed in form:

$$P_b \leq \frac{1}{2K} \sum_{d=d_{min}}^{N} B_d^{C_c} erfc(\sqrt{R_c d \frac{E_b}{N_0}}), \tag{8.1}$$

where K is the number of information bits in a code word, N is the number of code bits, R_c is the over-all code rate of the convolutional coupled code and $B_d^{C_c}$ is the sum over all the non-zero information bits of all weight d code words, i.e.

$$B_d^{C_c} = \sum_{w=1}^{K} w A_{w,d-w}^{C_c}, \tag{8.2}$$

thereby $A_{w,h}$ denotes the number of code words generated by an input information of Hamming weight w whose parity-check bits have Hamming weight h, so that the over-all hamming weight is $w + h$. One gets the input-redundancy weight enumerating function (IRWEF) as

$$T^{C_c}(X, Y) = \sum_{w,h} A_{w,h}^{C_c} X^w Y^h. \tag{8.3}$$

The computation of the union bound requires knowledge of the weight distribution of the code. Unfortunately, the presence of the interleavers in coupled code structures makes it impossible to calculate the weight distribution of the code. Thus, we calculate the union bound of the so called effective boundary coupled code CCC^{bd} as an estimation of the bit error probability P_b. The weight distribution $T^{bd}(X, Y)$ of the sub-code CCC^{bd} can be calculated computer based. The bit error rate of the effective boundary coupled code can be bounded for the AWGN channel by

$$\begin{aligned} P_b^{bd} &\leq \frac{1}{2} \sum_{d=d_{min}}^{N} \sum_{w=1}^{K} \frac{w}{K} A_{w,d-w}^{bd} erfc(\sqrt{R_c d \frac{E_b}{N_0}}) \\ &\leq \sum_{w=1}^{K} \frac{1}{2} \sum_{d=d_{min}}^{N} \frac{w}{K} A_{w,d-w}^{bd} erfc(\sqrt{R_c d \frac{E_b}{N_0}}) \\ &\leq \sum_{w=1}^{K} P_{b,w}, \end{aligned} \tag{8.4}$$

where $P_{b,w}$ is the BER, which is caused by the error patterns of weight w. Figure 8.1 shows the contribution of the probabilities $P_{b,2}$ to $P_{b,5}$ to the total BER P_b^{bd}. We note that below 10^{-3} more than 80% of P_b^{bd} are caused by error patterns of weight less than 4. The use of a BCH code, which is able to correct 3 errors, can obviate more than 80% of the bit errors. If the coupling code scheme is extended with an outer BCH code correcting t bit errors, we obtain form (8.4) the "union bound" of the with BCH code extended sub-code:

$$P_b^{BCH} \leq \frac{1}{2} \sum_{d=d_{min}}^{N} \sum_{w=t+1}^{K} \frac{w}{K} A_{w,d-w}^{bd} erfc(\sqrt{R_c d \frac{R_{BCH} E_b}{N_0}}). \tag{8.5}$$

Notice that we of course get a decreased E_s/N_0 due to the rate of the BCH code.

8.2 Simulation

In order to compare the performance of a conventional convolutional coupled code with the extended one, a simulation is accomplished. The BER of the extended convolutional coupled

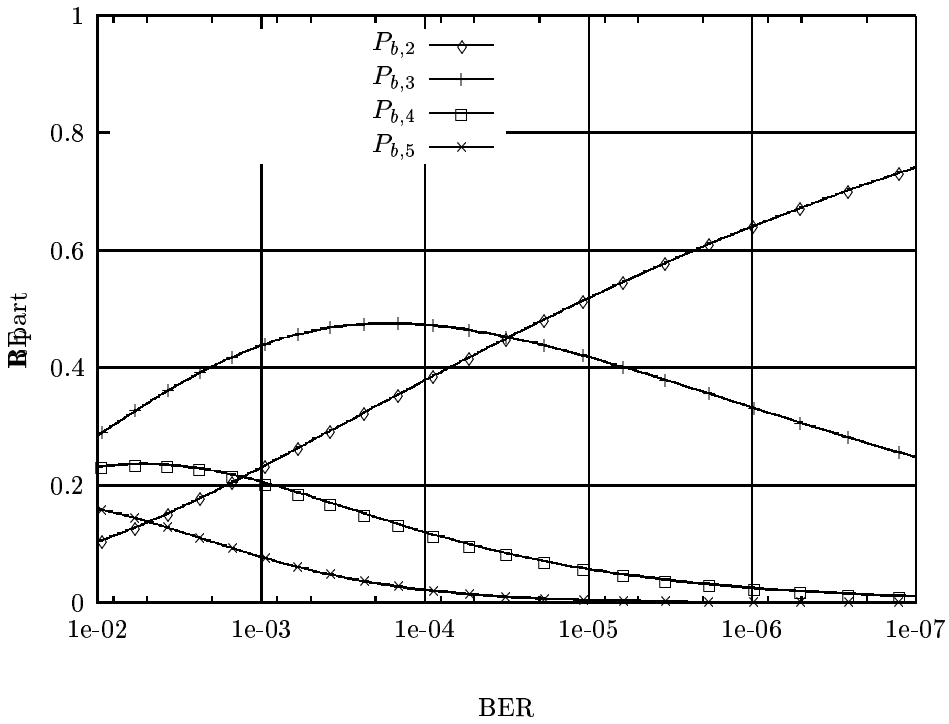

Figure 8.1: Contribution in percent of error events with increasing information weight w to the over-all bit error probability as function of the bit error probability.

code ($g_1 = 15_8, g_2 = 17_8, k = 147, \nu = 7$) with a BCH outer code ($R_{BCH} = 993/1023, t = 3$) was simulated. For comparison we investigate the BER performance of the conventional convolutional coupled code ($g_1 = 15_8, g_2 = 17_8, k = 142, \nu = 7$). The number of information bits per code word amounts $\nu \times k = 1029$, although the BCH code generates code words of length 1023. Thus, 6 bits are not used in each code word.

Figure 8.2 shows the simulated BER performance of the conventional convolutional coupled code and the corresponding with an outer BCH code extended one. Moreover, the "union bounds" from (8.4) and (8.5) are also depicted. We note, that in the "error floor region" the "union bound" of the sub-code is a very good estimation of the BER performance behavior of the conventional coupled code. We note the same observation for the extended convolutional coupled code, indeed the curve the the extended code converges at high SNRs to the "union bound" of the corresponding sub-code. In the "waterfall region" the conventional coupled code outperforms the extended one. This is due to the fact that the code rate of the BCH code is 0.965, which corresponds to a decrease in the symbol energy of 0.155dB (see section 3.2.1), i.e. if the BCH code corrected no errors, the BER of the extended code, compared with the conventional coupled code, would be shifted by 0.155dB to the right. This corresponds approximately the simulated behavior in the "waterfall region". In this region, the BCH code corrects practically no errors, but rather it degrades only the SNR. This is due to the fact that in this region, the bit errors are always caused by no convergence of the iterative decoder. The resulting error patterns have very large weight, and thus they can not be corrected by the BCH code. The increased free distance of the extended

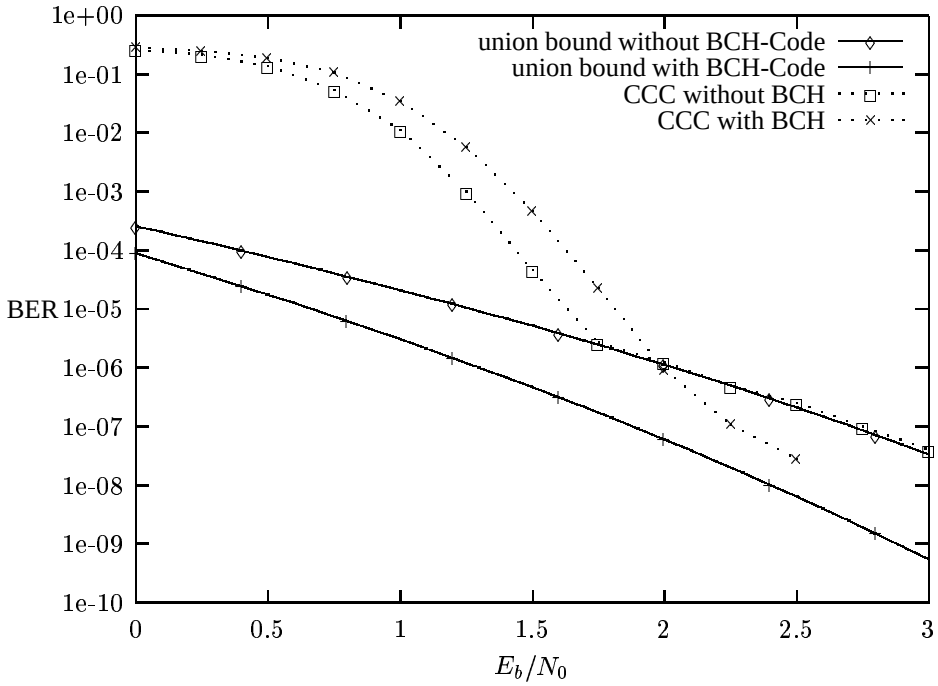

Figure 8.2: Simulation results and bounds.

convolutional coupled code has practically no effect in this region. In the “error floor region” the BER performance of the extended convolutional coupled code is superior to the conventional one and converges to a lower “union bound”. Here, it is ensured that the iterative decoder converges in the majority of cases. Bit errors are caused generally through a small free distance. Most of occurring error patterns mostly have a small weight and can be partly corrected by the BCH decoder.

8.3 Conclusion

An extended coding scheme is proposed including an outer BCH code correcting a few bit errors. In applications where the “error floor” is a problem an effective solution is to concatenate the complete convolutional coupled code with an outer algebraic code, thereby removing the low weight words. The price to be payed for that is slightly worse in the “waterfall region”, for the benefit of an improved performance in the “error floor region”.

Chapter 9

Conclusions

In this thesis the development of convolutional coupled codes has been described. After a short introduction of block and convolutional codes, concatenated codes were examined. This helped us to develop the necessary intuition and understanding of the problems concerning code coupling. Then, the fundamental principle behind Turbo coding has been introduced, including the encoder structure and the principle of iterative decoding. The central aspects of the Turbo encoder concerning constituent encoders, interleaver, termination, puncturing and distance spectra were first examined, and then the iterative decoding scheme was presented. This offers efficient decoding of a complex and powerful code, however at the price of being suboptimal to maximum likelihood decoding. Inspired from Turbo codes with which near Shannon limit error correction performance can be achieved, we have introduced the new code class of convolutional coupled codes. There are two purposes of this thesis. Firstly, it is an introduction of a new class of concatenated codes and, secondly, it is an overview over the major results and contributions to the field of designing convolutional coupled codes. The codes investigated here are constructed via concatenation of a number ν of outer systematic codes of rate k/n and a number k of inner systematic block codes of rate $1/2$. The bits of each information vector of the outer codes are scrambled by a given interleaving before entering to the inner encoders. Differently from the Turbo codes, in which the information symbols and the redundancy from the constituent codes are transmitted, we transmit only the redundancy produced from the inner and outer codes. The code properties were first investigated, whereby the structural properties of the generator matrices for the convolutional coupled codes was presented. A lower- and an upper bound for the minimum distance of the convolutional coupled code was given. Thereafter, we have introduced the term of effective free distance which seems to be one of the most important parameters for construction of convolutional coupled codes. As a major result, we have shown, that the structure of the these codes allows to give a lower bound on the error performance. In addition we have shown that the average bit error probability using the uniform interleaver which permits an easy derivation of the weight enumerating function of the coupled code, is not a significant bound for coupled codes with deterministic interleavers. This confirms the fact that the achieved gain in the decoding performance is attributed to the choice of an appropriate interleaving, which is ascribed to the basic idea of this construction where the use of the interleavers was visualized.

Moreover, this thesis shows that the performance of a convolutional coupled code depends mainly on the coupling factor. Indeed the behavior of the error floor can be constructed and controlled easily by analytical means. Moreover we observed, that the performance does not depend strongly from interleaver design which is for coupled codes simple and straightforward. This is in contrast to Turbo codes, where the performance depends very much on the interleaver structure, which has to be optimized in tedious investigations. Thus, in applications where the "error floor" is a problem an effective solution is to increase the coupling factor. A big coupling factor improves the distance properties, and is therefore important for the performance at high SNRs. On the other hand, at low SNRs a small coupling factor results in a better iterative performance. Thus, the choice of the coupling factor depends on the target operating SNR of the convolutional coupled code.

Maximum likelihood decoding of convolutional coupled codes is unfeasable, therefore we have suggested iterative soft-in/soft-out decoding similar to the decoding of Turbo codes. We have presented simulation results for convolutional coupled codes using this iterative algorithm. Thereby it was shown that iterative decoding works not only for systematic, but also for non-systematic codes. In addition, simulation results have shown, that convolutional coupled codes have the potential of being a realistic alternative to Turbo codes.

One purpose of this thesis is the design of convolutional coupled codes, that is, to give guidelines how to choose and/or design the components in a convolutional coupled encoder to get the best performance. Traditionally, the issue of designing codes is focused on the goal of creating the best possible Hamming distance spectrum and in some cases the best minimum distance. Indeed, this is a major issue also in the case of designing convolutional coupled codes. Since the performance of the coupled codes is strictly related to the defined effective free distance, we have proposed design guidelines to chose the outer convolutional codes. The selection criterion is the maximization of the effective free distance for a given memory and coupling factor. However, since convolutional coupled codes are decoded using an iterative and suboptimal decoding algorithm, new design criteria arise. In fact, the success of iterative decoding proves to be related to the choice of the components in the convolutional coupled codes. The Extrinsic Information Transfer (EXIT) Chart, introduced by Brink, was used as a tool to facilitate the design of convolutional coupled codes. It was shown, that EXIT charts, also for coupled codes , are an accurate and excellent means to predict the iterative decoding behavior of such a concatenated coding scheme. A code search based on the EXIT chart technique was proposed to obtain the best constituent codes which are chosen with respect to the turbo cliff position. Finally, an extended coding scheme was proposed including an outer BCH code correcting a few bit errors.

The investigation in this thesis have been restricted to a specific rate $1/2$ convolutional coupled code structure and BPSK modulation over a memoryless AWGN channel. The purpose of these reasonable and usual restrictions was to allow for more detailed studies of confined issues for code structure and decoding.

On the other hand, it leaves a number of interesting question open for further investigation. For example convolutional coupled code structure for different transmission environments, in the

form of modulation and channel characteristics. Up to now, we have only considered circulant matrices with one or two zeros in each row as coupling matrices. Using other matrices may improve the performance additionally. Here the complexity of the inner SISO decoder has to be discussed.

Appendix A

The BCJR Algorithm

A.1 Theoretical Description

The BCJR algorithm [2] is an optimal algorithm for calculating the a posteriori probabilities of symbols encoded with a convolutional code and transmitted on an AWGN channel. The algorithm is re-derived in this appendix, with notations and modifications appropriate when decoding rate -1/2 recursive systematic convolutional encoders in an iterative decoding environment (Turbo decoding). The review is based on the derivation in [2][11] [44][57].

The BCJR-algorithm is based on calculating the state transition probabilities of the Markov chain representing the encoding process. The encoder input sequence is denoted $\mathbf{u} = (u_1, u_2, \cdots, u_K)$ and the parity sequence generated by a rate-$1/2$ recursive convolutional encoder $\mathbf{c} = (c_1, c_2, \cdots, c_K)$, where $u_i, c_i \in \{0, 1\}$. This notation is illustrated in figure A.1, showing an encoder with generator polynomials $(1, 7/5)_8$. Using binary antipodal modulation, the u_i and c_i are converted to ± 1 before being transmitted on the channel. With the additive white Gaussian noise channel model, the received systematic and parity sequences, $\mathbf{x} = (x_1, x_2, \cdots, x_K)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_K)$, are modeled as

$$x_i = (2u_i - 1) + w_{x,i} \quad \text{(A.1)}$$

$$y_i = (2c_i - 1) + w_{y,i}, \quad \text{(A.2)}$$

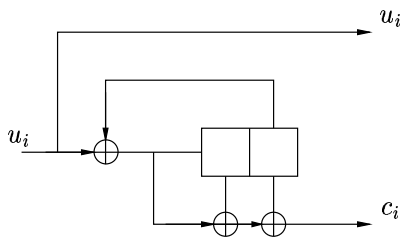

Figure A.1: Example of a recursive convolutional encoder with generator polynomials $(1, 7/5)_8$.

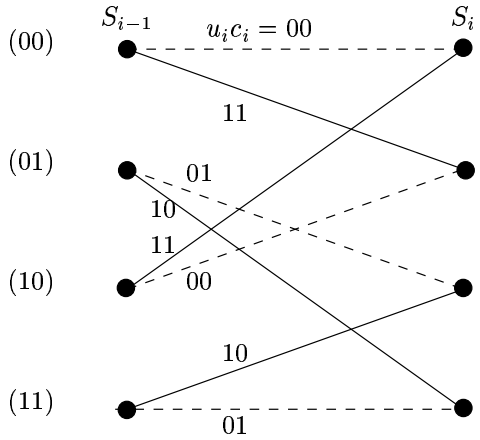

Figure A.2: A 4-state trellis diagram for an RSC encoder with generator polynomials described by $(1, 7/5)_8$. The trellis transitions corresponding to input 0's and 1's are marked with dashed and solid lines, respectively.

where $w_{x,i}$ and $w_{y,i}$ are independent and identically distributed zero mean Gaussian random variables with variance σ^2, i.e.

$$w_{x,i}, w_{y,i} \sim N(0, \sigma^2), \quad i = 1, \cdots, K. \tag{A.3}$$

Apart from the received code sequence **x** and **y**, the decoder has access to a priori information for each transmitted bit, denoted $La(u_i)$. The a priori information is given as the logarithm of the prior probabilities that information bit u_i is 1 and 0 respectively, that is,

$$La(u_i) = ln \frac{P(u_i = 1)}{P(u_i = 0)}. \tag{A.4}$$

The decoder inputs for an entire block of length K is denoted **R**, where $\mathbf{R} = (\mathbf{x}, \mathbf{y}, \mathbf{La})$.

The of the decoder is to calculate the a posteriori probabilities that information symbol u_i is 1 or 0, based on the received code sequences and the available a priori information, that is, calculate $P(u_i = 1|\mathbf{R})$ and $P(u_i = 0|\mathbf{R})$. These probabilities can be calculated by summing state transition probabilities in a trellis diagram. A section of the trellis diagram of the 4-state encoder is shown in figure A.2. Each state transition is labeled with the systematic and the parity bit generated by the encoder. In order to calculate $P(u_i = 1|\mathbf{R})$ and $P(u_i = 0|\mathbf{R})$, the probability of trellis transitions corresponding to $u_i = 0$, marked with solid and dashed lines in figure A.2 is summed. Denoting the encoder state after the ith input symbol by $S_i \in \{0, 1, \cdots, 2^{m-1}\}$, where m is the number of memory elements in the encoder, we get

$$P(u_i = k|\mathbf{R}) = \sum_{s'=0}^{2^m-1} \sum_{s=0}^{2^m-1} P(u_i = k, S_{i-1} = s', S_i = s|\mathbf{R}), \quad k = 0, 1. \tag{A.5}$$

For the sake of calculating the above state transition probabilities, it is convenient to define the joint probability density function

$$\sigma_i^k(s^{'}, s) = P(u_i = k, S_{i-1} = s^{'}, S_i = s, \mathbf{R}), \tag{A.6}$$

through which (A.5) becomes

$$P(u_i = k|\mathbf{R}) = \sum_{s'=0}^{2^m-1} \sum_{s=0}^{2^m-1} \sigma_i^k(s^{'}, s)/P(\mathbf{R}). \tag{A.7}$$

The probability density function (pdf) $\sigma_i^k(s^{'}, s)$ can be calculated in a recursive manner by factorizing it into three parts, denoted $\alpha_{n-1}\tilde{}(s')$, $\gamma_i^k(R_i\tilde{}, s', s)$ and $\beta_i\tilde{}(s)$. Firstly,

$$\begin{aligned}
\sigma_i^k(s^{'}, s) &= P(u_i = k, S_{i-1} = s^{'}, S_i = s, \mathbf{R}_{1,i}) \cdot \\
&\quad P(\mathbf{R}_{i+1,K}|u_i = k, S_{i-1} = s^{'}, S_i = s, \mathbf{R}_{1,i}) \\
&= P(S_{i-1} = s^{'}, \mathbf{R}_{1,i-1}) P(u_i = k, S_i = s, R_i|S_{i-1} = s^{'}, \mathbf{R}_{1,i-1}). \\
&\quad P(\mathbf{R}_{i+1,K}|u_i = k, S_{i-1} = s^{'}, S_i = s, \mathbf{R}_{1,i}).
\end{aligned} \tag{A.8}$$

Here, $P(u_i = k, S_i = s, R_i|S_{i-1} = s^{'}, \mathbf{R}_{1,i-1}) = P(u_i = k, S_i = s, R_i|S_{i-1} = s^{'})$, since the information bit u_i, the state S_i and the decoder input R_i are all independent on $\mathbf{R}_{1,i-1}$ if the encoder state S_{i-1} is known. Similarly, $P(\mathbf{R}_{i+1,K}|u_i = k, S_{i-1} = s^{'}, S_i = s, \mathbf{R}_{1,i}) = P(\mathbf{R}_{i+1,K}|S_i = s)$. Thus,

$$\begin{aligned}
\sigma_i^k(s^{'}, s) &= P(S_{i-1} = s^{'}, \mathbf{R}_{1,i-1}) P(u_i = k, S_i = s, R_i|S_{i-1} = s^{'}) \cdot \\
&\quad P(\mathbf{R}_{i+1,K}|S_i = s) \\
&= \tilde{\alpha}_{i-1}(s^{'}) \tilde{\gamma}_i^k(R_i, s^{'}, s) \tilde{\beta}_i(s),
\end{aligned} \tag{A.9}$$

where

$$\begin{aligned}
\tilde{\alpha}_{i-1}(s^{'}) &= P(S_{i-1} = s^{'}, \mathbf{R}_{1,i-1}), \\
\tilde{\gamma}_i^k(R_i, s^{'}, s) &= P(u_i = k, S_i = s, R_i|S_{i-1} = s^{'}), \quad \textit{and} \\
\tilde{\beta}_i(s) &= P(\mathbf{R}_{i+1,K}|S_i = s).
\end{aligned} \tag{A.10}$$

The recursive part is in the calculation of the $\tilde{\alpha}$s and the $\tilde{\beta}$s since they can be expressed in their preceding counterparts, respectively. More precisely,

$$\begin{aligned}
\tilde{\alpha}_i(s) &= P(S_i = s, \mathbf{R}_{1,i}) \\
&= \sum_{s'=0}^{2^m-1} \sum_{k=0}^{1} P(u_i = k, S_{i-1} = s', S_i = s, \mathbf{R}_{1,i}) \\
&= \sum_{s'=0}^{2^m-1} \sum_{k=0}^{1} P(S_{i-1} = s', \mathbf{R}_{1,i-1}) \cdot \\
&\quad P(u_i = k, S_i = s, R_i | S_{i-1} = s', \mathbf{R}_{1,i-1}) \\
&= \sum_{s'=0}^{2^m-1} \sum_{k=0}^{1} P(S_{i-1} = s', \mathbf{R}_{1,i-1}) \cdot P(u_i = k, S_i = s, R_i | S_{i-1} = s') \\
&= \sum_{s'=0}^{2^m-1} \sum_{k=0}^{1} \tilde{\alpha}_{i-1}(s') \tilde{\gamma}_i^k(R_i, s', s), \qquad \text{(A.11)}
\end{aligned}$$

and

$$\begin{aligned}
\tilde{\beta}_i(s') &= P(\mathbf{R}_{i+1,K} | S_i = s') \\
&= \sum_{s=0}^{2^m-1} \sum_{k=0}^{1} P(u_{i+1} = k, S_{i+1} = s, \mathbf{R}_{i+1,K} | S_i = s') \\
&= \sum_{s=0}^{2^m-1} \sum_{k=0}^{1} P(\mathbf{R}_{i+2,K} | u_{i+1} = k, S_{i+1} = s, R_{i+1}, S_i = s') \cdot \\
&\quad P(u_{i+1} = k, S_{i+1} = s, R_{i+1}, S_i = s') / P(S_i = s') \\
&= \sum_{s=0}^{2^m-1} \sum_{k=0}^{1} P(\mathbf{R}_{i+2,K} | S_{i+1} = s) \cdot \\
&\quad P(u_{i+1} = k, S_{i+1} = s, R_{i+1} | S_i = s') \\
&= \sum_{s=0}^{2^m-1} \sum_{k=0}^{1} \tilde{\beta}_{i+1}(s) \tilde{\gamma}_{i+1}^k(R_{i+1}, s', s). \qquad \text{(A.12)}
\end{aligned}$$

The calculation of the $\tilde{\alpha}$s and the $\tilde{\beta}$s according to (A.11) and (A.12) are referred to as the forward and the backward recursion.

It remains to calculate the density function $\tilde{\gamma}_i^k(R_i, s', s)$. From its definition in (A.10), $\tilde{\gamma}_i^k(R_i, s', s)$ can be expressed

$$\begin{aligned}
\tilde{\gamma}_i^k(R_i, s', s) &= P(u_i = k, S_i = s, R_i | S_{i-1} = s') \\
&= P(R_i | u_i = k, S_i = s, S_{i-1} = s') \cdot \\
&\quad P(S_i = s | u_i = i, S_{i-1} = s') P(u_i = k). \qquad \text{(A.13)}
\end{aligned}$$

Assuming that the encoder inputs are independent, conditioned on a certain trellis transition, and exchanging R_i for $(x_i, y_i, La(u_i))$, we get

$$\begin{aligned}
\tilde{\gamma}_i^k(R_i, s^{'}, s) &= P(x_i, y_i, La(u_i)|u_i = k, S_i = s, S_{i-1} = s^{'}) \cdot \\
&\quad P(S_i = s|u_i = i, S_{i-1} = s^{'})P(u_i = k) \\
&= P(x_i|u_i = k, S_i = s, S_{i-1} = s^{'}) \cdot \\
&\quad P(y_i|u_i = k, S_i = s, S_{i-1} = s^{'}) \cdot \\
&\quad P(La(u_i)|u_i = k, S_i = s, S_{i-1} = s^{'}) \cdot \\
&\quad P(S_i = s|u_i = k, S_{i-1} = s^{'})P(u_i = k) \\
&= P(x_i|u_i = k) \cdot \\
&\quad P(y_i|u_i = k, S_{i-1} = s^{'}) \cdot \\
&\quad P(La(u_i)|u_i = k)P(u_i = k) \cdot \\
&\quad P(S_i = s|u_i = k, S_{i-1} = s^{'}) \\
&= P(x_i|u_i = k) \cdot \\
&\quad P(y_i|u_i = k, S_{i-1} = s^{'}) \cdot \\
&\quad P(u_i = k|La(u_i))P(La(u_i)) \cdot \\
&\quad P(S_i = s|u_i = k, S_{i-1} = s^{'}). \qquad \text{(A.14)}
\end{aligned}$$

The received samples x_i and y_i are Gaussian distributed random variables with mean $(2u_i - 1)$ and $(2c_n - 1)$ respectively, and standard deviation σ, that is,

$$P(x_i|u_i = k) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i - (2k-1))^2}{2\sigma^2}}, \quad and \qquad \text{(A.15)}$$

$$P(y_i|u_i = k, S_{i-1} = s^{'}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - (2c_i - 1))^2}{2\sigma^2}}. \qquad \text{(A.16)}$$

Further, $P(u_i|La(u_i))$ is the a priori probability that information symbol $u_i = k$. Since $La(u_i) = ln\frac{P(u_i=1)}{P(u_i=0)}$, the a priori probabilities are

$$P(u_i = 1|La(u_i)) = \frac{e^{La(u_i)}}{1 + e^{La(u_i)}} \quad and \qquad \text{(A.17)}$$

$$P(u_i = 0|La(u_i)) = \frac{1}{1 + e^{La(u_i)}}. \qquad \text{(A.18)}$$

The probability density function $P(La(u_i))$ in (A.14) is unknown, however when forming the log-likelihood ratio $La(u_i)$ below, $P(La(u_i))$ is a constant factor common to both the numerator

and denominator. Similarly, knowledge of $P(La(u_i))$ is in principle required in the recursive calculation of the $\tilde{\alpha}$s and the $\tilde{\beta}$s. However, since $P(La(u_i))$ is the same for all the transition in a given section of the trellis (i.e. for a given i). it appears as a constant factor for all $\tilde{\alpha}_i(s)$ and $\tilde{\beta}_i(s)$, $s \in 0, 1, \cdots, 2^m - 1$. Finally, the probability $P(S_i = s|u_i = k, S_{i-1} = s^{'})$ in (A.14) is either one or zero, depending on if there exist a trellis transition between state $s^{'}$ and s associated with an input symbol equal to k.

The a posteriori log-likelihood ratio, denoted $L(\hat{u}_i)$ can now be expressed as

$$\begin{aligned}
L(\hat{u}_i) &= ln\frac{P(u_i = 1|\mathbf{R})}{P(u_i = 0|\mathbf{R})} \\
&= ln\frac{\sum_{s'}\sum_s \tilde{\alpha}_{i-1}(s^{'})\tilde{\gamma}_i^1(R_i, s^{'}, s)\tilde{\beta}_i(s)}{\sum_{s'}\sum_s \tilde{\alpha}_{i-1}(s^{'})\tilde{\gamma}_i^0(R_i, s^{'}, s)\tilde{\beta}_i(s)} \\
&= ln\frac{\sum_{s'}\sum_s \tilde{\alpha}_{i-1}(s^{'})P(x_i|u_i = 1)P(y_i|u_i = 1, S_{i-1} = s^{'})\cdot}{\sum_{s'}\sum_s \tilde{\alpha}_{i-1}(s^{'})P(x_i|u_i = 0)P(y_i|u_i = 0, S_{i-1} = s^{'})\cdot} \\
&\quad \frac{P(u_i = 1|La(u_i))P(La(u_i))P(S_i = s|u_i = 1, S_{i-1} = s^{'})\tilde{\beta}_i(s)}{P(u_i = 0|La(u_i))P(La(u_i))P(S_i = s|u_i = 0, S_{i-1} = s^{'})\tilde{\beta}_i(s)} \\
&= ln\frac{P(x_i|u_i = 1)P(u_i = 1|La(u_i))\sum_{s'}\sum_s(\tilde{\alpha}_{i-1}(s^{'}))\cdot}{P(x_i|u_i = 0)P(u_i = 0|La(u_i))\sum_{s'}\sum_s(\tilde{\alpha}_{i-1}(s^{'}))\cdot} \\
&\quad \frac{P(y_i|u_i = 1, S_{i-1} = s^{'})P(S_i = s|u_i = 1, S_{i-1} = s^{'})\tilde{\beta}_i(s)}{P(y_i|u_i = 0, S_{i-1} = s^{'})P(S_i = s|u_i = 0, S_{i-1} = s^{'})\tilde{\beta}_i(s)} \\
&= ln\frac{P(x_i|u_i = 1)}{P(x_i|u_i = 0)} + ln\frac{P(u_i = 1|La(u_i))}{P(u_i = 0|La(u_i))} + \\
&\quad ln\frac{P(u_i = 1|\mathbf{x}_{1,i-1}, \mathbf{x}_{i+1,K}, \mathbf{y}_{1,K}, \mathbf{La}_{1,i-1}, \mathbf{La}_{i+1,K})}{P(u_i = 0|\mathbf{x}_{1,i-1}, \mathbf{x}_{i+1,K}, \mathbf{y}_{1,K}, \mathbf{La}_{1,i-1}, \mathbf{La}_{i+1,K})} \\
&= \frac{2}{\sigma^2}x_i + La(u_i) + Le(\hat{u}_i), \qquad \text{(A.19)}
\end{aligned}$$

where $Le(\hat{u}_i)$ denotes the decoder extrinsic output. The partitioning of the posteriori log-likelihood ratio $L(\hat{u}_i)$ is useful in an iterative decoding environment, where it is only the extrinsic output $Le(\hat{u}_i)$ that is passed on to the next decoding step. Since the first and second term in (A.19) are easily obtained from the encoder inputs, the most straightforward way to calculate the extrinsic output is through

$$\begin{aligned}
Le(\hat{u}_i) &= L(\hat{u}_i) - \frac{2}{\sigma^2}x_i - La(u_i) \\
&= ln\frac{\sum_{s'}\sum_s \tilde{\alpha}_{i-1}(s^{'})\tilde{\gamma}_i^1(R_i, s^{'}, s)\tilde{\beta}_i(s)}{\sum_{s'}\sum_s \tilde{\alpha}_{i-1}(s^{'})\tilde{\gamma}_i^0(R_i, s^{'}, s)\tilde{\beta}_i(s)} - \frac{2}{\sigma^2}x_i - La(u_i), \qquad \text{(A.20)}
\end{aligned}$$

where $\tilde{\alpha}_{i-1}(s^{'})$ and $\tilde{\beta}_i(s)$ are recursively calculated using (A.11) and (A.12).

A.2 Implementation Aspects

When forming the LLR in (A.20), any factors that are common to all the terms in the summations in both the numerator and the denominator may be omitted. Beginning with the $\tilde{\gamma}_i^k(R_i, s^{'}, s)$s,

$$\begin{aligned}\tilde{\gamma}_i^k(R_i, s^{'}, s) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-(2k-1))^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i-(2c_i-1))^2}{2\sigma^2}} \frac{e^{kLa(u_i)}}{1+e^{La(u_i)}} P(La(u_i)) \cdot \\ &\quad P(S_i = s|u_i = k, S_{i-1} = s^{'}) \\ &= e^{-\frac{x_i^2+(2k-1)^2+y_i^2+(2c_i-1)^2}{2\sigma^2}} \frac{P(La(u_i))}{2\pi(1+e^{La(u_i)})} \cdot \\ &\quad e^{\frac{x_i(2k-1)+y_i(2c_{i-1})}{\sigma^2}+kLa(u_i)} P(S_i = s|u_i = k, S_{i-1} = s^{'}). \end{aligned} \tag{A.21}$$

Using $(2k-1)^2 = 1$ and $(2c_i - 1)^2 = 1$, we define

$$\begin{aligned}\mathrm{C}_{\gamma_i} &= \frac{P(La(u_i))}{2\pi\sigma^2(1+e^{La(u_i)})} e^{-\frac{x^2+y^2+2}{2\sigma^2}}, \quad and \\ \gamma_i^k(R_i, s^{'}, s) &= e^{\frac{x_i(2k-1)+y_i(2c_{i-1})}{\sigma^2}+kLa(u_i)} P(S_i = s|u_i = k, S_{i-1} = s^{'}), \end{aligned} \tag{A.22}$$

which leads to

$$\tilde{\gamma}_i^k(R_i, s^{'}, s) = C_{\gamma_i}\gamma_i^k(R_i, s^{'}, s). \tag{A.23}$$

Since C_{γ_i} is independent on both s and $s^{'}$, $\tilde{\gamma}_i^k(R_i, s^{'}, s)$ may be replaced by $\gamma_i^k(R_i, s^{'}, s)$ in (A.20). Further, the forward recursion can be written

$$\begin{aligned}\tilde{\alpha}_i(s) &= \sum_{s^{'}=0}^{2^m-1} \sum_{k=0}^{1} \tilde{\alpha}_{i-1}(s^{'}) \tilde{\gamma}_i^k(R_i, s^{'}, s) \\ &= C_{\gamma_i} \sum_{s^{'}=0}^{2^m-1} \sum_{k=0}^{1} \tilde{\alpha}_{i-1}(s^{'}) \gamma_i^k(R_i, s^{'}, s), \end{aligned} \tag{A.24}$$

and since C_{γ_i} is common to all the $\tilde{\alpha}_i(s)$, $s \in (0, 1, \cdots, 2^m - 1)$, it may be omitted from the recursion. The backward recursion is analogous. Thus, we define

$$\begin{aligned}\alpha_i(s) &= \sum_{s^{'}=0}^{2^m-1} \sum_{k=0}^{1} \alpha_{i-1}(s^{'}) \gamma_i^k(R_i, s^{'}, s) \quad and \\ \beta_i(s^{'}) &= \sum_{s^{'}}^{2^m-1} \sum_{k=0}^{1} \beta_{i+1}(s) \gamma_{i+1}^k(R_i, s^{'}, s). \end{aligned} \tag{A.25}$$

The initialization of the first set of α-values, i.e. $\alpha_0(s), s \in (0, 1, \cdots, 2^m - 1)$, is straightforward, it is simply the probabilities of the encoder being in each of its possible states. Since the encoder is initialized in its zero-state, we have

$$\alpha_0(s) = \begin{cases} 1 & s = 0, \\ 0 & otherwise. \end{cases} \tag{A.26}$$

Similarly, for encoders terminated in the zero-state the β-initialization is

$$\beta_K(s) = \begin{cases} 1 & s = 0, \\ 0 & otherwise. \end{cases} \tag{A.27}$$

If the encoder is not terminated in the zero-state, all ending states are equally probable and thus

$$\beta_K(s) = \frac{1}{2^m} \quad s = 0, 1, \cdots, 2^m - 1. \tag{A.28}$$

In each step in the recursive calculation of the αs and the βs they tend to get smaller and smaller. It is therefore a risk for numerical underflow , especially when decoding long blocks. This problem can be avoided by scaling. For example, the αs and the $beta$s for a specific time i can be scaled so that their sums are 1, that is, $\sum_s \alpha_i(s) = 1$ and $\sum_s \beta_i(s) = 1$.

Appendix B

Tables with good RSC codes

Tables B.2 and B.2 show the RSC codes of rate $2/3$ and $1/3$, which generate the maximum effective free distance of the convolutional coupled codes. Thereby, we assume that only one from the ν outer convolutional codes is different from zero. In the tables we show the coupling factor $\mathcal{K}$, the generator matrix through the polynomials $n(D)$ and $d(D)$, the weights w_{min} and h_{min} and the effective free distance $d_{free,eff}$ of the convolutional coupled code.

The rate $2/3$ RSC code is obtained from the puncturing of the rate $1/2$ RSC "mother" code. The puncturing matrix is given by

$$P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

$\mathcal{K}$	m	$d(D)$	$n(D)$	w_{min}	h_{min}	$d_{free,eff}$
3	2	7	5	2	2	8
	3	13	15	2	3	9
		15	13	2	3	9
	4	23	24,25,30,31	2	4	10
		31	24,25,30	2	4	10
5	2	7	5	2	2	12
	3	13	15	2	3	13
		15	13	2	3	13
	4	23	24,25,30,31	2	4	14
		31	24,25,30	2	4	14
7	2	7	5	2	2	16
	3	13	15	2	3	17
		15	13	2	3	17
	4	23	24,25,30,31	2	4	18
		31	24,25,30	2	4	18

Table B.1: Best rate $2/3$ RSC outer codes for coupling factor 3,5 and 7.

$\mathcal{K}$	m	$d(D)$	$n_1(D)$	$n_2(D)$	w_{min}	h_{min}	$d_{free,eff}$
3	2	7	4	5	2	6	12
		7	5	6	2	6	12
	3	13	15	17	3	7	16
		13	16	17	2	10	16
		15	13	17	3	7	16
		15	16	17	2	10	16
	4	21,31	27,33,35	37	3	9	18
5	2	7	4	5	2	6	16
		7	5	6	2	6	16
	3	13	15	17	2	12	22
		15	13	17	2	12	22
	4	21,31	27,33,35,36	37	3	9	24
7	2	7	4	5	2	6	20
		7	5	6	2	6	20
	3	13	11	15,17	2	12	26
		13	15	17	2	12	26
		15	11	13,17	2	12	26
		15	13	17	2	12	26
	4	21,31	27,33,35,36	37	3	9	30

Table B.2: Best rate $1/3$ RSC outer codes for coupling factor 3,5 and 7.

Appendix C

List of Symbols and Abbreviations

Symbols

C Channel Capacity

c Encoded sequence

CCC^{bd} effective boundary coupled code

d_{free} Free distance of a convolutional code

$d_{free,eff}$ Effective free distance

$d_H(\mathbf{c}_1, \mathbf{c}_2)$ Hamming distance between two vectors $\mathbf{c}_1$ and $\mathbf{c}_2$

d_{min} Minimum distance

$d_r(j)$ Extended row distance

$d_c(j)$ Extended column distance

$d_j^{r,m}$ Modified active row distance

E_b Average bit energy

E_s Average symbol energy

$E\{X\}$ Expectation of a variable X

f Frequency

$\mathbf{G}$ Generator matrix

$\mathbf{G}_{sys}$ Equivalent systematic generator matrix

$\mathbf{H}$ Parity check matrix

$\mathbf{I}_k$ $k \times k$ identity matrix

$I(X;Y)$ Average mutual information provided by the output Y about the input X

K Number of information bits

$\mathcal{K}$ Coupling factor

$L(.)$ Log. likelihood ratio

$L(\hat{u})$ a posteriori LLR value

$La(u)$ a priori LLR value

$Le(\hat{u})$ Extrinsic LLR value

$Li(x)$ Intrinsic LLR value

m Number of memory elements of a convolutional code

N Code word length

N_0 Spectral density

n_G Gaussian noise

$P(.)$ Probability

$P(.|.)$ Conditional Probability

P Coupling matrix

R Code rate

R_{ter} Code rate terminated code

R_{gain} Code rate gain

u Information sequence

$\hat{u}$ Estimated sequence

y Received sequence

$w(\mathbf{c})$ Hamming weight of a code word **c**

ν Number of outer codes

π Interleaver

π^{-1} Inverse interleaver

σ^2 Variance of the Gaussian random variable

Abbreviations

A/D Analog-to-Digital

APP A Posteriori Probability

AWGN Additive White Gaussian Noise

BCH Bose Chaudhuri Hocquenghem

BCJR Bahl Cocke Jelinek Raviv

BER Bit Error Rate

BPSK Binary Phase Shift Keying

BSC Binary Symmetric Channel

CC Convolutional Code

CCC Convolutional Coupled Code

CE Cross Entropy

D/A Digital-to-Analog

DMC Discrete Memoryless Channel

EXIT Extrinsic Information Transfer Chart

FF Feed Forward

GF Galois Field

IRWEF Input Redundancy Weight Enumerating Function

LLR Logarithmic Likelihood ratio

MAP Maximum A Posteriori

ML Maximum Likelihood

NSC Non Systematic Convolutional

PCC Parallel Concatenated Codes

PSTC Partial Systematic Turbo Code

RSC Recursive Systematic Convolutional

SISO Soft Input Soft Output

SOVA Soft Output Viterbi Algorithm

SPC Single Parity Check

SNR Signal to Noise Ratio

UI Uniform Interleaver

Bibliography

[1] J. B. Anderson, S. M. Hladik: Tailbiting MAP decoders, IEEE Journal on selected Areas in Communications, 16(2):297-302, Feb. 1998

[2] L. R. Bahl, J. Cocke, F. Jelinek and J- Raviv. Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Transaction on Information Theory*, page 284-286, March 1974

[3] A. S. Barbulescu, S. S. Pietrobon: Termination the trellis of turbo codes in the same state, *Electronic letters*, 31(1):22-23, Jan, January 1995

[4] G. Battail: On random-like codes: *Canadian Workshop on Information Theory*, 1995

[5] G. Battail, M. C. Decouvelaere, P. Goldlewski: Replication decoding, *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 332-345, May 1979

[6] G. Battail: Ponderation des symboles decodes par l'algorithme de Viterbi, *Annale Telecomunic*, 42(1-2):31-38, January 1987

[7] S. Benedetto, G. Montorsi: Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding, *IEEE Trans. on Inf. Theory*,Vol. 44, No. 3, March 1998

[8] S. Benedetto, G. Montorsi: Design of Parallel Concatenated Convolutional Codes, *IEEE Transaction on Communication*, Vol. 44, No. 5, May 1996

[9] S. Benedetto, G. Montorsi: Average performance of parallel concatenated block codes, *IEEE Electronic Letters*, 31(3)_156-158, Feb. 1995

[10] S. Benedetto, G. Montorsi: Unveiling turbo codes: Some result on parallel concatenated coding scheme, *IEEE Transaction on Information Theory*, 42(2):409-428, March 1996

[11] C. Berrou, A. Glavieux and P. Thitimajshima: Near Shannon limit error-correcting coding and decoding: "Turbo Codes", *IEEE Int. Conf. on Communication* (Geneva, Switzerland, 1993), pp. 1064-1070

[12] W. J. Blackert, E. K. Hall, S. G. Wilson: Turbo codes termination and interleaver conditions, *Electronic letters*, 31(24):2082-2084, Nov. 1995

[13] M. Bossert: Kanalcodierung. Teubner Verlag, 1998

[14] S. ten. Brink: Iterative decoding trajectoires of parallel concatenated codes, *In IEEE/ITG Conference on Source and Channel Coding*. Munich, Germany, January 2000

[15] S. Chaoui, U. Sorger: Coupling of Counvolutional Codes, *IEEE Workshop On Concatenated Codes*, Ulm, Germany, Okt. 1999

[16] S. Chaoui: Design of convolutional code coupling, 6*-th International Symposium on Communication Theory and Applications*, Ambelside, UK, July 2001

[17] S. Chaoui, I. Land: Comparison of Convolutional Coupled Codes and Partially Systematic Turbo Codes for Medium Code Lengths, *4th International ITG Conference on Source and Channel Coding*, Berlin, Germany, January 2002

[18] S. Chaoui, U. Sorger: Code Coupling, *Symposium on Turbo codes and related topics*, Brest, France, Sep. 2000

[19] S. Chaoui: Convolutional Coupled Codes between distance properties and convergence behavior of the iterative decoding scheme, *IEEE International Symposium on Information Theory*, Lausanne, Switzerland, 2000

[20] S. Chaoui: Some results on Convolutional Coupled Codes: Distance Properties, Performance, and Design, *Accepted to the International Journal of Electronics and Communications (AEÜ)*

[21] D. J. Costello, Jr: A construction technique for random error correcting convolutional codes, *IEEE Trans. on Inform. Theory*, IT-19:631-636. Sep. 1969

[22] T. M. Cover and J. A. Thomas: Elements of Information Theory, *John Wiley & sons, Inc. New York,* 1991

[23] R. H. Deng, D. J. Costello: High rate concatenated coding systems using bandwidth efficient trellis inner codes, *IEEE Trans. Commun*. vol. 37, pp. 420-427, May 1989

[24] U. Dettmar, U. Sorger: Bounded minimum distance decoding for unit memory codes. *IEEE Trans. Inform. Theory*, pp. 591-596, März 1995

[25] U. Dettmar, U. Sorger, V. Zyablov: Concatenated codes with convolutional inner and outer codes, *Workshop on Information Protection, Moscow, Russia,* Dezember 1993

[26] U. Dettmar: Partial Unit Memory Codes, *Dissertation, Institut für Netzwerk und Signaltheorie, Technische Universität Darmstadt*, 1994

[27] P. Elias: Coding for Noisy Channels, *IRE Conv. Rec*. Part 4, pp. 37-47, 1955

[28] G. D. Forney, Jr: Concatenated Codes, *MIT Press, Cambridge, Mass.*, 1966

[29] G. D. Forney: The viterbi algorithm, *Proc. IEEE*, (3):268-278, March 1973

[30] G. D. Forney: Coset codes II, *IEEE Trans. Inform. Theory*, IT-34:1152-1187, Sept. 1988

[31] M. Fossorier, F. Burkert, S. Lin, J. Hagenauer: On the equivalence between SOVA and Max-Log-MAP decoding, IEEE communication Letters 2(5):137-139, May 1998

[32] C. Fragouli, R. D. Wesel: Semi-Random Inteleaver Design Criteria, *Electrical Engineering Department, University of California, Los Angeles, Global Telecommunication Conference* - Globecom'99

[33] V. Franz, J. Anderson: Reduced-search BCJR algorithm, *In IEEE International Symposium on Information Theory*, page 230, 1997

[34] B. Friedrichs: Kanalcodierung, *Sringer-Verlag, Berlin*, 1996

[35] R. G. Gallager: Low-density Parity-check codes, *Cambridge MA: MIT Press*, 1963

[36] J. Hagenauer, P. Höher: Concatenated Viterbi decoding, *In Proc. 4th Joint swedish-sowiet Int. Workshop on Infotmation Theory (Gotland, Sweden, Studenlitteratur, Lund, Aug. 1989)*, pp.29-33

[37] J. Hagenauer, P. Hoeher: A Viterbi algorithm with soft-decision outputs and its applications, *In Globecom 1989*, pages 1680-1686. Dallas, Texas, USA, November 1989

[38] J. Hagenauer, L. Papke. Decoding "Turbo Codes" with the soft output viterbi algorithm (SOVA), *In IEEE international symposium on Information Theory*, page 164, June 1994

[39] J. Hagenauer, E. Offer, and L. Papke: Iterative decoding of binary block and convolutional codes, *IEEE Trans. Inform. Theory*, Vol 42. pp. 429-445. Mar. 1996

[40] R. W. Hamming, Coding and Information Theory, *Prentice-Hall*, New Jersey, 1986

[41] H. Herzberg: Multilevel turbo coding with short interleavers, *IEEE Journal on selected Areas in Communications*, pages 303-309, Feb. 1998

[42] R. Johannesson, K. Sh. Zigangirov: Fundamentals of convolutional coding, *IEEE Press. Piscataway*, N.J., 1999

[43] R. Johannesson, Z.-x. Wan: A linear algebra approach to minimal convolutional encoders, *IEEE Trans. on Inform. Theory*, IT-39:1219-1233, July 1993

[44] P. Jung, M. M. Naßhan: Comprehensive comparison of Turbo code decoders, *In IEEE Vehicular Technology Conference*, pages 624-628,Chicago, USA, July 1995

[45] A. Kolenberg, G. D. Forney, Jr.: Convolutional coding for channels with Memory, *IEEE Trans. Inf. Theory*, IT-14, pp. 618-626, Sep. 1968

[46] K. Kroschel: Statistische Nachrichtentheorie, *Erster Teil, 2. Auflage. Springer,* 1986

[47] I. Land, P. Hoeher: Partially Systematic Rate $1/2$ Turbo Codes, *2nd Int. Symposium on Turbo Codes & related Topics, Brest*, 4-7 Sep. 2000

[48] S. Lin D. J. Costello, Jr: Error Control Coding: Fundamentals and Applications, *Prentice Hall, Engleword Cliffs,* N. J., 1993

[49] L. Lin, R. Cheng: Improvements in SOVA-based decoding for Turbo codes, *In proceedings of International Conference on Communications, ICC 1997*, pages 1473-1478, Montreal, Canada, June 1997

[50] R. Lucas, M. Bossert: On Iterative Soft-Decision Decoding of linear Binary Block Codes and Product Codes, *IEEE Journal on Selected Areas in Communications*, Vol. 16 No. 2, 1998

[51] D. J. Muder: Minimal Trellises for block codes. *IEEE Trans. Inform. Theory*, IT-34(5):1049-1053, Sep. 1988

[52] L. C. Perez, J. Seghers, D. J. Costello, Jr: A Distance Spectrum Interpretation of Turbo Codes, *IEEE Transaction on Information Theory*, 42(6), 1698-1709. 1996

[53] R. M. Pyndiah: Near-optimum decoding of product codes: Block turbo codes, *IEEE Trans. on Communications*, 46(8):1003-1010, Aug. 1998

[54] J. L. Ramsey: Realisation of optimum interleavers, *IEEE Trans. on Information Theory*, 16(3), 338-345, 1970

[55] P. Robertson: Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes, *In Globecom Conference*, pp. 1298-1303, 1994

[56] P. Robertson, E. Villeburn, P. Hoeher: A Comparison of optimal and sub-optimal MAP decoding algorithm operating in the log domain, *In IEEE International Conference on Communications*, pages 1009-1013, Seattle, USA, 1995

[57] P. Robertson: Improving decoder and code structure of parallel concatenated recursive systematic (Turbo) codes, *In IEEE International Conference on Universal Personal Communications*, pages 183-187, San Diego, USA 1994

[58] C. E. Shannon: A mathematical theory of communication, *Bell Syst. Tech. Journal*, 27, pp. 379-423, July 1948

[59] C. E. Shannon: A mathematical theory of communication, *Bell Syst. Tech. Journal*, 27, pp. 623-656, Okt. 1948

[60] Rose Y. Shao, Shu Lin, and Marc P. C. Fossorier: Two Simple Stopping Criteria for Turbo Decoding, *IEEE Trans. on Communications*, Vol. 47, No. 8, August 1999

[61] Soonyoung Kim, Jinsu Chang, and Moon Ho Lee: A low delay stop criterion of Turbo decoding, 2*nd international Symposium on Turbo Codes & related Topics, Brest, France,* 2000

[62] C. Thommensen, J. Justesen: Bounds on distances and error exponents of unit memory codes, *IEEE Trans. Inform. Theory*, pp. 637-649, September 1993

[63] A. J. Viterbi: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, *IEEE Transaction on Information Theory,* pages 260-269, April 1967

[64] J. K. Wolf: Efficient maximum likelihood decoding of linear coding using a trellis, *IEEE Trans. Inform. Theory,* vol. IT-24, pp.76-80, Jan. 1978

[65] 3rd Generation Partnership Project, Technical Specification Group Radio Access Network, *Multiplexing and channel coding,* 3GPP TS 25.212, V3.4.0, Sep. 2000