On the Design of Turbo Codes

Hokfelt, Johan

2000

On the Design of Turbo Codes

Johan Hokfelt

Lund 2000

Department of Applied Electronics
Lund University
Box 118, SE-221 00 LUND
Sweden

No. 17
ISSN 1402-8662
ISBN 91-7874-061-4

Printed in Sweden by KFS AB, Lund
August 2000

Abstract

This thesis is about Turbo codes – codes constructed via parallel concatenation of two recursive convolutional encoders linked by an interleaver. The focus of the work is on the understanding and design of Turbo codes. This includes thorough investigation of central components that influence Turbo code performances, such as the constituent encoders and the interleaver, 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 Turbo codes are considered: (1) code properties, in terms of Hamming distance spectra, and (2) decoding properties, in terms of the performance of iterative decoding. It is asserted that both these aspects are influenced by both the choice of interleaver and the choice of constituent encoders. An interleaver design algorithm based on these observations is presented. Furthermore, guidelines for the choice of constituent encoders are outlined. As regards the interleaver, it can be designed to result in both good code- and decoding properties. In contrast, the choice of constituent encoders involves a trade off between the two.

A measure that comprises the interleaver properties influencing the performance of iterative decoding is presented. This measure is called *iterative decoding suitability* (IDS), and it is derived using a model that approximates correlation properties of decoder inputs and outputs.

The aspect of trellis termination of Turbo codes is also investigated. It is demonstrated that with proper interleaver design, very competitive error rate performances are obtained also without trellis termination. In addition, it is demonstrated that the 'error-floor' Turbo codes are claimed to suffer from at medium- to high signal-to-noise ratios can be significantly lowered by proper combination of constituent encoders and interleaver design.

Preface

In 1995, Prof. Torleiv Maseng engaged two new Ph.D. students at the Department of Applied Electronics, in the area of modulation and coding. These students were to work on two different projects: one on 'Orthogonal Frequency Division Multiplex', and one on 'Turbo codes'.

Thanks to being a fortunate friend of Fredrik Tufvesson, I applied for one of the positions as Torleiv Maseng's student. Both Fredrik and I were accepted to the programme and now the question were: "Which one of us were to work with which project?"

From the time as an undergraduate student, I had always been intrigued by channel coding. After some brief discussions between Fredrik and I, we had agreed on which projects to choose. Of limited surprise, I was to work with Turbo codes.

With Turbo codes being a recent invention (1993), I was frustrated by the fact that I was to investigate and apply these codes to various communication systems without fundamental understanding of the properties that govern their behavior and performances. As a consequence, my work was directed towards exactly these issues. In this thesis, I have tried to gather what I have learned about the fundamental behavior of Turbo codes during my time as a Ph.D. student.

Acknowledgments

During the years as a student at the department of Applied Electronics, I have had plenty of time and opportunities to become indebted to many people. Even so, perhaps the greatest gratitude of all is of a more abstract nature: atmosphere. Being able to go to work with an easy mind, knowing that positive people and encouraging minds await you, has been invaluable to me. For this, I am indebted to everybody at the department.

There are of course some that I have been working closer with than others. First of all, I am very grateful to my supervisor Prof. Torleiv Maseng. It has been truly reassuring to know that you are always willing to share your time with me. Your quest for new problems and new solutions forms a very challenging

and rewarding environment. I am also immensely grateful to Dr. Ove Edfors, my co-supervisor and friend. You are an indescribable source of energy – thank you for sharing it with me and the other members of the Radio Communications Group. You are, as you know, invaluable. All in all, I am indebted to all the members, as well as former members, of the Radio Communications Group. It has been a true pleasure and privilege to work with you. Fredrik Tufvesson – were it not for you, I would not be in this group in the first place. Carl Fredrik Leanderson. Among all things, thank you for all the discussions we've had – both at work and elsewhere. May they have just begun! Peter Malm. Thank you for being curios, persistent and persuasive. Finally, thank you Bengt-Arne Molin, Henrik Börjeson, Fredrik Florén and André Stranne. I would also like to thank all the members of the Signal Processing Group, with whom we share office space, office hours and coffee breaks.

Further, I am grateful to:

The personnel at the department that helped me with all kinds of practical problems. Especially, Lars Hedenstjerna, Birgitta Holmgren, Erik Jonsson and Britta Olsson.

Prof. Paul H. Siegel and Mats Öberg at the Center for Wireless Communications, University of California in San Diego, for organizing a stay for me at the Center for Wireless Communications in 1997.

My sister and my parents – I love you.

Per and Ulf Scharfenort, for being the best friends I could ever ask for. Thank you for constantly chasing all signs of boredom out of my life.

Lena Brandsten, for turning the last few months into the easiest time of my life, although predicted the opposite.

Finally, I thank NUTEK (The Swedish National Board for Industrial and Technical Development) for supporting this project financially.

Lund, August 2000
Johan Hokfelt

Contents

Thesis Outline

This thesis is about the understanding and the design of Turbo codes. It consists of two parts; the first part provides an introduction to the research field, as well as a review of my research results. The second part consists of papers previously published elsewhere, or submitted for publication.

In Part I, Chapter 1 provides a brief introduction to communication systems in general and to the research field of this thesis, *i.e. channel coding*. A few fundamental concepts of channel coding and the goals that the coding community are striving towards are introduced. In particular, bounds on achievable coding performances are reviewed, and compared to the performance of an example Turbo code.

Chapter 2 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. Further, the mechanism inherent in the structure of Turbo codes that lies behind the lower part of the distance spectra of Turbo codes is discussed.

Chapters 3 and 4 provide an overview of research results regarding the design of the central components of a Turbo code. The work behind this thesis has been primarily focused on the role of the interleaver, but also on trellis termination for Turbo codes and the choice of constituent encoders. The results obtained regarding the interleaver are reviewed in Chapter 3, where two conceptually different interleaver design criteria are discussed. Based on these criteria, an interleaver design algorithm suitable for both small and large interleavers is presented. To our knowledge, the presented algorithm produces interleavers with unsurpassed performances. Chapter 4 discusses the choice of the constituent encoders. As in the case of interleaver design, the choice of constituent encoders is considered both from a distance spectrum- and an iterative decoding point of view.

Finally, in Chapter 5 the research findings are summarized, including a review of the major contributions of this thesis to the research field.

Part II consists of the following papers:

I *Methodical Interleaver Design for Turbo Codes.* Johan Hokfelt and Torleiv Maseng, International Symposium on Turbo Codes & Related Topics, pp. 212–215 Brest, France, September 1997.

II *On the Convergence Rate of Iterative Decoding.* Johan Hokfelt and Torleiv Maseng, IEEE Communications Theory Mini-Conference, held in conjunction with Globecom, pp. 149–154. Sydney, Australia, November 1998.

III *Turbo Codes: Correlated Extrinsic Information and its Impact on Iterative Decoding Performance.* Johan Hokfelt, Ove Edfors and Torleiv Maseng, IEEE Vehicular Technology Conference, pp. 1871–1875. Houston, Texas, May, 1999.

IV *A Turbo Code Interleaver Design Criterion Based on the Performance of Iterative Decoding.* Johan Hokfelt, Ove Edfors and Torleiv Maseng. Accepted for publication in IEEE Communications Letters.

V *Interleaver Structures for Turbo Codes with Reduced Storage Memory Requirement.* Johan Hokfelt, Ove Edfors and Torleiv Maseng, IEEE Vehicular Technology Conference, pp. 1585–1589. Amsterdam, Holland, September 1999.

VI *On the Theory and Performance of Trellis Termination Methods for Turbo Codes.* Johan Hokfelt, Ove Edfors and Torleiv Maseng, submitted to IEEE Journal on Selected Areas in Communications; The Turbo Principle: From Theory to Practice.

Apart from the papers included in this thesis, the following papers on related topics have been published: [45, 46, 48, 49, 51, 52, 64, 65].

Finally, the work on Turbo codes has resulted in three submissions to the *European Telecommunications Standards Institute* (ETSI), regarding the use of Turbo codes within the *Universal Mobile Telecommunications Systems* (UMTS) standardization [39, 40, 41].

Part I

Chapter 1

General Introduction

There are several different parts or functional components of a modern communication system that are loosely referred to as 'coding', for example so diverse things as image compression and cryptography. Turbo codes belong to the group of codes referred to as *channel codes*. In this introductory chapter, the general concepts involved in a communication system are introduced, and the role of channel coding is described. Further, fundamental channel coding concepts are introduced, together with some theoretical limits useful when evaluating code performances.

1.1 A Model of Communication Systems

A complete communication system embraces 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 and lower level designers. 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 information *source* generates messages that are to be transmitted to the receiver. These messages can be either analog or digital, depending on the type of source. For example, the voice transmitted during a phone conversation is represented by the continuous time and continuous amplitude signal generated by a microphone, which together with the speaker constitutes an analog source. In a digital communication system, the messages delivered from an analog source are converted into a digital signal, usually represented by sequences of binary digits, or *bits*. This operation is performed by the *source encoder*, which strives to represent the input messages with as few bits as possible. If the information source is digital, the source encoding does not require an analog to digital (A/D)

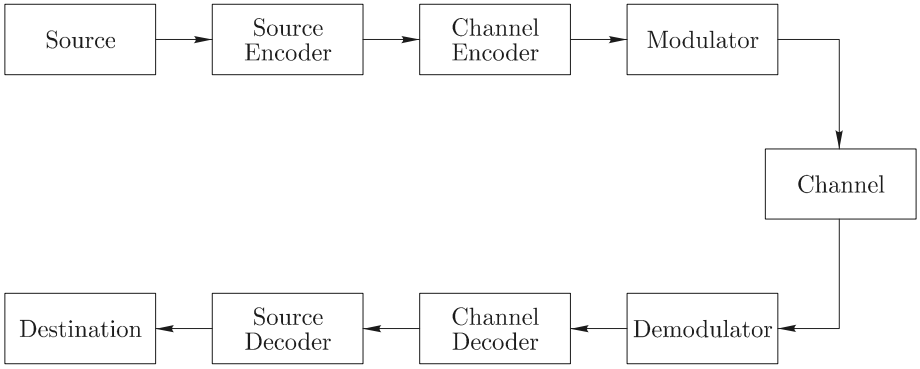


Figure 1.1: Block diagram of a communication system.

conversion. However, the task to represent the message with as few bits as possible remains. This process is called *data compression*, a process during which the amount of redundancy present in the source messages is reduced or, ideally, removed.

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 to be 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. Turbo codes – the topic of this thesis – belong to this type of codes, that is, channel codes.

The channel serves as a conveying medium over which the encoded messages are transmitted. In many practical situations, for example in radio communications, the channel is a waveform channel. Hence, it cannot be used to directly transmit the sequences of binary digits. Instead, the digital sequences must be converted into waveforms suitable for the specific characteristics of the channel – a task performed by the *modulator*. The waveforms output from the modulator are then corrupted in a way that depends on the characteristics of the channel. The corrupted waveforms are input to the *demodulator* whose function is the inverse of the modulator, *i.e.* to convert the received waveforms into a discrete-time sequence, ideally identical to the one that entered the modulator. Due to the noise and interference introduced by the channel, the demodulator output will not be exactly the same as the input to the modulator. This is where the channel decoder comes into play; its role is to counteract the channel-induced disturbances, by exploiting the redundancy introduced by the channel encoder.

In general, for optimal system performance the source encoder, channel en-

coder and modulator should be considered as a single function, and not separated into disjoint blocks as in Figure 1.1. However, Shannon showed in [84] that, asymptotically, source and channel encoding can indeed be separated into individual units, without loss of optimality. The same does not hold for the channel coding and the modulator functions. For example, it is well known that *trellis coded modulation* (TCM) [92, 93], which is a technique that combines channel coding and modulation, can achieve higher performance than methods that consider channel coding and modulation separately.

1.1.1 Channel Models

When studying and comparing channel codes, it is convenient to consider the modulator and demodulator as being part of the channel. The result is a composite channel with discrete-time inputs and discrete-time outputs, characterized by the possible inputs, the possible outputs, and transition probabilities that relates the outputs to the inputs. In general, these transition probabilities may vary with time, and they may also be correlated from one time to another. Further, the channel output does not necessarily depend on the current input only; it may depend also on previous channel inputs. In such cases, the channel is said to have memory. The channels encountered in mobile communication systems are examples of channels that often must be modeled as time varying and with memory. These effects arise from the combination of movement and the numerous reflections of the transmitted radio wave, caused by the propagation environment between the transmitting and receiving antennas.

This thesis is about understanding and designing Turbo codes. Due to the complex nature of these codes, it is desirable to use fairly simple channel models which do not introduce 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. As suggested by the model name, the channel output Y is modeled by adding a Gaussian distributed random variable to the channel input X, that is,

$$Y = X + G \tag{1.1}$$

where G is a zero mean Gaussian random variable with variance σ^2, *i.e.* $G \sim \mathcal{N}\left(0, \sigma^2\right)$. For a given input symbol $X = x$, the channel output Y is a Gaussian distributed random variable with mean x and variance σ^2, that is,

$$p_{Y|X}\left(y|x\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-x)^2}{2\sigma^2}}, \tag{1.2}$$

where $p\left(\cdot\right)$ denotes a probability density function of a random variable.

A simplified communication model where the modulator/demodulator pair and the waveform channel is replaced by a discrete-time channel is shown in Figure 1.2. In this model, further simplifications has been made by replacing

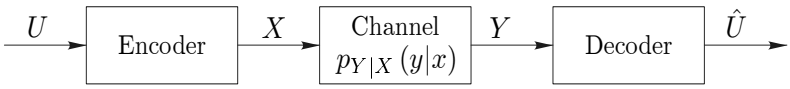


Figure 1.2: A communication model where the waveform channel and the modulator/demodulator pair are replaced by a discrete-time channel.

the source and the source encoder with a sequence of binary symbols. These symbols are independent and identically distributed (i.i.d.) random variables, with equal probability of being 0 and 1. This corresponds to an ideal situation in which the source encoder has removed all the redundancy present in the original information sequence. The result of these simplifications is a more confined model which is frequently used in studies and comparisons of channel codes [17, 18, 67].

1.1.2 Channel Coding

The purpose of channel coding is to make the transmitted messages less susceptible to the noise introduced by the channel. This is achieved by adding structured redundancy to the transmitted sequence, thereby increasing the number of bits to transmit. There are two basic types of codes used to introduce this redundancy: *block codes* [69] and *convolutional codes* [58]. A block code maps an information word of length k to a codeword of length n, where $n > k$. A convolutional encoder is instead continuously introducing redundancy to a stream of incoming symbols. For both types of codes, the code rate R is the number of information symbols transmitted per codeword symbol, that is, for block codes $R = k/n$. This thesis is restricted to investigations of binary codes, for which each symbol can take on one of two possible values, for example 0 and 1.

Thanks to the redundancy added by the channel encoder, the receiving end can *detect* and *correct* errors introduced by the channel. Channel codes that are designed and used for the latter purpose are called *error correcting codes*. Turbo codes belong to this group of codes. The performance of an error correcting code is measured in, for example, the signal-to-noise ratio (SNR) required to obtain communication with a certain bit or packet error rate. Typically, the SNR required to achieve a certain error rate is considerably lower when using error correcting coding compared to uncoded signalling. However, there are two drawbacks using error correcting codes: it increases the number of channel accesses and thereby the required modulation bandwidth, and it leads to an increased implementation complexity in transmitters and receivers.

Owing to the theories presented by Shannon in 1948 [84], it has been long known that there exist codes that achieve certain performance limits. However, the theories do not reveal how to design codes that perform as good as proved possible. Consequently, communication theorists and engineers have been in

quest for such codes for many years. In the next section some fundamental limits and bounds are reviewed, which are useful when placing the performance of a code or a system into perspective.

1.1.3 Performance Bounds

As part of the mathematical theory of communication, Shannon defined the concept of *channel capacity* [84]. The channel capacity is a measure of the amount of information that can be conveyed between the input X and the output Y of a channel. The capacity measure is related to the mathematical definition of information; the *average mutual information* between the continuous random variables X and Y, in bits, is defined as [18]

$$I(X;Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p_{X,Y}(x,y) \log_2 \frac{p_{X,Y}(x,y)}{p_X(x)\, p_Y(y)} dy dx, \tag{1.3}$$

where $p_{X,Y}(x,y)$, $p_X(x)$ and $p_Y(y)$ are joint and marginal probability densities for X and Y. The unit of "bits" follows from the base of the logarithm; for example, if the natural logarithm is used instead, the unit is "nats". The channel capacity C is defined [18] as the maximum value of $I(X;Y)$, maximized over the input distribution $p_X(x)$, that is,

$$C = \max_{p_X(x)} I(X;Y), \tag{1.4}$$

measured in bits per channel use. The relevance of the capacity C is asserted by the *channel coding theorem* [56, 84]. The channel coding theorem provides a relationship between the channel capacity C and the information transmission rate R:

For any rate $R < C$, it is possible to achieve arbitrarily small error probability by using sufficiently large codes. Conversely, for rates $R > C$, the error probability is lower bounded by some positive number.

For the discrete-time AWGN channel with continuous inputs and continuous outputs, the channel capacity is [18]

$$C = \frac{1}{2} \log_2 \left(1 + \frac{\sigma_X^2}{\sigma^2}\right) \text{ bits/channel use}, \tag{1.5}$$

where the channel input power is limited by $E\left[X^2\right] \leq \sigma_X^2$, and σ^2 is the noise variance. In practice, the physical waveform channel embedded in the discrete-time channel is band-limited. Thus, it is not possible to indefinitely increase the transmission rate by accessing the channel indefinitely often. For an AWGN waveform channel limited to a bandwidth of W Hz and a double-sided noise power spectral density $N_0/2$, the capacity is [18]

$$C = W \log_2 \left(1 + \frac{P}{N_0 W}\right) \text{ bits/s}, \tag{1.6}$$

where P is the average input power. Letting the channel bandwidth W increase towards infinity, the infinite-bandwidth channel capacity C_∞ is found as

$$C_\infty = \frac{P}{N_0} \log_2 e. \tag{1.7}$$

The channel input power can be expressed as $P = E_b r_b$, where E_b is the transmitted energy per information bit and r_b is the information rate in bits/s. Combining the channel coding theorem with the infinite-bandwidth channel capacity (1.7) yields $r_b < \frac{P}{N_0} \log_2 e$, which leads to a fundamental requirement for reliable communication:

$$\frac{E_b}{N_0} > \frac{1}{\log_2 e} = \ln 2. \tag{1.8}$$

where ln is the natural logarithm. Hence, it is not possible to design a communication system having arbitrarily low error rate if $E_b/N_0 < -1.6$ dB. On the other hand, as long as $E_b/N_0 > -1.6$ dB, it is in theory possible to achieve an arbitrary low error probability by using sufficiently large codes.

The above bound provides a lower limit on the SNR in terms of energy-per-bit to noise-spectral-density ratio (E_b/N_0) required in order to achieve error-free communication. However, the assumptions used to derive the limit are too optimistic for most practical situations. Firstly, the channel bandwidth is in reality limited. Secondly, the derivation of the channel capacity allows the transmitted codewords to be infinitely long, thus introducing an infinite delay in the transmission. Clearly, practical communication systems are designed to transmit the information within a limited amount of time. Furthermore, the limit is derived under the assumption of a continuous input, whereas digital communication systems use modulation schemes with finite alphabet sizes. Due to these assumptions, the required E_b/N_0 for a specific situation is in reality higher than -1.6 dB. The sphere packing lower bound, whose properties are summarized below, takes some of these assumptions into account, thereby offering a bound that is closer to realistic conditions.

The Sphere-Packing Bound

The sphere-packing bound [28, 85] provides a lower bound that includes parameters such as the bandwidth expansion and the transmission delay. Further, it is valid for a certain specified error rate, in contrast to error-free communication. For evaluation and comparison of channel codes, the bandwidth expansion and the transmission delay can be expressed in terms of code rate R and code block length n. Given these, the bound provides the minimum E_b/N_0 required on an AWGN channel in order to achieve a certain frame-error rate (FER).

Figure 1.3 shows the minimum E_b/N_0 required to obtain frame-error probabilities of 10^{-2}, 10^{-4}, 10^{-6} and 10^{-8} as functions of the information block length k, using rate-1/3 coding ($n = 3k$). As expected, the E_b/N_0 required for

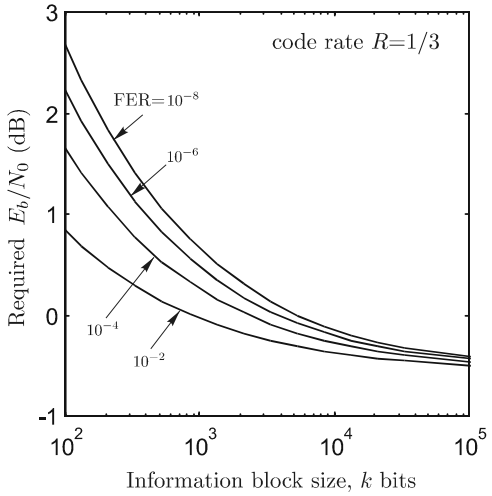


Figure 1.3: Sphere-packing lower bounds on the required E_b/N_0 as a function of the information block size, for code rate $R = 1/3$ and frame error rates ranging from 10^{-8} to 10^{-2}.

a certain FER is lower the larger the block length. Furthermore, the figure illustrates that the required increase in E_b/N_0 for lowering the frame error rate is higher for small block sizes. Figure 1.4 shows the corresponding sphere-packing bounds on the SNRs required to achieve a frame error rate of 10^{-4} for code rates ranging from 1/2 to 1/8. Assuming that performances according to the bounds in Figure 1.4 are achievable, the gain achieved by decreasing the code rate is practically independent of the information block size k (for $k \geq 100$), measured in reduced E_b/N_0-requirement.

Bounds Based on Code Properties

The performance of a communication system is related to how successful the receiver is in distinguishing between transmitted messages. The less alike these messages are, the easier for the receiver to discern the transmitted message from the others. For channel codes the amount of differences between codewords is summarized by the *Hamming distance spectrum* or equivalently, for linear codes, the *weight distribution*[1] [69]. The Hamming distance between two codewords is the number of positions/symbols the codewords differ. The Hamming weight of a codeword is the number of non-zero symbols in the codeword. The weight distribution of a code is a listing of the number of codewords having each possible

[1] The terminology 'distance spectrum' and 'weight distribution' are used interchangeably throughout this thesis.

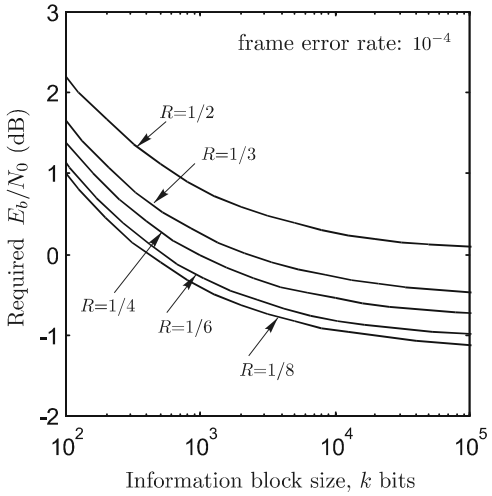


Figure 1.4: Sphere-packing lower bounds on the required E_b/N_0 as a function of the information block size, for code rates ranging from 1/8 to 1/2 and a frame error rate of 10^{-4}.

weight, ranging from 0 to n. These numbers are referred to as *multiplicities*, denoted a_d, $d = 0, 1, \ldots, n$. Thus,

$$a_d = \text{the number of codewords with Hamming weight } d.$$

The smallest distance $d > 0$ for which $a_d \neq 0$ is called the *minimum distance* of the code, denoted $d_{\min}$. The minimum distance of a code has a central role in code design, since it has a large influence on the performance of the code. In general, code design aims at constructing codes with as large minimum distance as possible.

In the following, a few useful bounds based on the distance spectrum of a code are reviewed. These bounds assume that the decoder uses a maximum likelihood (ML) decoding algorithm, which minimizes the decoding error probability (if all information sequences are equally likely) [67]. On an AWGN channel, the probability that an ML-decoder will choose a codeword c_j in favor of a codeword c_i, the transmitted codeword, is [67]

$$\Pr\left(c_i \longrightarrow c_j\right) = Q\left(\sqrt{\frac{2d_{ij}RE_b}{N_0}}\right), \tag{1.9}$$

where d_{ij} is the Hamming distance between codeword c_i and c_j, R is the code rate, and $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$ is the upper tail probability of a normalized Gaussian random variable.

A Lower Bound. A lower bound on the decoding performance is obtained by using the probability that a decoding error is made to a specific codeword at distance $d_{\min}$. Denoting the decoding error probability P_e, a lower bound is

$$P_e \geq Q\left(\sqrt{\frac{2d_{\min}RE_b}{N_0}}\right). \tag{1.10}$$

The Union Bound. An upper bound on the decoding error probability based on the pair-wise probability that a transmitted codeword is erroneously decoded to each one of the other codewords is called the union bound [67, 71, 74, 76, 83]. The union bound is formed by summing the pair-wise error probabilities that a transmitted codeword c_i is mistaken for the codeword c_j for all $j \neq i$:

$$P_e \leq \sum_{\substack{j=1 \\ j\neq i}}^{M} \Pr\left(c_i \longrightarrow c_j\right) = \sum_{\substack{j=1 \\ j\neq i}}^{M} Q\left(\sqrt{\frac{2d_{ij}RE_b}{N_0}}\right), \tag{1.11}$$

where M is the number of codewords in the code. Using the weight distribution of the code, the union bound can be formulated as

$$P_e \leq \sum_{d=d_{\min}}^{n} a_d Q\left(\sqrt{\frac{2dRE_b}{N_0}}\right), \tag{1.12}$$

where a_d is the number of codewords with Hamming weight d. For large E_b/N_0:s, the union bound is dominated by the first terms in (1.12), which means that it is mainly influenced by the lower part of the distance spectrum. For low E_b/N_0:s, the bound is influenced by a larger number of terms. Especially, it rapidly becomes excessively pessimistic when the *computational cutoff* rate R_0 approaches the code rate R. The computational cutoff rate is a practical limit on the highest rate at which a sequential decoder can operate [17]. In practice, a sequential decoder that operates at rates $R > R_0$ will have severe computational loading causing buffer overflows, as a consequence of the large decoding complexity. For an AWGN channel, the cutoff rate is $R_0 = 1 - \log_2\left(1 + e^{-RE_b/N_0}\right)$ [17].

The Tangential Sphere Bound. Since the introduction of Turbo Codes in 1993, it is known that there exist codes that can be decoded with reasonable complexity at SNRs for which the cutoff rate is lower than the code rate. Hence, there is a need for improved upper bounds, which are tighter than the union bound at low SNRs. In [73], Poltyrev derived the *tangential sphere bound*, which was shown in [43] to be tighter than both the union bound and the *tangential bound* presented by Berlekamp in [11].

Figure 1.5 shows both the union bound and the tangential sphere bound for an example distance spectrum derived for rate-1/3 coding. As seen, the union

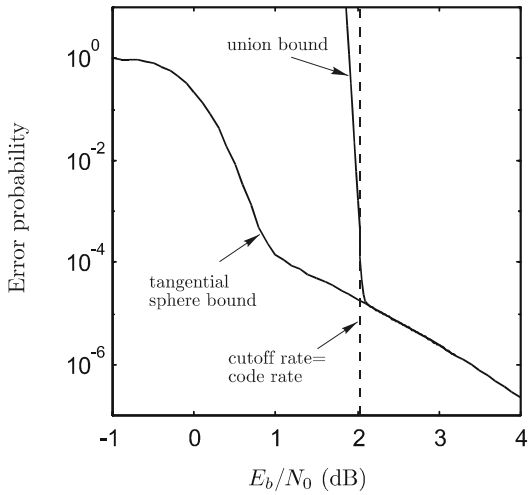


Figure 1.5: Example of the union bound and the tangential sphere bound on the decoding error probability. (The bounds are derived using the average spectrum for 500-bit Turbo codes with $(35/23)_8$ constituent encoders.)

bound begins to grow larger than 1 when the cutoff rate approaches the code rate, while the tangential sphere bound is potentially realistic (*i.e.* < 1) for the entire range of E_b/N_0:s. The region of useful SNRs is thus considerably larger for the tangential sphere bound than for the union bound. This is indeed useful for evaluations of Turbo codes since these are known to operate above the cutoff rate, that is, they yield low error rates even at E_b/N_0:s for which the cutoff rate is lower than the code rate. In Figure 1.5, this corresponds to $E_b/N_0 < 2$ dB. An example of such performance is shown in the following section.

1.2 Summary and Motivation of Research Area

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 exists 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 [13] constitutes a very important step forward, both in the search of good codes as well as in the search of efficient decoding algorithms. An example of the remarkable performance obtainable

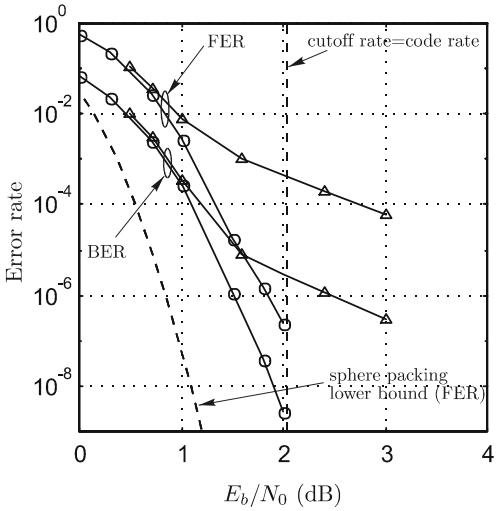


Figure 1.6: Example of Turbo code frame- and bit-error rate performances. Triangles: random interleaving, circles: designed interleaver.

with Turbo codes[2] is illustrated in Figure 1.6. The code that corresponds to the circle-marked lines operates above the cutoff rate at frame- and bit-error rates below 10^{-6} and 10^{-8}, respectively, which is indeed remarkable. Further, for frame error rates down to 10^{-6}, the performance of this code is less than 1 dB above the sphere-packing lower bound.

The Turbo code corresponding to the circled-marked lines in Figure 1.6 uses a designed interleaver[3]. In contrast, the triangle-marked performances correspond to Turbo codes using random interleaving, which corresponds to the average performance of Turbo codes using this particular interleaver size. As seen, the average Turbo code performance suffers from a rather severe 'error-floor', whereas the Turbo code that uses a designed interleaver shows no error-floor behavior even at frame error rates as low as 10^{-7}. The issue of investigating and understanding the mechanisms of Turbo codes that give rise to their performances, both when good and when bad, is the focus of this thesis. This includes investigation of the central Turbo coding components including the interleaver, the constituent encoders[4], as well as aspects such as trellis termination[5].

[2] These rate-1/3 Turbo codes use 500-bit interleavers, 8-state encoders and a maximum of 15 decoding iterations with the BCJR-algorithm.

[3] The interleaver is one of the central parts of Turbo codes, which is introduced in Section 2.1.2.

[4] The constituent encoders are introduced in Section 2.1.1.

[5] Trellis termination for Turbo codes is introduced in Section 2.1.3.

Chapter 2

A Turbo Code Preliminary

"...good codes just might be messy"
J. L. Massey

Turbo codes where introduced by Berrou, Glavieux and Thitimajshima in their groundbreaking paper *"Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes"* [13]. 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. The reason for the doubts are not far-fetched: the performance results presented were significantly better than what had previously been seen and, more important, what was anticipated feasible. In fact, the paper presented results of communication over the AWGN channel less than 1 dB above the lower 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 2.1. 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 codeword. By the use of sophisticated algorithms, the decoders can exchange information on their decoding results and thereby cooperate in finding the correct codeword. 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

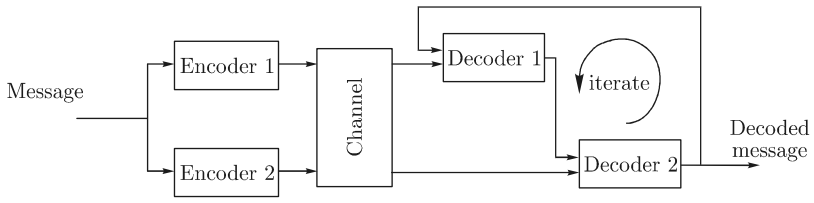


Figure 2.1: The Turbo coding/decoding principle.

again, with the functionality of a Turbo combustion engine[1]. Hence, the term "Turbo" is indicative on the decoding method rather than of 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 proposed a new code construction: parallel concatenation of recursive convolutional encoders. Through the combination of this code construction and the iterative decoding algorithm, the authors had found a way to construct very large and complex codes, which can be decoded with high performance and reasonable complexity. This combination is very appealing, in light of the channel coding theorem which is derived on the basis that the codeword length increases towards infinity. The encoder structure and the design of its components is the topic of this thesis, and it will be discussed in the sequel.

A last comment on the terminology of Turbo codes. There is no clean-cut definition of what Turbo codes are. For example, it is possible to exchange the constituent codes to any other type of code, for example block codes. Such constructions are called *Block Turbo Codes* [75]. Further, it is possible to concatenate several constituent codes by arranging additional encoders in parallel [25, 42]. Such constructions are still denoted Turbo codes, or sometimes *multiple Turbo codes*. In the investigations reported in this thesis, we are exclusively referring to the original encoder structure presented by Berrou *et al.* in [13], that is, parallel concatenation of two recursive systematic convolutional encoders.

2.1 Encoder Structure

A general Turbo encoder structure using two constituent encoders is illustrated in Figure 2.2. It consists of three basic building blocks: an interleaver, the constituent encoders, and a puncturing and multiplexing unit. The interleaver is a device that re-orders the symbols in its input sequence. Together with the lower constituent encoder, the interleaver can be considered to implement 'Encoder 2' in Figure 2.1. In the following, the building blocks used in a Turbo code encoder, as depicted in Figure 2.2, are described.

[1] Turbochargers utilize energy in the exhaust gases to boost the air intake.

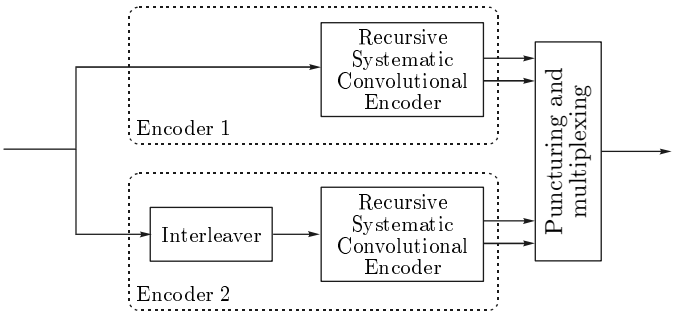


Figure 2.2: Block diagram of a Turbo code encoder.

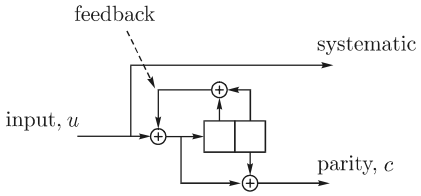


Figure 2.3: Recursive systematic convolutional encoder with feedback polynomial 7_8 and parity polynomial 5_8.

2.1.1 The Constituent Encoders

The constituent encoders are *recursive systematic convolutional* (RSC) encoders, *i.e.* systematic convolutional encoders with feedback. Such an encoder with two memory elements is depicted in Figure 2.3. For systematic codes, the information sequence is part of the codeword, which corresponds to the direct connection from the input to one of the outputs. For each input bit the encoder generates two codeword bits: the systematic bit and the parity bit. Thus, the code rate is 1/2. The encoder input and parity bits are denoted u and c, respectively.

A convenient and common way to specify a convolutional encoder is to use the octal representation of the polynomial in D that describes the encoder. The delay operator D indicates a delay of one symbol time; D^n indicates a delay of n symbol times. Thus, the polynomial in D that describes the parity bit in Figure 2.3, as a function of the encoder memory contents, is $1 + D^2$. Similarly, the feedback polynomial is $1 + D + D^2$. The generator matrix of this encoder, in polynomial representation is $G(D) = \left(\begin{array}{cc} 1 & \frac{1+D^2}{1+D+D^2} \end{array} \right)$. Representing the coefficients in the polynomials as binary numbers, the parity and feedback polynomials can be represented by 101_2 and 111_2 respectively, which is 5_8 and 7_8 in

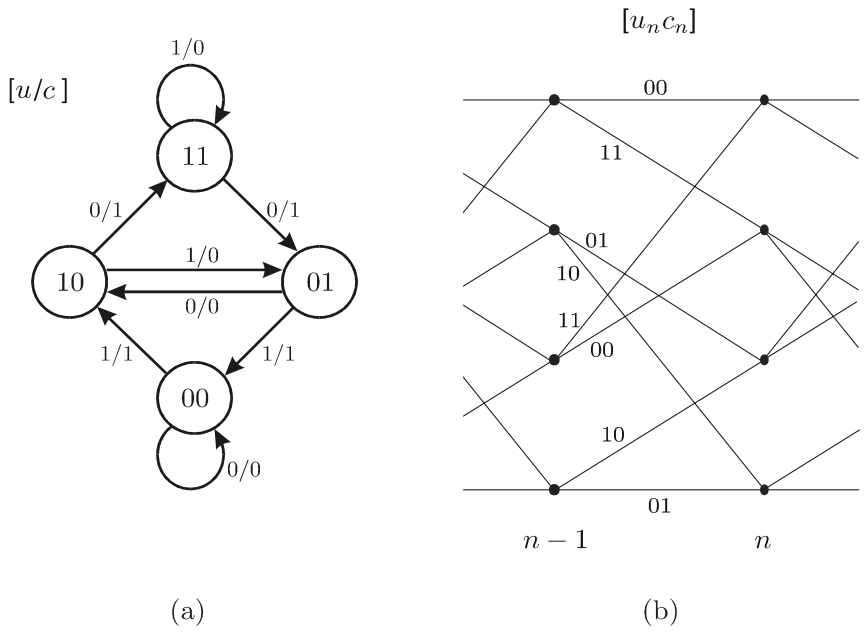


Figure 2.4: (a) State- and (b) trellis-diagrams for the convolutional encoder in Figure 2.3.

octal representation[2]. A compact representation of the encoder in Figure 2.3 is thus $\begin{pmatrix} 1 & 5/7 \end{pmatrix}_8$. Throughout this thesis, the parity polynomial is assumed to be monic with the same degree as the feedback polynomial.

The functionality of convolutional encoders is conveniently described by *state-* and *trellis-*diagrams, shown in Figure 2.4. Each state in the state-diagram corresponds to the contents of the memory elements in the encoder, and the state-transitions are labeled with the input and parity bit corresponding to each transition. The trellis-diagram includes a time-dimension, hereby incorporating the history of the encoder. Each codeword belonging to the code is made up by the labels $[u_n c_n]$ on the trellis transitions that correspond to a specific path through the trellis.

In general, the constituent encoders can be of any code rate. In the original Turbo code paper, Berrou *et al.* proposed systematic rate-1/2 encoders. Since then, lower rate Turbo coding schemes have also been investigated, *e.g.* in [8, 25, 62, 65].

[2] The binary digits can be partitioned into 3-tuples starting both from the most significant bit (MSB) and the least significant bit (LSB). Throughout this thesis, the octal representation is based on partitioning from the LSB.

2.1.2 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 Section 2.3 below. The reason that the interleaver enables the use of iterative decoding is that it 'de-correlates' nearby[3] decoder inputs. This is essential, since nearby decoder inputs that are correlated have a detrimental influence on the decoding performance. This issue is further studied in Chapter 3, addressing interleaver design criteria.

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 of length N. The re-ordering is then performed within each such block. Typically, the performance of a Turbo code is improved when the interleaver size is increased, which has an positive influence on both the code properties and the iterative decoding performance.

There are many possibilities of representing the interleaver rule. Among the most common is the use of a vector $\boldsymbol{\pi}$ of length N, 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 interleaving rule can be represented by the vector $\boldsymbol{d}$, where $d(i)$ holds the input position that is interleaved to position i. The former representation may seem more natural, but in conjunction with the interleaver design algorithm described in Chapter 3, the use of $\boldsymbol{d}$ is more convenient.

A third way of representing interleavers is by using permutation matrices. A permutation matrix is an $N \times N$ matrix containing exactly one 1 in each row and in each column, all other elements being zero. One of the benefits of using permutation matrices is that they efficiently visualize interleaver structures. For example, Figure 2.5 shows two permutation matrices, where each 1 is represented by a dot. To the left, the permutation matrix of a pseudo-random interleaver, and to the right, a 10×10 block[4] interleaver, both of length 100 bits. The deterministic and structured properties of the block interleaver, compared to the pseudo-random interleaver, are obvious. Other interleaver properties that are nicely visualized with permutation matrices are discussed in Chapter 3, in conjunction with interleaver design and iterative decoding performance.

[3] With 'nearby' is meant positions that are close to each other in the input sequence.

[4] A block interleaver is obtained by writing the elements to a matrix row-wise and reading them column-wise.

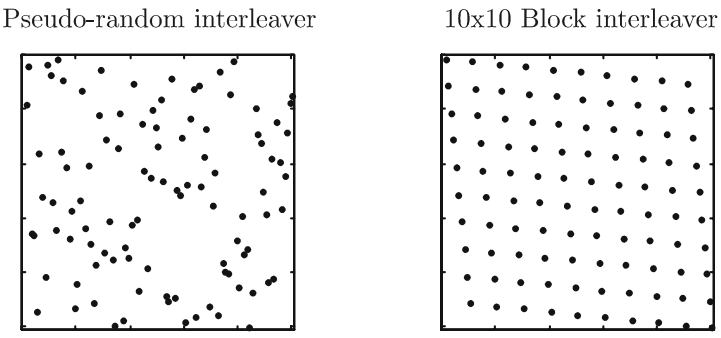


Figure 2.5: Illustration of the permutation matrices of two 100-bit interleavers. Left: a pseudo-random interleaver. Right: a 10×10 block interleaver.

2.1.3 Trellis Termination

As mentioned in the introductory chapter, and to be further addressed in later sections, the performance of a code is highly dependent on its Hamming distance spectrum. For convolutional codes, to which the constituent codes in Turbo codes belong, the Hamming distances between codewords are the result of taking different paths through the trellis. In principle, the larger the number of trellis transitions in which two paths differ, the larger is the possible Hamming distances between the corresponding codewords. It is thus desirable that the shortest possible detour from a trellis path is as long as possible, to ensure a large Hamming distance between the two codewords that correspond to the two paths.

For convolutional codes that are truncated at some point, which is the case of block-oriented Turbo codes, severe degradation of the distance spectra may occur. If no precautions are taken before the truncation, each of the encoder states is a valid ending state and thus the shortest possible difference between two trellis paths is made up of only one trellis transition. Naturally, this procedure may result in very poor distance properties, with accompanying poor error correcting performance.

A solution to the above problem is *trellis termination*. With trellis termination, ν *tail bits* are appended to the information sequence, where ν is the number of memory elements in the encoder. The tail bits are chosen so that, after encoding these bits, the encoder is terminated in the zero-state. This way, the shortest possible trellis detour is the same as without truncation, and the distance spectrum is preserved. Using feed-forward convolutional encoders, *i.e.* encoders without feedback, trellis termination is easily obtained by appending tail bits that are all zeros. For recursive encoders, however, the situation becomes more complicated, since the tail bits depend on the information sequence.

A second approach to the problem of trellis truncation is *tail-biting*. With

tail-biting, the encoder is initialized to the same state as it will end up in by the end of the block. For feed-forward encoders tail-biting is readily obtained by inspection of the last ν bit in the input sequence, since these dictate the encoder ending state. However, as in the case of trellis termination, the situation becomes more complicated when using recursive encoders [57].

The advantage of using tail-biting compared to trellis termination is that tail-biting does not require transmission of tail bits (the use of tail bits reduces the code rate and increases the transmission bandwidth). For large blocks, the rate-reduction imposed by tail-bits is small, often negligible. For small blocks, however, it may be significant.

In this thesis, the issue of trellis termination is studied in [Paper VI]. Additional investigations on trellis termination and tail-biting for Turbo codes are found in [5, 14, 26, 34, 38, 48, 51, 55] and [2, 20, 88, 95, 96], respectively.

2.1.4 Puncturing

Puncturing is the process of removing certain symbols/positions from the codeword, thereby reducing the codeword length and increasing the overall code rate. In the original Turbo code proposal, Berrou *et al.* punctured half the bits from each 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 overall code rate is $R = 1/2$. The most common alternative to this is not to puncture the parity bits of either constituent encoder, which results 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 effect on different choices of interleavers, and on different constituent encoders. Thus, as we are interested in the issues of interleaver design and the choice of constituent encoders, we have not used puncturing in our investigations. Consequently, the overall Turbo code rate of the codes presented in this thesis is $R = 1/3$.

An equivalent Turbo code encoder that corresponds to the assumptions outlined above is shown in Figure 2.6. Here, the systematic bit stream is separated from both encoders, but it can of course still be considered part of both constituent codes. Further, the multiplexing unit has been omitted. This is convenient when discussing the decoding algorithm, which make use of the different bit streams separately.

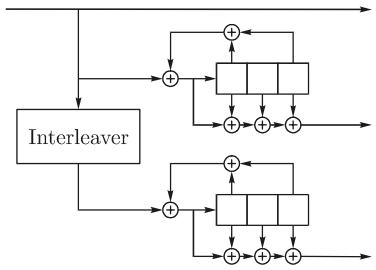


Figure 2.6: Equivalent structure of a Turbo code encoder, using constituent encoders with generator polynomials $(17/15)_8$ and no puncturing.

2.2 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 a Turbo code. A block diagram of an iterative decoder that matches the encoder in Figure 2.6 is shown in Figure 2.7. 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 decoder processes input blocks of size N, *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 soft information of length N 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 results from the second decoder it is conceivable 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* probabili-

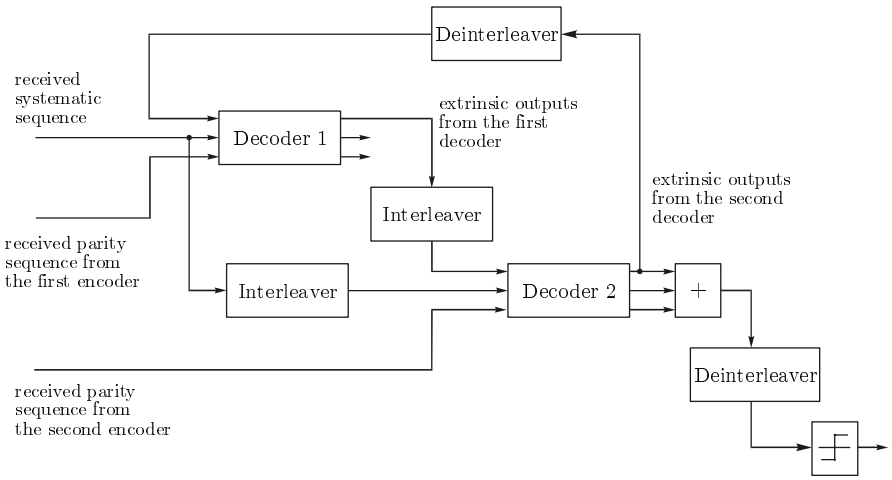


Figure 2.7: Block diagram of an iterative Turbo code decoder.

ties, 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 $\Pr(u_n = 1)$ and $\Pr(u_n = 0)$. For binary information symbols, it is convenient to represent the *a priori* information as a log-likelihood ratio[5] (LLR). Thus, denoting the nth *a priori* input $\tilde{\Lambda}_n$, we have

$$\tilde{\Lambda}_n = \ln \frac{\Pr(u_n = 1)}{\Pr(u_n = 0)}. \tag{2.1}$$

The input to the decoder consists of three sequences, for the rate-1/2 constituent encoders considered. These inputs are: the received systematic sequence, x_1^N, the received parity sequence, y_1^N, and the *a priori* input sequence, $\tilde{\Lambda}_1^N$. These are sequences are in short referred to as R_1^N, that is,

$$R_1^N = \left(x_1^N, y_1^N, \tilde{\Lambda}_1^N\right). \tag{2.2}$$

The task of an APP decoder is to derive the *a posteriori* probabilities $\Pr\left(u_n = 1 | R_1^N\right)$ and $\Pr\left(u_n = 0 | R_1^N\right)$, where u_n is information symbol n. Similar to the *a priori* input, the decoder soft-output, denoted Λ_n, is represented

[5] The term *log-likelihood ratio* is used for all logarithms of probability-ratios in this thesis. Even though this is often the case in turbo coding literature, it can be argued that it is an abuse of terminology.

as an LLR

$$\Lambda_n = \ln \frac{\Pr\left(u_n = 1 | R_1^N\right)}{\Pr\left(u_n = 0 | R_1^N\right)} \qquad n = 1, 2, \ldots, N. \tag{2.3}$$

The decoder hard decision is achieved by using the sign of the APP soft-output; if $\Lambda_n > 0$, the decoder hard decision is $\hat{u}_n = 1$, 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 on to the next constituent decoder. Assuming that the encoder inputs are independent, the LLR (2.3) can be partitioned into three parts (see Appendix A for details):

$$\begin{aligned} \Lambda_n &= \ln \frac{\Pr\left(u_n = 1 | \tilde{\Lambda}_n\right)}{\Pr\left(u_n = 0 | \tilde{\Lambda}_n\right)} + \ln \frac{p\left(x_n | u_n = 1\right)}{p\left(x_n | u_n = 0\right)} + \\ &\quad \ln \frac{\Pr\left(u_n = 1 | x_1^{n-1}, x_{n+1}^N, y_1^N, \tilde{\Lambda}_1^{n-1}, \tilde{\Lambda}_{n+1}^N\right)}{\Pr\left(u_n = 0 | x_1^{n-1}, x_{n+1}^N, y_1^N, \tilde{\Lambda}_1^{n-1}, \tilde{\Lambda}_{n+1}^N\right)}. \end{aligned} \tag{2.4}$$

Thus, the *a posteriori* LLR Λ_n is composed of three LLRs. The first LLR contains the contribution from the input *a priori* information $\tilde{\Lambda}_n$. The second LLR is denoted Li_n and called the *intrinsic information*, since it contains the contribution directly from systematic channel observation x_n. The third LLR is denoted Le_n and called the *extrinsic information*, since its contribution to Λ_n stems from other sources than the *a priori* information $\tilde{\Lambda}_n$ and the systematic channel observation x_n. Noting that $\Pr\left(u_n = i | \tilde{\Lambda}_n\right) = e^{i\tilde{\Lambda}_n} / \left(1 + e^{\tilde{\Lambda}_n}\right)$ (*cf.* (A.19) and (A.20) in Appendix A), the output LLR Λ_n can be expressed as

$$\Lambda_n = \tilde{\Lambda}_n + Li_n + Le_n. \tag{2.5}$$

In an iterative decoder, the *a priori* information originates from the previous constituent decoder; thus, it should not be included in the information that is passed on to that decoder in the next decoding step. Likewise, the intrinsic information Li_n is directly available to the next decoder through the channel observation x_n. Consequently, the only portion of Λ_n that is passed on to the next decoder is the extrinsic information Le_n. 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-iterations with a superscript (k), we have

$$\Lambda_n^{(k)} = Le_{n'}^{(k-1)} + Li_n + Le_n^{(k)}, \tag{2.6}$$

Figure 2.8: Inputs and outputs to each constituent encoder.

where the subscript n' indicates the re-ordering performed by the interleaver and deinterleaver; output n' from half-iteration $k-1$ is interleaved (if k even), or deinterleaved (if k odd), to position n before being passed to the kth half-iteration. Figure 2.8 shows an APP decoder, with inputs and outputs corresponding to the described iterative decoding situation. In the following section, algorithms that can be used to derive the APPs are discussed.

2.2.1 APP Decoding

A device capable of producing *a posteriori* probabilities of each information symbol based on the channel observations and *a priori* probabilities is called an *APP decoder*. An optimal algorithm for estimating state- and state transition probabilities of a Markov source was presented by Bahl *et al.* in 1974 [3]. The algorithm is suitable for decoding of convolutional codes, whose codewords can be viewed as the output from a Markov source. The algorithm, often referred to as the BCJR-algorithm[6], is optimal in the sense of minimizing the symbol error rate. 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. 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 [30, 94]. However, the Viterbi algorithm does not produce *a posteriori* probabilities. Therefore, modifications to the Viterbi algorithm have been proposed. In [7] Battail presented a method to estimate the reliability of symbols decoded with the Viterbi algorithm. A similar approach was taken by Hagenauer and Hoeher in [35], proposing the *Soft Output Viterbi Algorithm* (SOVA) which was later used for iterative decoding of Turbo codes in [37]. Lin and Cheng proposed improvements to the SOVA algorithm in [66], resulting in the same algorithm as originally proposed by Battail in [7]. The same modifications to the SOVA algorithm were independently proposed in [31], where it was also shown that this algorithm is equivalent to the Max-Log-MAP algorithm [78]. 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 [78]. Additional low-complexity

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

variants of the BCJR-algorithm, based on reducing the trellis search-space, are reported in [32, 33].

The BCJR Algorithm

Here, we give a brief review of the BCJR-algorithm; a full derivation of the is given in Appendix A. The encoder input is, as above, represented by $R_1^N = \left(x_1^N, y_1^N, \tilde{\Lambda}_1^N\right)$. However, in the iterative decoding environment the 'true' *a priori* information $\tilde{\Lambda}_1^N$ is replaced by the extrinsic output from the previous decoding stage. Hence, the encoder input at time n is $R_n = \left(x_n, y_n, Le_{n'}^{(k-1)}\right)$, where $Le_{n'}^{(k-1)}$ is the extrinsic output from the previous decoder that is input at bit position n in this decoding stage.

From the derivation in Appendix A, the decoder soft output decision variable after the kth half-iteration, *i.e.* the log-likelihood ratio $\Lambda_n^{(k)}$, can be expressed as

$$\Lambda_n^{(k)} = \ln \frac{\Pr\left(u_n = 1 | R_1^N\right)}{\Pr\left(u_n = 0 | R_1^N\right)} = \ln \frac{\sum_{s'} \sum_s \alpha_{n-1}\left(s'\right) \gamma_n^1\left(R_n, s', s\right) \beta_n\left(s\right)}{\sum_{s'} \sum_s \alpha_{n-1}\left(s'\right) \gamma_n^0\left(R_n, s', s\right) \beta_n\left(s\right)}, \quad (2.7)$$

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

$$\begin{aligned} \gamma_n^i\left(R_n, s', s\right) &= e^{\left(x_n(2i-1) + y_n(2c_n-1)\right)/\sigma^2 + iLe_{n'}^{(k-1)}} \cdot \\ &\quad \Pr\left(S_n = s | u_n = i, S_{n-1} = s'\right), \end{aligned} \quad (2.8)$$

where c_n is the parity bit associated with a transition between the states s' and s, caused by an information symbol $u_n = i$. The probability $\Pr\left(S_n = s | u_n = i, S_{n-1} = s'\right)$ is either 0 or 1, depending on if there exists a trellis transition between the states s' and s caused by an information symbol $u_n = i$. The α- and β-values are calculated recursively, starting from the beginning and the end of the received block, respectively:

$$\alpha_n\left(s\right) = \sum_{s'=0}^{2^\nu - 1} \sum_{i=0}^{1} \alpha_{n-1}\left(s'\right) \gamma_n^i\left(R_n, s', s\right) \quad (2.9)$$

$$\beta_n\left(s'\right) = \sum_{s'=0}^{2^\nu - 1} \sum_{i=0}^{1} \beta_{n+1}\left(s\right) \gamma_{n+1}^i\left(R_{n+1}, s', s\right), \quad (2.10)$$

where ν is the number of memory elements in the encoder. With encoders

initialized and terminated in their zero states, the recursions are initialized with

$$\alpha_0(s) = \begin{cases} 1 & s = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

and

$$\beta_N(s) = \begin{cases} 1 & s = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.12}$$

For encoders not terminated in the zero state by the end of each block, the initialization of the backward recursion is instead[7]

$$\beta_N(s) = \frac{1}{2^\nu}, \qquad s = 0, \ldots, 2^\nu - 1. \tag{2.13}$$

Finally, after calculating the decoder soft-output $\Lambda_n^{(k)}$, the extrinsic information is found as

$$Le_n^{(k)} = \Lambda_n^{(k)} - Le_{n'}^{(k-1)} - Li_n, \tag{2.14}$$

where $Le_{n'}^{(k-1)}$ is the *a priori* input and Li_n is calculated as

$$Li_n = \frac{2}{\sigma^2} x_n. \tag{2.15}$$

2.3 Distance Spectra

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 estimations since, at moderate to high SNRs, a large part of the decoding errors are made to codewords 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 specific constituent encoders and a certain interleaver length N, where the average is taken over the ensemble of all interleavers. The presented method made possible several important observations regarding the unexplained and unmatched performances of Turbo codes. The method is summarized in [Paper VI]; for a thorough description *cf.* [10]. Additional work related to the distance spectra of Turbo codes is found in for example [15, 23, 24, 27, 51, 72, 89].

In the following sections the structure of Turbo codes is exploited to illustrate the origin of the lower part of the distance spectrum of a Turbo code. These observations also serve as an introduction to the criteria required for successful interleaver design, addressed in Chapter 3.

[7] Several alternatives to the initialization of the backward recursion for non-terminated trellises have been proposed. A discussion on this subject is found in [Paper VI].

2.3.1 Distance Spectrum of a Specific Turbo Code

Due to the parallel structure of Turbo code encoders, the Hamming weight of a codeword is made up of three separate parts: the weight of the systematic sequence, the weight of the first parity sequence, and the weight of the second parity sequence. Let $\boldsymbol{u}$ denote the systematic sequence, *i.e.* $\boldsymbol{u} = u_1^N = (u_1, u_2, \ldots, u_N)$, and $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ the corresponding parity sequences. Further, let $\boldsymbol{c}$ denote the concatenated Turbo code codeword, so that $\boldsymbol{c} = (\boldsymbol{u}, \boldsymbol{c}_1, \boldsymbol{c}_2)$. Then, the Hamming weight of a Turbo code codeword is $w_H(\boldsymbol{c}) = w_H(\boldsymbol{u}) + w_H(\boldsymbol{c}_1) + w_H(\boldsymbol{c}_2)$, where $w_H(\cdot)$ denotes the Hamming weight operator. For notational convenience, the short notations $w \triangleq w_H(\boldsymbol{u})$, $p_1 \triangleq w_H(\boldsymbol{c}_1)$, $p_2 \triangleq w_H(\boldsymbol{c}_2)$ are used in the following. When appropriate, the sequences are represented by their corresponding D-transform. For example, the input sequence $\boldsymbol{u}$ can be represented by $U(D) = \sum_{k=1}^{w} D^{i_k}$, where $i_k \in \{0, \ldots, N-1\}$ corresponds the position of the kth input 1. Similarly, the interleaved input sequence $\boldsymbol{u}'$ is $U'(D) = \sum_{k=1}^{w} D^{i'_k}$, where $i'_k \in \{0, \ldots, N-1\}$ represents the position of the kth input 1 in the interleaved sequence. Note that i'_k is not necessarily the interleaved position of i_k, since these indices are ordered from the first to last input 1's in the respective sequence.

Figure 2.9 depicts the state-diagrams of two recursive convolutional encoders, having generator polynomials $(17/15)_8$ and $(15/17)_8$, respectively. Since the encoders are recursive, a single input 1 will cause the encoder to cycle in a loop of states. Such *zero-input loops* are marked with solid lines in Figure 2.9. For encoders with *primitive* feedback polynomials [54], there exist only one zero-input loop. However, for non-primitive polynomials several zero-input loops exist. In such cases, 'the zero-input loop' refers to the loop entered directly from the zero-state, following an input-1 transition. Let L denote the length of the zero-input loop, in terms of the number of state transitions required to fulfill one cycle, and p_{loop} the number of parity 1's generated by each cycle in this loop. The length of the zero-input loop corresponds to the periodicity of the infinite impulse response of the encoder, and it is also referred to as the period of the feedback polynomial. For the encoders in Figure 2.9, L is 7 and 4 respectively, while p_{loop} is 4 and 2. For memory-3 encoders, the maximum value of L is 7. This follows from the relation [74]

$$L \leq 2^{\nu} - 1, \tag{2.16}$$

where ν is the number of memory elements. In general, (2.16) hold with equality for primitive feedback polynomials, thus maximizing L for a given number of encoder memory elements.

Assume that all codewords with Hamming weight less than or equal to a threshold d_{max} are to be found. A low-weight codeword must have low weight on all three of the concatenated sequences, *i.e.* $\boldsymbol{u}$, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. To begin with, the weight of the input sequence must be less than d_{max}, *i.e.* $w \leq d_{\mathrm{max}}$. Further, it is input sequences with low weight, in particular with $w = 1, 2, 3$ and 4, that

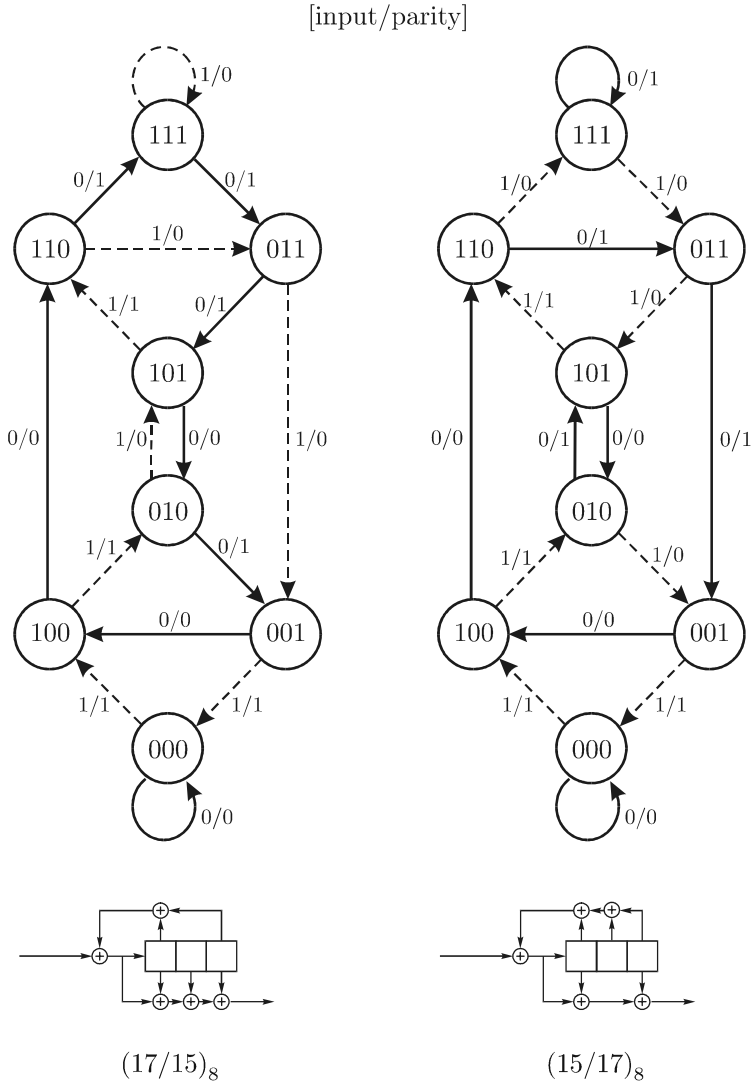


Figure 2.9: State-diagrams of recursive convolutional encoders with generator polynomials $(17/15)_8$ and $(15/17)_8$, respectively. The zero-input loops are indicated by the state transitions with solid lines.

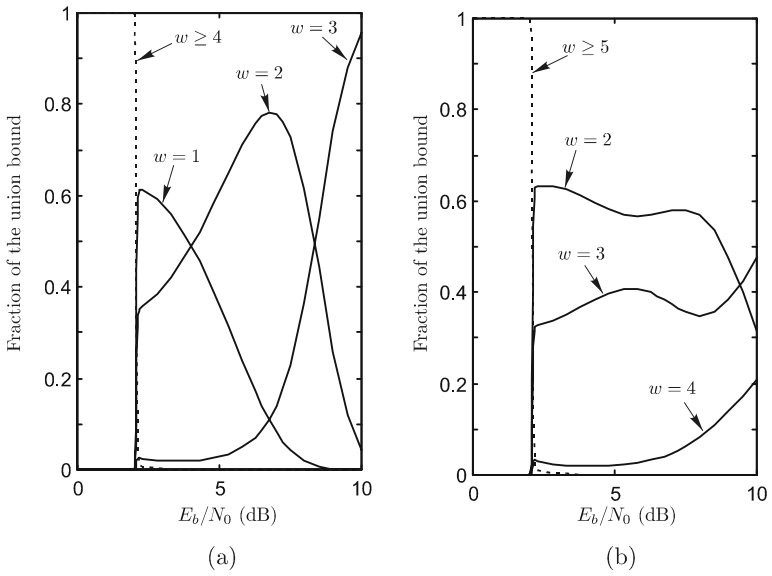


Figure 2.10: Relative contribution to the union bound for low-weight input sequences, for the ensemble of Turbo codes using 500-bit interleavers and constituent encoders with generator polynomials $(17/15)_8$. (a) No trellis termination and (b) termination of the first trellis only. The Hamming weight of the input sequence is denoted w.

are most likely to cause low-weight codewords. This is motivated in Figure 2.10, showing the relative contributions to the union bound for Turbo codes using 500-bit interleavers and constituent encoders with generator polynomials $(17/15)_8$, for various input weights w. Figure 2.10(a) shows the situation for Turbo codes using no trellis termination at all, while in Figure 2.10(b), the first constituent encoder is terminated. For both alternatives, it is the input sequences with Hamming weight less than or equal to 4 that have the major influence on the union bound, for $E_b/N_0 > 2$ dB (the union bound is unrealistic for $E_b/N_0 < 2$ dB). In the following, the mechanism behind the origin of these low-weight codewords is reviewed.

Weight-1 input sequences

To start with, we study input sequences of weight 1, *i.e.* $w = 1$. Since the constituent encoders are recursive, the single input 1 will cause them to cycle in the zero-input loops until the end of the block, as illustrated in Figure 2.11. The more time spent in these loops, the larger amount of parity 1's is generated in the parity sequences. For a low-weight codeword to be generated, it

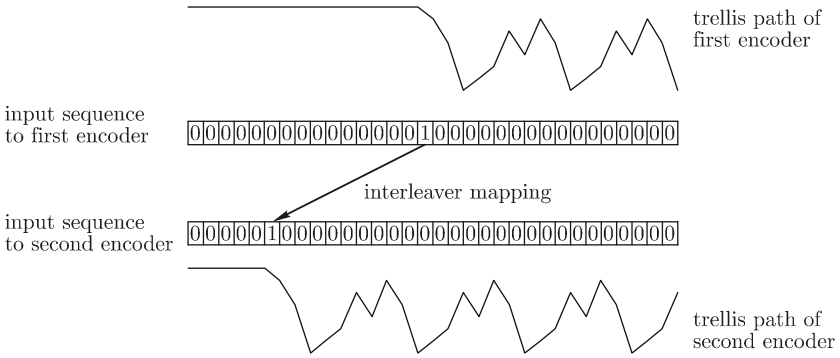


Figure 2.11: Illustration of the trellis paths caused by a weight-1 input sequence, for a specific interleaver mapping.

is necessary that both encoders spend only a limited time in their zero-input loops. Hence, low-weight codewords resulting from weight-1 input sequences are produced whenever a position near the end of the input sequence is interleaved to a position near the end of the interleaved sequence. The number of such low-weight codewords are easily found for a specific interleaver, simply by encoding all the weight-1 input sequences.

A weight-1 input sequence $U(D) = D^{i_1}$ will spend $N-i_1-1$ trellis transitions in the zero-input loop, corresponding to $\lfloor (N - i_1 - 1)/L \rfloor$ full cycles[8]. The weight of the generated parity sequence is thus lower bounded by

$$p_1|_{U(D)=D^{i_1}} \geq 1 + \left(\frac{N - i_1 - 1}{L} - 1 \right) p_{\text{loop}}, \tag{2.17}$$

since $\lfloor x \rfloor \geq x - 1$. When searching for all Turbo code codewords with weight less than or equal to $d_{\max}$, there is no point in encoding input sequences for which the weight of the first parity sequence is larger than $d_{\max}$ minus the lowest possible weight of the other two sequences, *i.e.* input sequences for which $p_1 > d_{\max} - w - \min_{w=1} p_2 = d_{\max} - 2$. Hence, it is sufficient to encode input sequences $U(D) = D^{i_1}$ with $i_1 \geq N - 1 - ((d_{\max} - 3)/p_{\text{loop}} + 1) L$.

It is common to employ trellis termination of at least the first constituent encoder in a Turbo code encoder. Then, the encoder is forced to the zero-state by the end of the input sequence by appending appropriate tail bits. Hence, the sequence that enters the encoder can no longer consist of less than two 1's and, consequently, there exists no codewords that are generated by weight-1 input sequences. Thus, for Turbo codes which employ trellis termination of at least one constituent encoder within the length of the interleaver, there are no low-weight codewords that originate from weight-1 input sequences.

[8] $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x.

Weight-2 input sequences

The recursive property of the constituent encoders is of importance also for input sequences of weight larger than 1. An input weight-2 sequence returns the encoder to the zero state only if the second input 1 is at a position when the encoder is in state $0\ldots01$, that is, in state 001 for the memory-3 encoders in Figure 2.9. Therefore, all weight-2 input sequences that return a recursive encoder to the zero state can be expressed on the form

$$U(D) = D^{k_1}\left(1 + D^{k_2 L}\right), \tag{2.18}$$

where k_1 and k_2 are integers such that $0 \leq k_1 \leq N-1$ and $0 \leq k_1 + k_2 L \leq N-1$.

Examples of weight-2 input sequences that cause low-weight parity sequences in the first constituent encoder are shown in Figure 2.12. For the particular interleaver mapping shown in 2.12(a), the input 1's are mapped to two positions that also after interleaving are separated by a multiple of L positions. Consequently, the trellis detour taken by the second encoder is also of limited length, and the overall Turbo code codeword weight is potentially low. In Figure 2.12(b), however, the interleaving of the input 1's are such that the interleaved positions are *not* separated by a multiple of L positions. As a consequence, the trellis path of the second encoder runs in the zero-input loop until the end of the block, generating a codeword with large Hamming weight (assuming a reasonably large interleaver, and $i'_1 \ll N$, where i'_1 is the position of the first 1 in the interleaved sequence).

To summarize, finding low-weight codewords caused by weight-2 input sequences consists in encoding all input sequences on the form $U(D) = D^{k_1} \cdot \left(1 + D^{k_2 L}\right)$, for which $p_1 \leq d_{\max} - w - \min_{w=2} p_2 = d_{\max} - 2 - 1$. (The minimum possible weight of the second parity sequence is $\min_{w=2} p_2 = 1$, which may arise when the two input 1's are interleaved to the two very last positions[9].) Since the weight of the first parity sequence is $p_1 = 2 + k_2 \cdot p_{\text{loop}}$, it is necessary to encode all weight-2 input sequences for which $k_2 \leq \lfloor (d_{\max} - 5)/p_{\text{loop}} \rfloor$.

If the first encoder is not terminated in the zero state, it is also necessary to investigate the weight-2 input sequences that *are not* on the form $U(D) = D^{k_1}\left(1 + D^{k_2 L}\right)$, *i.e.* those for which $(i_2 - i_1) \bmod L \neq 0$. The parity weight resulting from such inputs is at least

$$p_1|_{(i_2 - i_1) \bmod L \neq 0} \geq 1 + \left\lfloor \frac{(N - i_1 - 1)}{L} - 1 \right\rfloor p_{\text{loop}}, \tag{2.19}$$

essentially following the argumentation used to derive (2.17). Thus, for non-zeroterminating input sequences it is sufficient to encode those whose first input 1 is at a position $i_1 \geq N - 1 - ((d_{\max} - 4)/p_{\text{loop}} + 2)\, L$.

[9] Dependending on the generator polynomial, the weight of the second parity sequence is either one or two for the input sequence $U'(D) = D^{N-1} + D^N$.

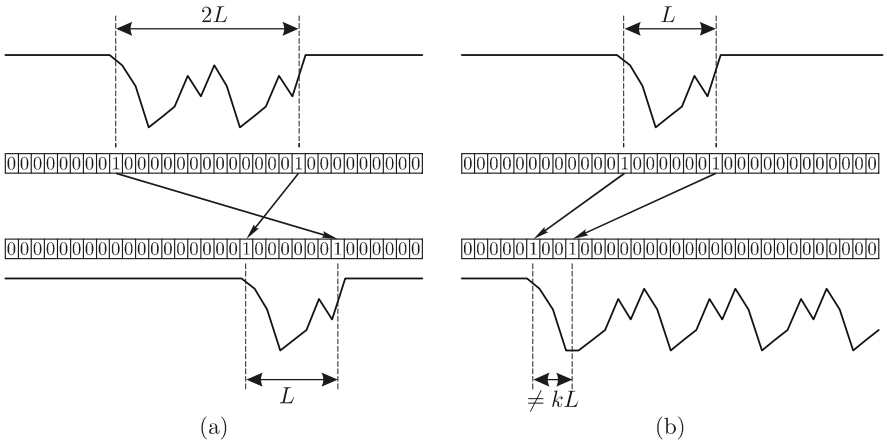


Figure 2.12: Illustration of trellis paths caused by weight-2 input sequences, for constituent encoders with $L = 7$. (a) An interleaver mapping for which both encoder inputs are zero-terminating sequences, and (b) an interleaver mapping for which the input to the second constituent encoder is not zero-terminating.

Higher-weight input sequences

Similar to the case of weight-2 input sequences, most of the low-weight codewords in a Turbo code originate from interleaver mappings where a zero-terminating input sequence is mapped to another zero-terminating sequence. When the input weight is larger than 2, the minimum length trellis detour is no longer determined by the length of the zero-input loop, L. In contrast, for weight-3 and weight-4 input sequences, the shortest possible trellis detours are instead related to the number of encoder memory elements. More precisely, the shortest possible trellis detour is lower bounded by $\nu + 1$ transitions. Since ν is typically in the range of 2 to 4 for Turbo codes, said trellis detours may be short, with accompanying low-weight parity sequences. However, when using for example random interleaving, the probability that 3 input 1's that generate a length-3 trellis detour is interleaved to another length-3 detour is small, particularly for large interleavers.

Remark. As a final remark regarding the distance spectra of Turbo codes, it is worthwhile to observe the influence of the period L. First of all, the length of the period L directly determines the shortest possible trellis detour that can be caused by weight-2 input sequences. The combination of two trellis detours that have these minimum lengths, *i.e.* $L + 1$, results in a Turbo code codeword with a potentially low weight, unless L is large. The weight of this codeword is called *the effective free distance* [24] and it proves to have an important impact on

Turbo code performances, especially when using for example random- or pseudo-random interleaving. Further, the length of the period L has an important impact on the probability that any given sequence with a certain number of 1's is zero-terminating. The last input 1 must be input at a position when the encoder is in state $0 \ldots 01$, which, in principle, occurs with probability $1/L$ for an encoder that cycles its zero-input loop. Thus, the probability that a randomly chosen interleaver mapping results in an interleaved input sequence causing a short trellis detour in the second encoder is larger the shorter the period L. All in all, a long period has a favorable influence on the distance spectra of Turbo codes.

2.4 Summary

The fundamental principles behind Turbo coding have been introduced, including the encoder structure and the principles of iterative decoding. The central components of a Turbo code encoder are the *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.

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 sequences and *a priori* information. An optimal algorithm for computing APPs is the BCJR-algorithm, also called the MAP-algorithm. Due to the high complexity of the BCJR-algorithm, soft-output versions of the Viterbi algorithm have been suggested (SOVA) for Turbo decoding. The SOVA algorithm have essentially half the implementation complexity compared to low-complexity implementations of the BCJR-algorithm, and approximately 0.5 dB worse performance when used as a constituent decoder in an iterative decoder.

The mechanism that lies behind the lower part of Turbo code distance spectra was discussed. To conclude, the lower part of Turbo code distance spectra originates from low-weight input sequences, typically of weight-2, 3 and 4, for Turbo codes using trellis termination of at least one constituent encoder. Further, the period L of the feedback polynomial has a large impact on the distance spectra. The longer the period, the better the distance spectrum. For this reason, from a distance spectrum point of view, it is attractive to use primitive feedback polynomials, since they maximize L for a given number of memory elements.

Chapter 3

Interleaver Design

The interleaver is a key component of Turbo codes, and its design is essential for achieving high performance. Indeed, the issue of designing Turbo code interleavers has received substantial attention among Turbo code researchers. Since the original introduction of the Turbo coding concept, a large amount of interleaver design criteria and algorithms have been presented, for example in [1, 4, 5, 21, 22, 27, 29, 38, 47, 50, 53, 70, 77, 80, 86, 91]

In this chapter, two conceptually different interleaver design criteria presented in [Paper I] and [Paper IV] are reviewed. Based on these criteria, an interleaver design algorithm is formed. This design algorithm has a rather low computational complexity, allowing efficient design of both small and large interleavers. Interleavers designed with this algorithm yield state-of-the-art performing Turbo codes, to our knowledge not surpassed by any other interleaver design methodology.

3.1 Design Criteria

In line with the traditional design criterion for error correcting codes, a successful interleaver design must ensure that the distance spectrum of the resulting Turbo code is "good". There is no straightforward answer to what a "good" distance spectrum is; it depends on the target operating scenario of the system. For example, at high SNRs it is well-known that the code minimum distance has a major influence on the code performance [71]. Thus, for such systems it is possible to design codes based solely upon the minimum distance parameter. In contrast, at low SNRs a larger part of the distance spectrum comes into role, and the design criteria may require proper adjustments.

The distance spectrum is a good design-base for codes that are decoded using a maximum likelihood (ML) decoding algorithm. However, the iterative decoding procedure used for Turbo codes does not guarantee ML-decoding decisions. In fact, it is easy to verify that a Turbo code decoder occasionally makes non-

ML decoding decisions. Thus, it is natural to study the properties of a Turbo code that govern the performance of iterative decoding. In particular, the choice of interleaver has a significant influence on the iterative decoding performance; hence, it is natural to formulate an interleaver design criterion based on these observations.

In the following sections, two interleaver design criteria are reviewed. The first criterion targets code properties, in terms of the distance spectrum, while the second criterion targets the performance of iterative decoding. The latter is described in more detail, since this criterion has received less attention in the literature.

3.1.1 Criterion Based on the Code Distance Spectrum

In the previous chapter, the influence of the interleaver on the distance spectrum of a Turbo code was discussed. In particular, it was demonstrated how specific interleaver mappings result in low-weight codewords. A natural interleaver design approach is to avoid precisely such mappings, thereby improving the distance spectrum. This approach has been dominating in the Turbo Code interleaver design literature, with results reported in for example [1, 21, 22, 27, 77], as well as in [Paper I].

The interleaver design algorithm presented in [Paper I] is based on avoiding interleaver mappings that generate low-weight codewords. In line with the observation in the previous chapter, that it is primarily low-weight input sequences that produce low-weight codewords, the algorithm was designed to take input sequences of weight-2, 3 and 4 into account. Since the first encoder is assumed to be terminated, weight-1 input sequences need not be considered. The algorithm was evaluated for Turbo codes using 105-bit[1] interleavers and constituent encoders with generator polynomials $(17/15)_8$. For this setup, the algorithm produced a Turbo code with minimum distance 22 and multiplicity 8. Among the interleavers compared to, the best interleaver found was a *block helical simile* [5] interleaver, resulting in a Turbo code with minimum distance 19, multiplicity 24. The distance spectra of these codes are listed in Table 1 in [Paper I], for Hamming distances up to 29 resulting from input sequences of weight 6 and less. The distance spectrum of our interleaver is clearly superior compared to the other spectra.

Even though the design criterion in [Paper I] is limited to weight-2, 3 and 4 input sequences, the computational complexity of such algorithms increases rapidly with the interleaver length. Thus, the algorithm is suitable primarily for short interleavers, up to a few hundred bits. For a given target minimum distance the design complexity, in terms of the number of interleaver mappings that must be avoided, is decreased if the feedback period L is increased. Thus,

[1] The size of 105 bits was chosen since the best interleaver found to compare with, about 100 bits long, was a 105-bit block helical simile [5] interleaver.

larger interleavers can be designed if for example constituent encoders with many memory elements and primitive feedback polynomials are used.

Other distance spectrum criteria/algorithms

A genetic algorithm adapted for interleaver design based on the distance spectrum criterion was proposed by Bartolome in [6]. Using this algorithm, Bartolome generated a 105-bit interleaver yielding a code with minimum distance 23 and multiplicity 29. Thus, the minimum distance was slightly improved from 22, achieved using the algorithm in [Paper I].

An interesting observation made when investigating the distance spectra of Turbo codes concerns the characteristics of the input sequences that generate low-weight codewords; these input sequences often consist of a short burst of 1's, at positions such that the encoder is returned to the zero-state by the last 1 (*cf.* Section 2.3). Hence, an efficient way of avoiding interleaver mappings that generate low-weight codewords is to ensure that nearby positions *are not* interleaved to positions again being close to each other, after interleaving. A design criterion based on this observation was proposed by Dolinar *et al.* in [27]. The proposed design criterion ensures that two input positions that are separated by S or less positions are interleaved to two positions at least S positions apart. Besides that, this *S-random design* is the same as when "designing" a pseudo-random interleaver. Interleavers designed with this *spreading* criterion are known to yield good-performing Turbo codes. However, trying to design small S-random interleavers, in the range of 100 bits, results in minimum distances clearly inferior to those reported in [Paper I] and [6].

3.1.2 Criterion Based on the Performance of Iterative Decoding

In the derivation of the BCJR algorithm, several assumptions regarding independencies between stochastic variables are made. For example, consider the derivation of the forward and backward recursions used for calculating the α- and the β-values, that is, equations (2.9) and (2.10). In the derivation of the forward recursion, independence is assumed between decoder input R_n, as defined in (2.2), and all the previous inputs R_1^{n-1}, conditioned on knowing the trellis state at position $n-1$. The independence assumption is manifested by the use of

$$p\left(u_n = i, S_n = s, R_n | S_{n-1} = s', R_1^{n-1}\right) = p\left(u_n = i, S_n = s, R_n | S_{n-1} = s'\right) \tag{3.1}$$

in Appendix A, expression (A.13). Similarly, the backward recursion assumes R_{n+1}^N to be independent of R_n, conditioned on knowing the trellis state at position n, through the use of

$$p\left(R_{n+1}^N | u_n = i, S_n = s, R_n, S_{n-1} = s'\right) = p\left(R_{n+1}^N | S_n = s\right) \tag{3.2}$$

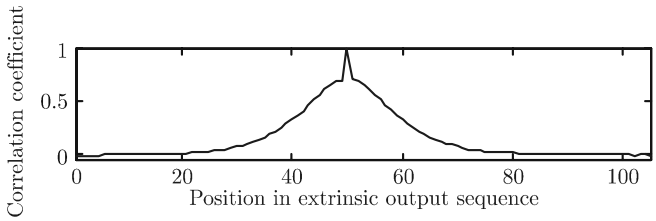


Figure 3.1: Empirical correlation bewteen extrinsic output 50 and the entire sequence of extrinsic outputs (105 bits), for a recursive convolutional encoder with generator polynomials $(17/15)_8$.

in expression (A.14). In an iterative decoder, the input *a priori* information $\tilde{\Lambda}_n$ is approximated by the extrinsic output from the previous decoder, $Le_{n'}$. Thus, the decoder input is $R_1^N = \left(x_1^N, y_1^N, Le_1^N\right)$, and the above assumptions imply that each extrinsic output $Le_{n'}$ is independent of all the other extrinsic outputs. However, the extrinsic outputs are in contrast likely to be statistically dependent on each other, since they are functions of the same input variables. That this is indeed the case can be verified by examining the correlation between the extrinsic outputs, since correlated variables implies a statistical dependency. Figure 3.1 shows empirical correlation coefficients[2] between extrinsic output 50 and the entire sequence of extrinsic outputs (105 bits long), when decoding a received sequence with the BCJR-algorithm. The encoder in this example has generator polynomials $\begin{pmatrix} 1 & 17/15 \end{pmatrix}_8$, and the received codewords are corrupted by additive white Gaussian noise. Evidently, extrinsic output 50 is indeed correlated to its neighboring extrinsic outputs, with decreasing correlation as the separation is increased. Consequently, there exists a dependency between extrinsic outputs, which is translated into a dependency between the *a priori* inputs in the next decoding step. Thus, the assumptions made in the derivation of the forward and backward recursions in the BCJR algorithm are violated. The amount of correlation between two extrinsic outputs $Le_i^{(1)}$ and $Le_j^{(1)}$ is larger the smaller the distance $|i-j|$. Thus, it is desirable to use two extrinsic outputs with small separation $|i-j|$ as inputs as far away from each other as possible in the next decoding step.

Another independence assumption exploited in the BCJR algorithm is when calculating the γ-values. Behind the derivation of (2.8), independence is assumed between the encoder inputs x_n, y_n and $Le_{n'}$, conditioned on knowing the state transition from position $n-1$ to n, when separating the probability

[2] When transmitting random information sequences, the correlation between extrinsic outputs measured in terms of $Cov\left(Le_i, Le_j\right)$ is zero, since the information symbols u_i and u_j are uncorrelated. The correlation coefficients reported here are obtained by 'rectifying' each extrinsic value by its information symbol, *i.e.* via $Cov\left(Le_i \cdot (2u_i - 1), Le_j \cdot (2u_j - 1)\right)$. The same holds for correlation coefficients between extrinsic outputs and noise samples, etc.

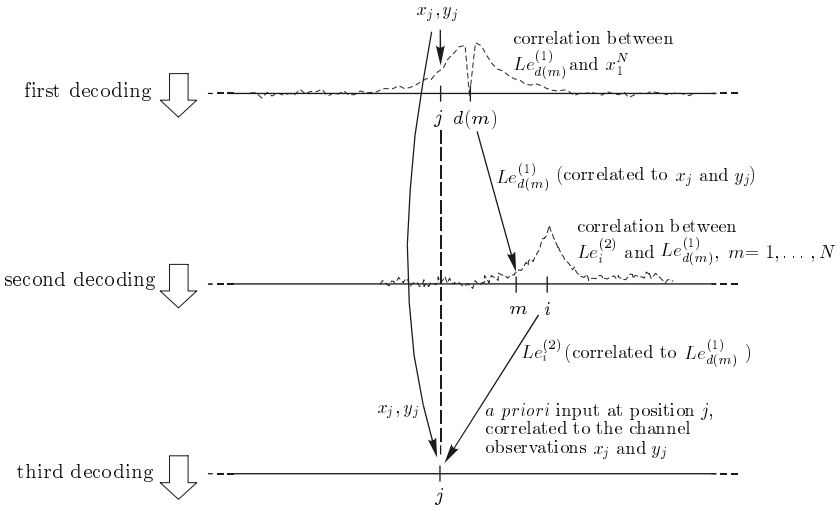


Figure 3.2: Illustration of the origin of the dependency between the *a priori* input j in the third decoding step and the corresponding channel observations, *i.e.* x_j and y_j. In this illustration, $d(i) = j$, or $d^{-1}(j) = i$.

density function

$$p(x\ ,y\ ,Le\ '|u\ =i,S\ =s,S\ _{-1}=s') = p(x\ |u\ =i,S\ =s,S\ _{-1}=s')\cdot p(y\ |u\ =i,S\ =s,S\ _{-1}=s')\cdot p(Le\ '|u\ =i,S\ =s,S\ _{-1}=s') \tag{3.3}$$

in Appendix A, in the derivation of (A.16). For the first and second decoding step this assumption is valid; however, starting with the third decoding step, there exists a dependency. The origin of this dependency is illustrated in Figure 3.2. In the first decoding step, no *a priori* information is used, and thus there exist of course no dependencies. As for the second decoding, *a priori* input $Le^{(1)}_{d(m)}$ is independent of the parity observation $y^{(2)}, \forall n$, since the second parity sequence is not used at all by the first decoder. As for the independence between $Le^{(1)}_{d(m)}$ and $x_{d(m)}$, where $x_{d(m)}$ is the systematic channel observation interleaved to position m, the independence follows from the calculation of the extrinsic output in (2.14). Indeed, the subtraction of the intrinsic information $Li_{d(m)}$ from the soft-output $\Lambda^{(1)}_{d(m)}$ provides independence between the extrinsic information $Le^{(1)}_{d(m)}$ and $x_{d(m)}$. However, in the third decoding step the process of iterative decoding is asserted. The *a priori* input at position $j = d(i)$, $Le^{(2)}_i$, stemming

from output position i from the second decoder (*cf.* Figure 3.2), is dependent on the previous *a priori* input at position m, *i.e.* $Le^{(1)}_{d(m)}$. This dependency is demonstrated by the dashed curve at the second decoding stage in Figure 3.2, showing empirical correlation between *a priori* input $Le^{(1)}_{d(m)}$ and extrinsic output $Le^{(2)}_{i}$. In turn, the *a priori* input $Le^{(1)}_{d(m)}$ is dependent on the channel observations x_j and y_j, again illustrated by a set of empirically found correlation coefficients. In total, through this decoding chain it is plausible that the *a priori* input $Le^{(2)}_{i}$, which is input at position j in the third half-iteration, is dependent on the channel observations x_j and y_j. That this is indeed the case is easy to verify by Monte-Carlo simulations. As a consequence, the assumptions used in (3.3) are violated.

In the above paragraphs, we have seen three assumptions made in the derivation of the BCJR algorithm that are not valid when used in an iterative decoding scheme. For all violations, the degree of correlation is reduced with increasing distances between positions in the encoder input and output sequences. Referring to the notations in Figure 3.2, this means that the sum of the distances $|d(m) - d(i)|$ and $|m - i|$, *i.e.* the length of the cycle formed by interleaver mappings $d(m)$ and $d(i)$, should be made large.

Naturally, the size of the interleaver sets a limit on how large $|d(m) - d(i)| + |m - i|$ can be. More importantly, it imposes restrictions on how large the smallest value of $|d(m) - d(i)| + |m - i|, \forall i, m$ can be. Conceptually, it is plausible that the spectrum of distances $|d(m) - d(i)| + |m - i|, \forall i, m$ has an influence on the performance of the iterative decoding. In [Paper III], a measure that comprises all these distances into a single scalar is investigated. This measure is called *iterative decoding suitability* (IDS), and it proves to be related to the convergence rate of iterative decoding.

In [Paper IV], an interleaver design criterion based on the performance of iterative decoding is proposed. In principle, the criterion is formulated with the goal to reduce the degree of suboptimality of the iterative decoding algorithm, as a consequence of the aforementioned assumptions. Both the design criterion and the IDS measure use an approximation of the correlation between extrinsic outputs after the second decoder and channel noise samples. This approximation takes the form of an exponential decrease in correlation as the separation between two positions is increased. To start with, the correlation between extrinsic output i from the first decoder and systematic noise sample j, denoted $\rho^{(1)}_{i,j}$, is approximated as

$$\hat{\rho}^{(1)}_{i,j} = \begin{cases} ae^{-c|j-i|} & \text{if } j \neq i \\ 0 & \text{otherwise} \end{cases} \qquad i, j = 1, 2, \ldots, N. \tag{3.4}$$

The value of the constants a and c depend on the choice of constituent encoders; for example, for the generator polynomial $(17/15)_8$, $a = 0.23$ and $c = 0.18$. As for the corresponding correlation after the second decoding stage, denoted $\rho^{(2)}_{i,j}$,

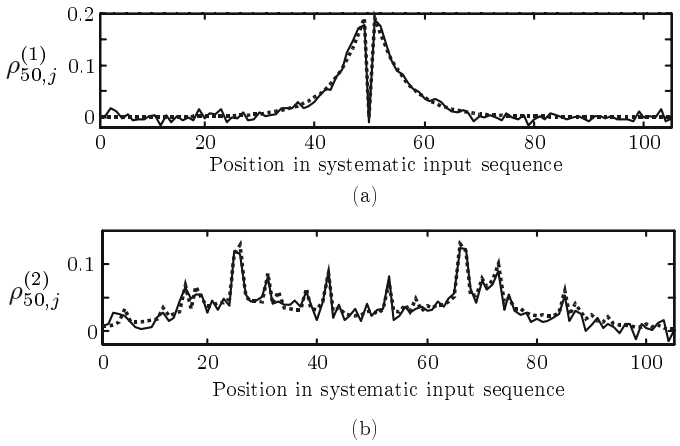


Figure 3.3: Illustration of the correlation between extrinsic outputs and channel observations, after the first (a) and second decoding (b) stages. The solid lines are empirically found coefficients, while the dotted lines correspond to the approximations (3.4) and (3.5).

these coefficients are approximated as [Paper III]

$$\hat{\rho}^{(2)}_{i,j} = \frac{1}{2}\underbrace{\left(1-\delta_{d^{-1}(j)-i}\right)ae^{-c|d^{-1}(j)-i|}}_{\text{from the systematic input}} + \frac{1}{2}\underbrace{\sum_{\substack{m=1\\ m\neq i, d(m)\neq j}} a^2 e^{-c(|d(m)-j|+|i-m|)}}_{\text{from extrinsic inputs}}, \tag{3.5}$$

where δ is the Kronecker delta-function, and $d^{-1}(j)$ is the position to which input j is interleaved. Both approximated and empirically achieved correlation coefficients for a 105-bit Turbo code, after the first and second decoding stages, are shown in Figure 3.3. Even though (3.5) is a heuristic approximation of the correlation coefficients after the second decoder, it yields reasonable results and it proves to be very useful for the purpose of interleaver design.

With the above used interleaver description, $d(i)$ represents the position to which the ith output from the second decoder is *deinterleaved*[3]. In Figure 3.2, extrinsic output i from the second decoder is deinterleaved to position j, *i.e.* $d(i) = j$. The process of designing an interleaver can be organized as choosing the positions $d(i), i \in \{1, \ldots, N\}$. For each choice, $d(i)$ is chosen to minimize the correlation between extrinsic output $Le^{(2)}_i$ (which is deinterleaved to be used

[3] Equivalently, $d(i)$ represents the position in the original sequence that is interleaved to become the ith input to the second decoder.

as *a priori* input at position $d(i)$) and the channel observations $x_{d(i)}$ and $y_{d(i)}$. Thus, an appealing design criterion is to choose $d(i)$ as

$$d(i) = \arg\min_j \hat{\rho}_{i,j}^{(2)}, \tag{3.6}$$

where $\hat{\rho}_{i,j}^{(2)}$ is calculated using the already defined interleaver mappings. Note that for $j = d(i)$ the first part of (3.5), corresponding to the systematic input, is zero since $1 - \delta_{d^{-1}(j)-i} = 0$. Consequently, the above minimization is equivalent to

$$d(i) = \arg\min_j \sum e^{-c(|d(m)-j|+|i-m|)}. \tag{3.7}$$

This means that $d(i)$ is assigned a position j so that the sum in (3.7) is minimized. The longer the cycles formed by the mappings $d(m)$ and $d(i)$, *i.e.* the larger the $|d(m) - d(i)| + |i - m|$, the smaller the sum. This coincides with the observations by Wiberg in [97], where he analyses iterative decoding using Tanner graphs. Wiberg finds indications that to avoid poor decoding performance, short cycles should be avoided.

In the next section, we describe an interleaver design algorithm that exploits both the correlation criterion and the distance spectrum criterion.

3.2 A New Interleaver Design Algorithm

In the two previous sections, we have reviewed two conceptually different interleaver design criteria; one that targets code properties, and one that targets the performance of iterative decoding. In this section, we combine these two criteria into an interleaver design algorithm.

As before, the interleaver is described by a vector of length N, defined by the elements $d(m), m = 1, \ldots, N$. In the algorithm described here, the elements in this vector are assigned values one by one, until the whole vector is defined. For each new assignment, a position is chosen based on both the distance spectrum and the correlation criteria. Regarding the distance spectrum, the algorithm is formulated to avoid interleaver mappings that result in a codeword with Hamming weight less or equal to a certain value, called the 'design distance', d_{design}.

It is not straightforward how to weigh the importance of the distance spectrum of the code and the performance of the iterative decoding. Especially since it is not obvious how much the iterative decoding performance is degraded by a certain amount of correlation between the decoder inputs. In the design algorithm presented here, we have adopted the following strategy:

For each new interleaver element to assign, choose a position with "good" correlation properties among the positions that fulfill the distance spectrum requirements imposed by d_{design}.

In principle, the elements in the interleaver vector can be assigned values in any order; however, in our investigations we have found it efficient to assign the values in straight order, either from position 1 to N, or from N to 1. For reasons described in [Paper VI], which deals with interleaver design from a trellis termination point of view, there are advantages with designing interleavers in reversed order, that is, from N to 1. This is the strategy adopted in the algorithm presented here.

Below, the basic stages of the interleaver design algorithm are stated. Various aspects of the algorithm are commented in the sections to follow.

Definitions:

$\boldsymbol{d}$: The vector that defines the interleaver. $\boldsymbol{d} = \left[\begin{array}{cccc} d(1) & d(2) & \ldots & d(N) \end{array} \right]$, where the mth element holds the input position, $d(m)$, that is interleaved to the mth position in the interleaved sequence.

$d_{\mathbf{design}}$: The maximum codeword Hamming weight that is consciously avoided by the interleaver design algorithm. This value may be compared with the $d_{\max}$ parameter used in Section 2.3, when searching for codewords with weights less than or equal to $d_{\max}$.

$\mathcal{M}$: The set of positions in the interleaved sequence not yet associated with a position in the original sequence, *i.e.* the positions i for which $d(i)$ is not yet defined.

$\mathcal{L}$: The set of positions in the original sequence that has not been assigned to interleaved positions, *i.e.* the counterpart of $\mathcal{M}$.

H: A vector of length N containing the Hamming weights of the lowest-weight codewords that result from the possible choices for $d(i)$. In particular, $H(l)$ stores the weight of the lowest weight codeword that results from the choice $d(i) = l, l \in \mathcal{L}$.

Algorithm:

Initialization

1. Set $\mathcal{M} = \{1, \ldots, N\}$ and $\mathcal{L} = \{1, \ldots, N\}$.
2. Choose a design distance d_{design}.

Design loop

1. Choose a position $i \in \mathcal{M}$, for which $d(i)$ is to be assigned. With reversed order interleaver design, $i = \max \mathcal{M}$. $d(i)$ is to be chosen among the positions $l \in \mathcal{L}$.

2. Calculate the lowest codeword weight resulting from each of the choices $d(i) = l, l \in \mathcal{L}$, taking Hamming weights less than or equal to d_{design} into account. Thus, for each position $l \in \mathcal{L}$ there is an associated minimum codeword weight that will exist in the code, given that the choice $d(i) = l$ is made. This set of weights is stored in the vector H. If no low-weight codeword ($\leq d_{\text{design}}$) is found for the choice $d(i) = l$, then $H(l)$ is assigned a large number (arbitrary, but larger than d_{design} and equal for all such l's).

3. Let the positions in $\mathcal{L}$ that result in the highest weight codewords form the set $\mathcal{L}'$, *i.e.*

$$\mathcal{L}' = \mathcal{L} \cap \left\{ l : H(l) = \max_m H(m) \right\}. \tag{3.8}$$

4. Choose to deinterleave extrinsic output $Le_i^{(2)}$ to the position among the positions in $\mathcal{L}'$ it has the lowest correlation to, according to (3.7). Thus, choose $d(i)$ as

$$d(i) = \arg\min_{l \in \mathcal{L}'} \sum_{m \notin \mathcal{M}} e^{-c(|d(m)-l|+|i-m|)}. \tag{3.9}$$

   If more than one of the positions $l \in \mathcal{L}'$ result in the same minimum correlation, choose one of them at random.

5. Remove i and $d(i)$ from $\mathcal{M}$ and $\mathcal{L}$, respectively.

6. If $\mathcal{M}$ is not empty, go to 1.

The design process is illustrated in the flow chart in Figure 3.4. Many modifications of the algorithm are of course possible; a few of which are discussed in the following.

Optimization criterion. The process of choosing the position $d(i) = l, l \in \mathcal{L}'$, corresponding to the channel observations that extrinsic output i is least correlated to, up to the point of the present design step, by no means guarantees that a global optimum of the overall minimization problem is found. In fact, a global optimization criterion has not even been stated. Such a criterion could for example be based on the IDS-measure, hereby including the correlation properties for the entire interleaver. However, experience from designing a large amount of interleavers indicates that the presented algorithm produces interleavers with very good correlation properties, compared to what is believed to be achievable. This is motivated in Appendix B.1, where the correlation properties of interleavers designed with the presented algorithm are compared to whose of *golden interleavers* [21]. Golden interleavers are based on the golden section, and they result in interleavers with very good correlation properties.

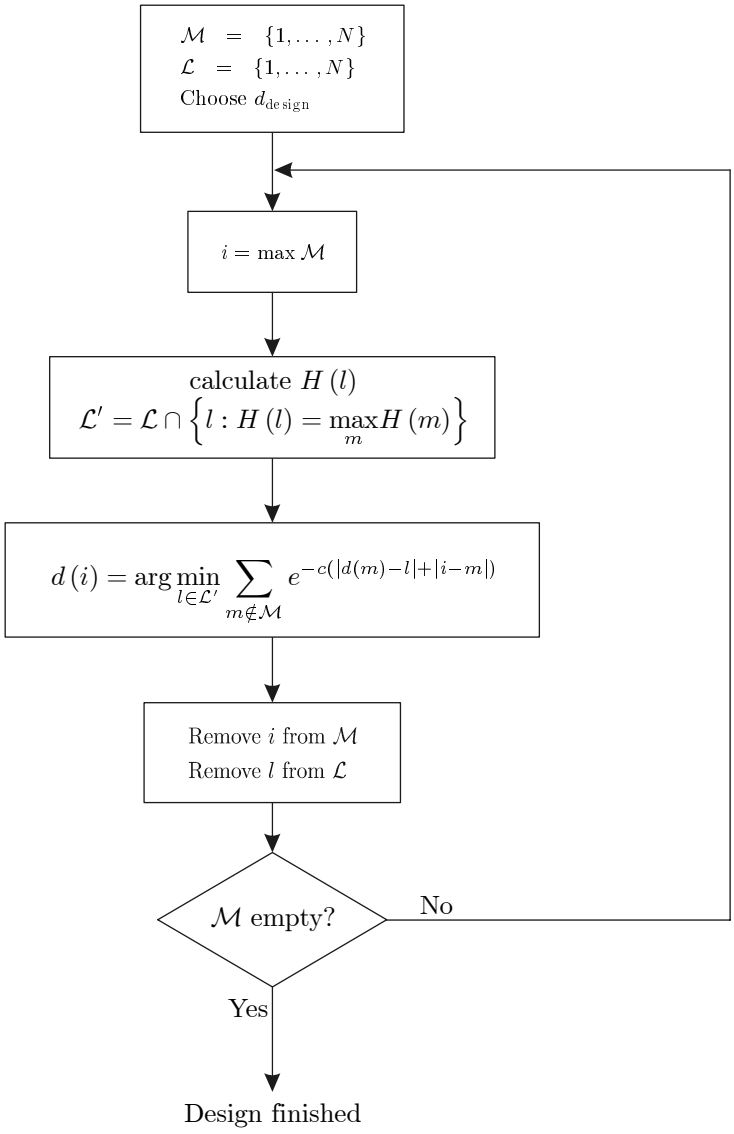


Figure 3.4: Flow chart of the interleaver design algorithm.

The motivation for the use of golden interleavers in [21] is however not based on the performance of iterative decoding, but rather on its successful spreading of error bursts. Golden interleavers perform very well for small interleaver sizes, but suffer from high multiplicity of rather low-weight codewords for large interleaver sizes. Consequently, the error-floor for golden interleavers appear earlier than what is achievable by including a distance spectrum criterion in the design.

Design order. It is not obvious how to choose the design order, that is, the order in which the elements are chosen from the set $\mathcal{M}$ in step 1 of the design loop. The most suitable design order is related to the size of the interleaver, as illustrated in Appendix B.2. The use of straight-order design (*e.g.* from last to first) yields good results for a wide range of interleaver sizes. However, for very small interleavers, in the order of 100 bits, there might be a reason to investigate alternative strategies, for example random design order.

Choosing d_{design}. It is not straightforward to choose a suitable value of d_{design}. Naturally, it is tempting to use a large d_{design} so that the minimum distance of the code becomes as large as possible. However, if d_{design} is chosen much larger than the minimum distance the design algorithm can achieve, it is likely that the distance restrictions have had a negative influence on the correlation properties of the designed interleaver. Therefore, it is often necessary to choose d_{design} based on experience. This process is discussed in Appendix B.4, where the effect of choosing different values of d_{design} is illustrated.

Self-terminating interleavers. The design algorithm can easily be modified so that the designed interleaver is a self-terminating interleaver [5, 38]. Self-terminating interleavers have the advantage that both constituent encoders are terminated in the same state. Consequently, the issue of trellis termination is readily solved by inserting flush bits so that the first encoder is terminated in the zero-state. However, experience show that the self-termination constraint may severely restrict the interleaver design. Results on this issue are reported in [Paper VI].

Interleaver structures. It is possible to include arbitrary design restrictions in the presented algorithm. A reason for doing so can be implementation aspects. For example, by restricting the design to *odd-even symmetric* interleavers [Paper V], that is, interleavers for which $d(i)$ is odd if i is even and $d(i) = j \Longrightarrow d(j) = i$, the storage requirement of the resulting interleaver rules are less than that of non-structured interleavers. These aspects, including additional storage-saving interleaver structures, are studied in [Paper V].

Distance calculations. The most demanding part of the design process, in terms of computational complexity, is Step 2 in the design loop, *i.e.* the distance

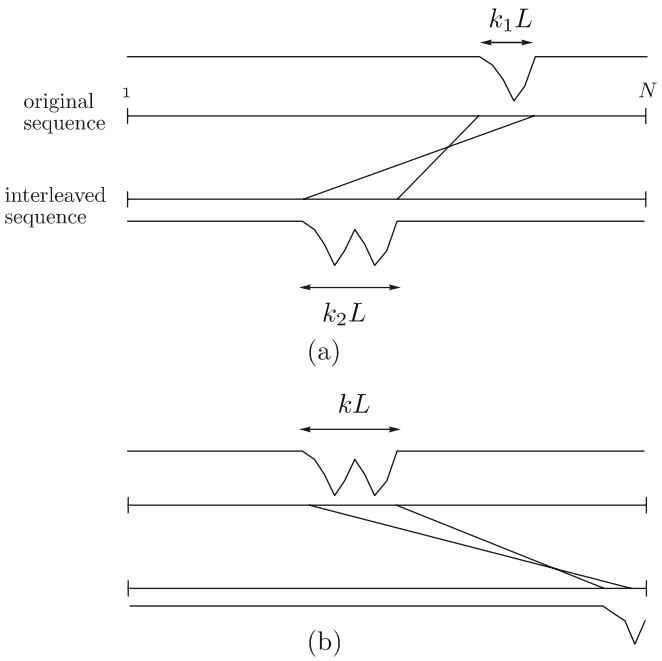


Figure 3.5: Typical weight-2 input sequence that are considered when searching for positions that result in a codeword with weight less than or equal to d_{design}. Valid for design of interleavers used in a Turbo code where the trellis of the first constituent encoder is terminated.

related calculation of H. Fortunately, the correlation criterion has a positive influence on the distance spectrum, especially for higher-weight input sequences. As a consequence, the number of checks that need to be performed in the design is significantly reduced. For example, most zero-terminating weight-3 and weight-4 input sequences are effectively broken up by the correlation criterion. However, there are certain mappings that spreading criteria[4] in general fail to avoid. Figures 3.5–3.7 show the type of weight-2, 3 and 4 input sequences that are not always avoided. Thus, it is the Hamming distances resulting from these types of sequences that are accounted for by the H-vector. The sequences shown in Figures 3.5–3.7 are valid when designing an interleaver for a Turbo code whose first constituent encoder is terminated in the zero-state. If another termination method is used, other types of sequences need to be considered. Specific interleaver design for different trellis termination methods is addressed in [Paper VI]. One of the results thereof is that as long as the proper types of

[4] Both the correlation design and the S-random design are examples of interleaver design criteria that yield good spreading properties.

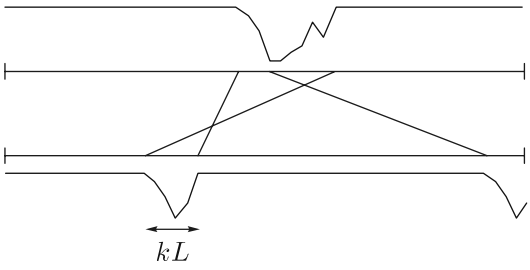
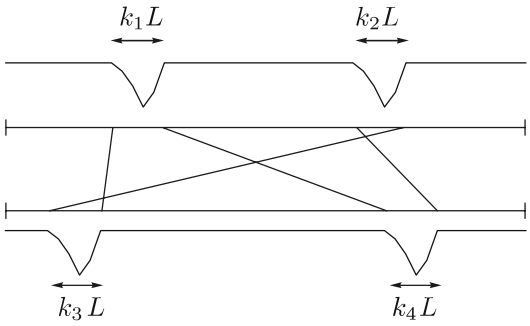
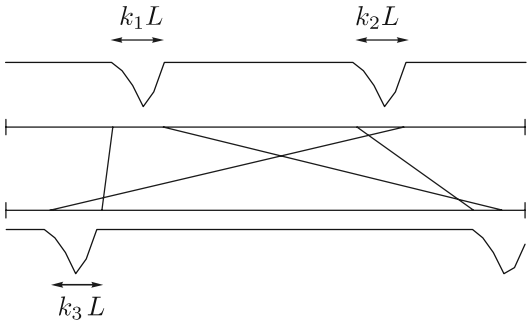


Figure 3.6: Typical weight-3 input sequence that are considered when searching for positions that result in a codeword with weight less than or equal to d_{design}. Valid for design of interleavers used in a Turbo code where the trellis of the first constituent encoder is terminated.



(a)



(b)

Figure 3.7: Typical weight-4 input sequence that are considered when searching for positions that result in a codeword with weight less than or equal to d_{design}. Valid for design of interleavers used in a Turbo code where the trellis of the first constituent encoder is terminated.

input sequences are considered in the interleaver design, the performances of different trellis termination methods are very similar, also for low error rates.

3.3 Performance Examples

This section demonstrates how the presented interleaver design criteria influence the error rate performances of Turbo codes. For this purpose, four different 105-bit interleaver constructions are compared:

1. Random interleaving. A different pseudo-random interleaver is chosen for each block to transmit.

2. A block helical simile interleaver [5], with 5 rows and 21 columns. $d_{\min} = 19$, $a_{d_{\min}} = 24$.

3. A distance spectrum-designed interleaver (from [Paper I]). $d_{\min} = 22$, $a_{d_{\min}} = 8$.

4. A distance spectrum- *and* correlation-designed interleaver. Designed with the algorithm presented in the previous section. $d_{\min} = 18$, $a_{d_{\min}} = 1$.

All compared interleavers are 105 bits long, with permutation matrices[5] and IDS-values shown in Figure 3.8. Both the block helical simile interleaver (#2) and the correlation-designed (#4) interleaver have good correlation properties, judging from the IDS-values and from inspecting their permutation matrices[6]. Similarly, both the random interleaver (#1) and the distance spectrum-designed interleaver (#3) appear to have poor correlation properties.

The frame error rates as a function of the number of decoding iterations for the resulting Turbo codes are shown in Figure 3.9 for SNRs corresponding to $E_b/N_0 = 2.0$ and 3.5 dB. The constituent encoders have generator polynomials $(17/15)_8$, and the decoders use the full BCJR-algorithm. It is observed that the interleavers with the best correlation properties (lowest IDS) yield faster converge than the other interleavers, both at medium and high SNRs. As a consequence, for a Turbo decoder employing less than 12 half-iterations (or 6 full iterations), the best performance is obtained with the interleavers having superior correlation properties. At $E_b/N_0 = 3.5$ dB, the distance spectrum designed-interleaver (#3) performs better than the block helical simile interleaver (#2) after about 7 decoding iterations. Before that, the superior convergence rate of the block helical simile interleaver is of higher importance.

[5] For the random interleaving case, the permutation matrix shown corresponds to one single pseudo-random interleaver, while the IDS-value is the mean IDS of 500 pseudo-random interleavers. The standard deviation from the mean value is 0.03.

[6] In the permutation matrices, two dots that are close to each other correspond to two input postitions that are close to each other both before and after interleaving. Thus, they result in high correlation between the decoder inputs.

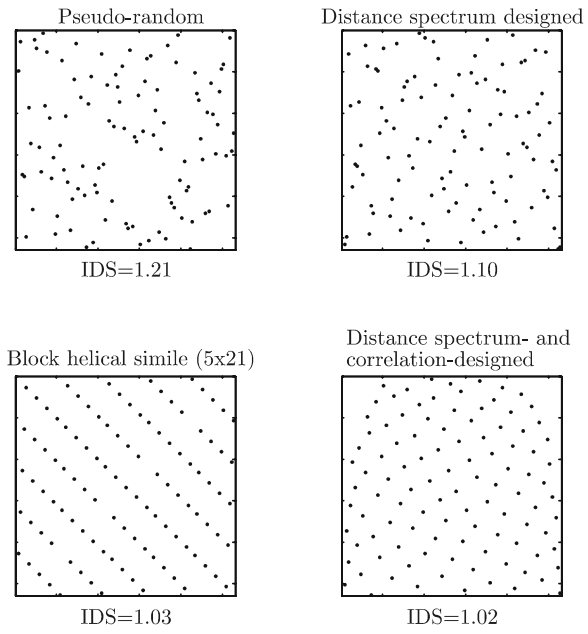


Figure 3.8: Permutation matrices and IDS-values for four compared 105-bit interleavers. The IDS-values are normalized to the IDS of a golden interleaver of the same size.

3.4 Summary

When designing interleavers for Turbo codes, there are two important concepts to take into account: (1) the distance properties of the code and (2) its suitability two be iteratively decoded. The latter concept is a new code design criterion, required as a consequence of using sub-optimum iterative decoding. The interleaver design criterion related to the code distance spectrum is based upon avoiding interleaver mappings of self-terminating sequences into another self-terminating sequence. The design criterion related to the decoding performance is instead based on the correlation properties of the inputs to the constituent decoders. Fortunately, the two criteria are not in conflict with each other. On the contrary, the spreading achieved when designing interleavers with good correlation properties has positive influence on the distance properties of the code.

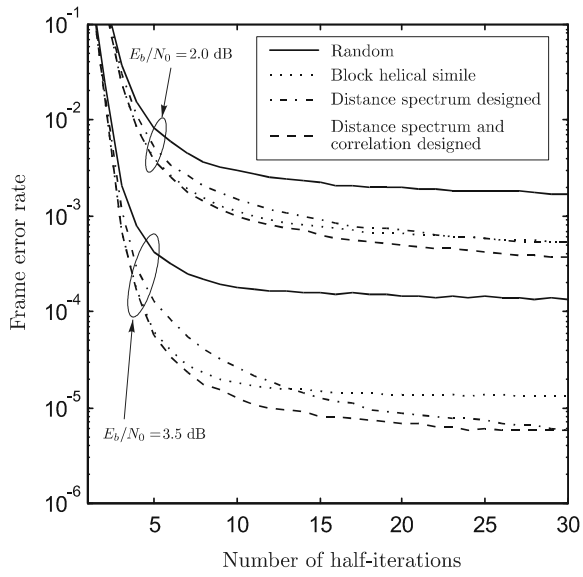


Figure 3.9: Simulated frame error rate performances vs. the number of decoding iterations for Turbo codes using four different 105-bit interleavers. Tranmission over an AWGN channel with $E_b/N_0 = 2.0$ and 3.5 dB.

Chapter 4

Choosing Constituent Codes

In the previous chapter, in the review of interleaver design criteria, two fundamental properties that govern the performance of a Turbo code were exploited: the distance spectrum and the performance of iterative decoding. In this chapter, the same approach is used to investigate selection criteria for the choice of constituent codes, with focus on the feedback polynomial. This review is not exhaustive, but indicative of properties that should be considered in the process of designing Turbo codes. In contrast to the conclusions regarding interleavers, which can be designed both for good Hamming distance properties and good correlation properties, there is a trade-off between the two when choosing the constituent encoders. Therefore, the guidelines for choosing constituent encoders are dependent on the target operating point of the code.

The issue of choosing generator polynomials from a distance spectrum perspective only, including both the feedback and parity polynomials, is investigated in for example [8, 25, 44]. The issue of choosing encoders from a decoding perspective is investigated in [90, 16], on which the results presented in this chapter are based. The main extensions to previous observations are due to the use of interleavers designed with the criteria presented in Chapter 3, which influences the regions for which different design guidelines apply.

All simulation results reported in this chapter are valid for 15 decoding iterations using the full BCJR-algorithm. Further, the results are obtained for Turbo codes using trellis termination of the first constituent encoder.

4.1 Distance Spectra

For a specific pseudo-random interleaver, as well as for the average of all interleavers of a specific length, the Turbo code distance spectrum is highly influenced

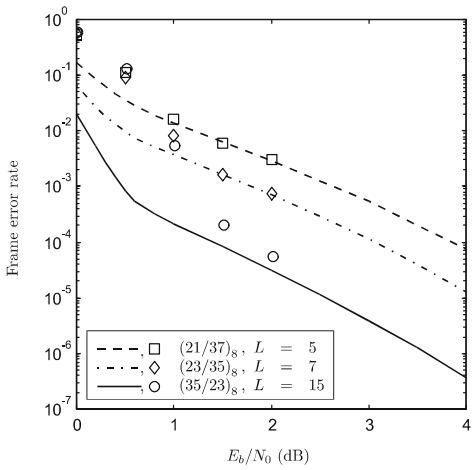


Figure 4.1: Tangential sphere upper bound (lines) of the performances of rate-1/3 Turbo codes using three different generator polynomials, for the ensemble of all 500-bit interleavers. Also shown are the corresponding simulation results (markers).

by the period L of the feedback polynomial. In general, the longer the period, the better the distance spectrum of the Turbo code. Especially, the minimum distance and the *effective free distance* [24] are directly influenced by the period length for reasons outlined in Section 2.3. Thus, the feedback period has a large impact on the asymptotic performance of the code, that is, the performance at high SNRs [71]. As an illustration, the tangential sphere bounds [73, 81] of the ML-decoding performances of three Turbo codes using different constituent encoders are shown in Figure 4.1 (lines). These correspond to the distance spectra achieved with the ensemble of all 500-bit interleavers, for the respective constituent codes. The compared codes all use 16-state constituent encoders, described by the generator polynomials $(21/37)_8$, $(23/35)_8$ and $(35/23)_8$, with periods 5, 7 and 15, respectively. As expected, the best distance spectrum corresponds to the feedback polynomial with the longest period, *i.e.* $(35/23)_8$. The relative performances are also verified by simulation results (markers) for each choice of generator polynomial, obtained using a new pseudo-random interleaver for each transmitted block. For $E_b/N_0 \gtrsim 1.5$ dB, the simulation results approach the tangential sphere upper bound.

Figure 4.2 shows simulated frame error rate performances of Turbo codes using the same generator polynomials as above, but for 4096-bit interleavers. As seen, these performances suffer from the same type of error floors as the 500-bit Turbo codes, caused by the poor minimum distances resulting from the use of random interleaving. Together with Figure 4.1 this illustrates that the

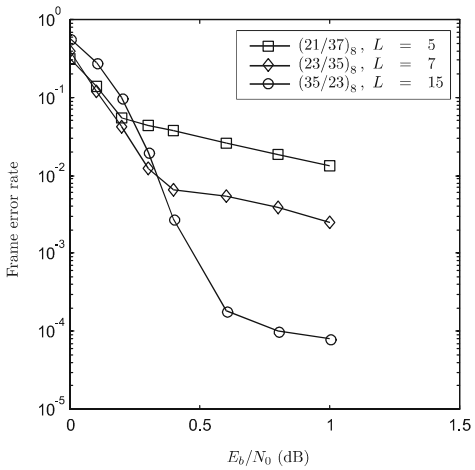


Figure 4.2: Simulated frame error rate performances of rate-1/3 Turbo codes using random 4096-bit interleaving, and different constituent encoders.

length of the feedback period is important for both small and large interleavers.

Having observed how the period L influences Turbo code distance spectra, it is reasonable to choose feedback polynomials with as large periods as possible. Since primitive polynomials achieve the largest possible L for a given number of memory elements, the use of primitive feedback polynomials is common in Turbo coding literature.

The performances obtained in Figures 4.1 and 4.2 are valid for the ensemble of interleavers of a certain length. When using designed interleavers, the feedback period L is not necessarily as important. With sophisticated interleaver design, the lower part of the distance spectra can often be made sufficiently good also for feedback polynomials with shorter periods. 'Sufficiently good' implies that the code is operating at SNRs for which it has not reached its asymptotic performance, as dictated by its minimum distance and the corresponding multiplicity. For such codes and operating regions, the benefit of further improving the distance spectra is limited. As a consequence, there might be selection criteria other than the distance spectrum that deserves consideration when choosing the feedback polynomial. Such criteria, related to the performance of iterative decoding, are discussed in the next section.

4.2 Iterative Decoding Performance

The discussion on distance spectrum in the previous section provides guidelines for choosing the feedback polynomials for Turbo codes operating at high SNRs. For low and medium SNRs, in the 'waterfall' region, the properties that govern

the performance of Turbo codes are more subtle. It has been observed, for example in [90], that the error correcting performance in this region is better for Turbo codes using non-primitive instead of primitive feedback polynomials. It is difficult to construct reliable bounds and estimates of the error performance based on the distance spectra in this region of SNRs. Hence, it is doubtful if it is possible to explain these observations using distance spectra argumentation. For example, the tangential sphere bounds for the 500-bit interleavers in Figure 4.1 indicate that the encoder using primitive feedback polynomials retains its superior performance throughout the entire range of SNRs. However, the simulation results at $E_b/N_0 = 0.5$ dB indicate that this code actually has the worst performance, among the three. The same behavior, even more evident, is observed for the 4096-bit interleavers in Figure 4.2.

The Turbo code properties that govern the error correcting performance in the waterfall region are still not completely understood. Since simulated performances are *above* the upper bounds valid for ML-decoding, the iterative decoding is clearly suboptimal. Hence, it may come to a point where the decoding performance is of higher importance than the distance properties of the code. As in the case of interleaver design, the choice of the constituent encoders influences the performance of iterative decoding. Here, we present a few indications on how different choices of constituent encoders affect the performance of iterative decoding. In particular, we focus on the choice of feedback polynomials, since these are found to have a large influence on said performance.

As the iterative decoding proceeds, the statistical dependencies increase between decoder inputs and outputs, as well as between the transmitted symbols and the decoder outputs. In [16], 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 reviewed briefly in the following paragraphs.

Let $I_E^{(k)}$ denote the mutual information between transmitted symbols (antipodal) and the corresponding extrinsic output after the kth half-iteration, $0 \leq I_E^{(k)} \leq 1$. For an iterative decoder that improves its decoding performance at the kth half-iteration this mutual information is increased, that is, $I_E^{(k)} > I_E^{(k-1)}$ [16]. In contrast, if $I_E^{(k)} = I_E^{(k-1)}$ the iterative decoding process does not improve in the kth half-iteration. Figure 4.3 shows principal appearances of the *extrinsic information transfer characteristics* of a soft-in/soft-out decoder [16]. The situation corresponding to $I_E^{(k)} = I_E^{(k-1)}$ is indicated by diagonal lines (dashed). In Figure 4.3(a), the influence of different SNRs is shown. An increase in E_b/N_0 results in increased mutual information at the decoder

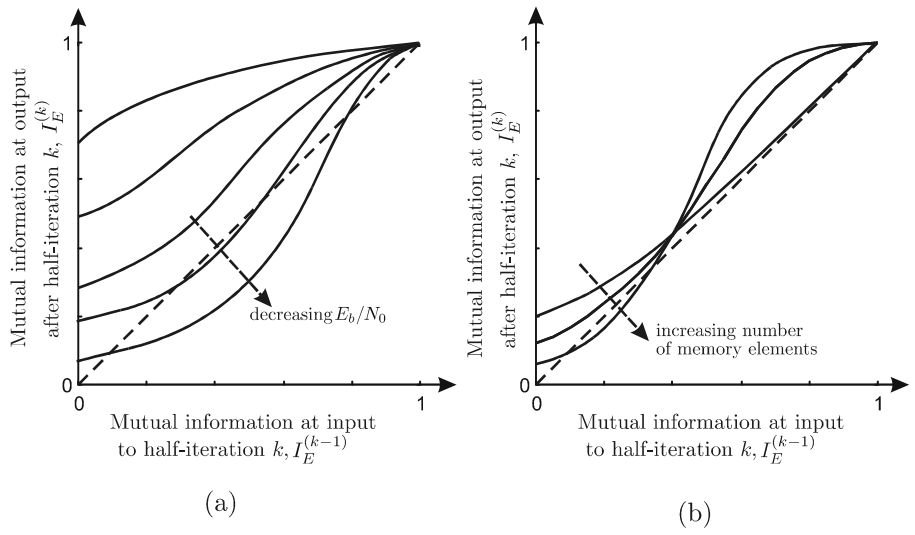


Figure 4.3: Principal appearances of extrinsic information transfer characteristics of soft-in/soft-out decoding of recursive systematic convolutional codes. (a) different SNRs for a specific generator polynomial, and (b) different number of encoder memory elements for a specific E_b/N_0.

output, for a given mutual information at the decoder input[1]. This corresponds to improved decoding performance. In Figure 4.3(b), the influence of using convolutional encoders with different number of memory elements is shown. For *a priori* inputs with low mutual information with the transmitted sequence (*i.e.* the lower parts 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 characteristics are also dependent on the generator polynomials [16]. Figure 4.4 shows the characteristics for two memory-4 constituent encoders using generator polynomials $(35/23)_8$ and $(21/37)_8$, respectively. It is noted that in the early decoding stages, that is for mutual information $I_E^{(k)} < 0.5$, the largest decoding progress for each half-iteration is achieved for the $(21/37)_8$-code. Equivalently, it is observed that if E_b/N_0 is decreased, the characteristic of the $(35/23)_8$-code reaches the diagonal first, which corresponds to a cease in the iterative decoding convergence.

[1] The extrinsic outputs from the previous half-iteration $(k-1)$ act as *a priori* inputs at half-iteration k.

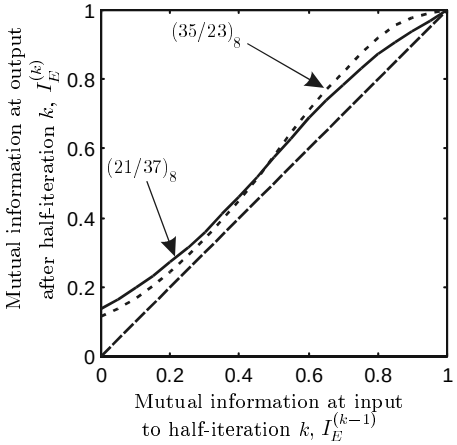


Figure 4.4: Illustration of extrinsic information transfer characteristic for constituent encoders with 4 memory elements and generator polynomials $(35/23)_8$ and $(21/37)_8$, respectively. Results from [16].

When designing Turbo codes that operate in the waterfall region, the E_b/N_0 is such that the iterative decoding is barely converging. Thus, the extrinsic information transfer characteristic has a region where it is just above the diagonals in Figure 4.3. This corresponds to only a slight increase in the mutual information $I_E^{(k)}$ for each new half-iteration. The region that tends to cause most problems regarding convergence is in the range $0.2 < I_E^{(k)} < 0.6$ [16]. In this region, the curves in Figure 4.3(b) indicate that the best convergence properties are obtained with constituent encoders with few memory elements. However, at later decoding steps the situation is reversed, as commented above. Thus, the idea of using *asymmetrical Turbo codes*, which implies different constituent encoders to be used in the Turbo encoder, is indeed attractive. Asymmetrical Turbo codes are investigated in [90, 16].

The extrinsic information transfer characteristics suggest that the 'Turbo-cliff' is obtained first, *i.e.* for the lowest SNR, with encoders using few memory elements and feedback polynomials with short period. This is verified in Figure 4.5, showing the bit error rate performances[2] of Turbo codes using memory-3 and memory-4 encoders and 4096-bit interleavers. The interleavers are designed using the algorithm in Section 3.2. In Figure 4.5(a), memory-3 constituent encoders with generator polynomials $(17/15)_8$ and $(15/17)_8$ are used. These feedback polynomials have periods $L = 7$ and $L = 5$, respectively. As

[2] The plots presented in this section use bit error rate as the performance measure. The same principle holds also for frame error rates, although the differences in E_b/N_0 are smaller, or sometimes negligible.

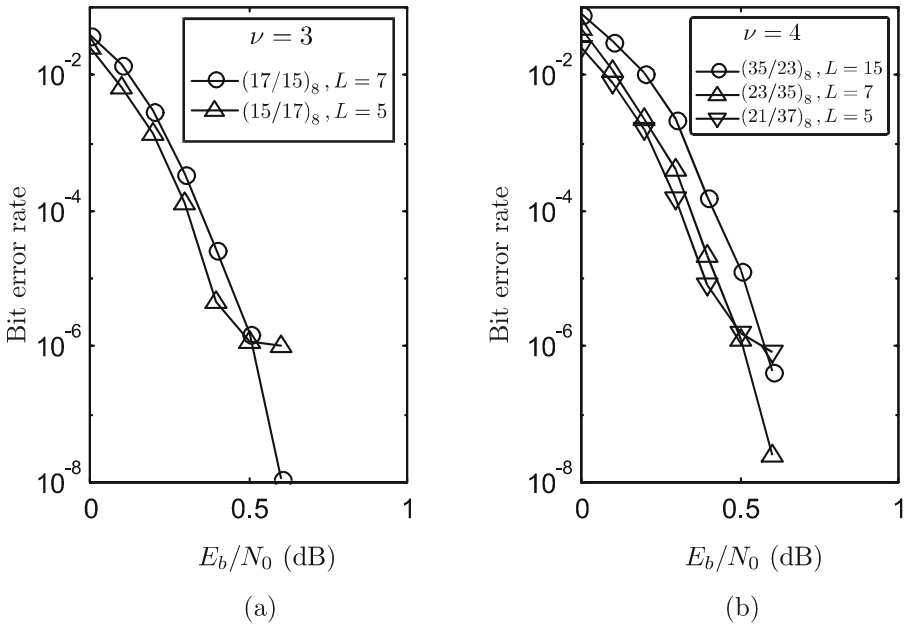


Figure 4.5: Simulated bit error rate performances of rate-1/3 Turbo codes using (a) memory-3 and (b) memory-4 constituent encoders. The interleaver size is 4096 bits, and each Turbo code uses an interleaver designed for the particular constituent encoders.

anticipated, the Turbo-cliff appears first for the non-primitive feedback polynomial, that is, polynomial 17_8. The same behavior is observed in Figure 4.5(b), showing memory-4 encoders with generator polynomials $(35/23)_8$, $(23/35)_8$ and $(21/37)_8$, with periods 15, 7 and 5, respectively. For both the memory-3 and memory-4 encoders, the Turbo-cliff appear in the order suggested by the feedback polynomial periods.

It is also interesting to notice the effect on the distance spectrum resulting from the use of shorter period feedback. For $L = 15$ and $L = 7$, and for the error rates shown in Figure 4.5, that is, as low as 10^{-8} in BER, the interleaver design algorithm succeeds in designing interleavers for which the error floor is not reached. However, for $L = 5$, the error floor is reached at approximately 10^{-6} in BER, even though a designed interleaver is used.

Finally, we compare the performances of Turbo codes having the same periods of their feedback polynomials but different number of memory elements. For example, even though the number of memory elements differ, the polynomials 15_8 and 35_8 both have period $L = 7$, and the polynomials 17_8 and 37_8 both have period $L = 5$. The performances of the codes using these feedback

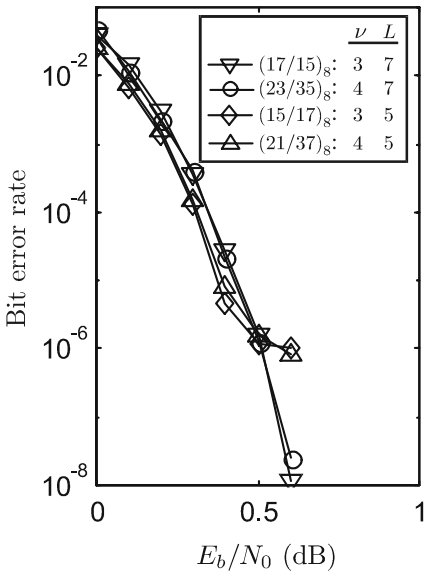


Figure 4.6: Comparison of simulated bit error rate performances of rate-1/3 Turbo codes using constituent encoders with feedback period $L = 5$ and $L = 7$, with different number of memory elements. The interleaver size is 4096 bits, and each Turbo code uses an interleaver designed for the particular constituent encoders.

polynomials in Figure 4.5 are shown in the same plot in Figure 4.6. It is remarkable how close the performances with identical polynomial periods are to each other, even though the number of memory elements differ. Indeed, the period of the feedback polynomial has the major influence on the location of both the Turbo-cliff and on the distance properties. In contrast, the number of encoder memory elements has virtually no influence when comparing the above encoders with identical periods.

To conclude, when choosing feedback polynomial there is a trade off between a low error-floor and an early Turbo-cliff; either a long L with a corresponding low error-floor, or a short L with an early Turbo-cliff.

Figure 4.7 shows the bit error rate performances achieved with 500-bit interleavers, using the same constituent encoders as for the 4096-bit interleavers in Figure 4.5. The 500-bit interleavers are also designed with the algorithm presented in Section 3.2. The behavior is in principle the same as for the 4096-bit interleavers. However, very little gain is achieved, if any, by using polynomials with period 5 instead of 7. Again, the performances are more influenced by the period of the feedback polynomial than by the number of memory elements in the encoders. This is verified by comparing the performances of the $(17/15)_8$-

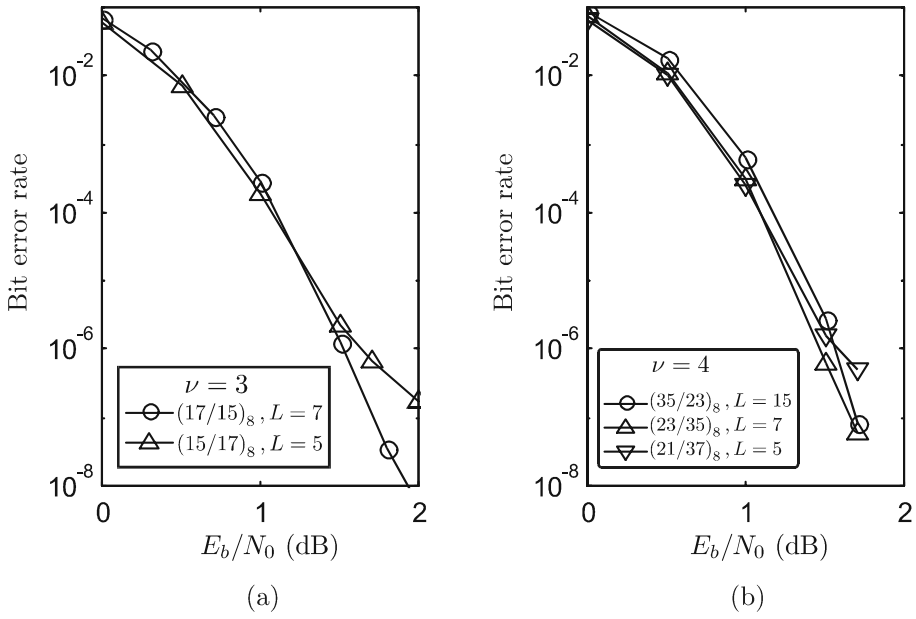


Figure 4.7: Simulated bit error rate performances of rate-1/3 Turbo codes using (a) memory-3 and (b) memory-4 constituent encoders. The interleaver size is 500 bits, and each Turbo code uses an interleaver designed for the particular constituent encoders.

and $(23/35)_8$-codes, as well as the $(15/17)_8$- and $(21/37)_8$-codes.

Finally, we stress the importance of using properly designed interleavers in conjunction with constituent encoders having short periods. For this purpose, we compare the performances of two additional interleavers: random interleaving and golden interleavers [21]. In short, random interleaving result in neither good distance spectra nor good correlation properties. Golden interleavers, on the other hand, result in very good correlation properties, but often suffer from large multiplicities of relatively low-weight codewords. Figure 4.8 shows the BER performances using generator polynomials $(35/23)_8$ and $(17/15)_8$ for 500-bit (a) golden-, (b) random- and (c) designed interleavers. As seen, for the golden- and random interleaving alternatives the use of the memory-3 encoders has better performance above BER of approximately 10^{-4}, while the corresponding limit for the designed interleaver is at approximately 10^{-7} in BER. As asserted, the benefits of using a feedback polynomial with short period are valid for a much wider range of error rates when using properly designed interleavers.

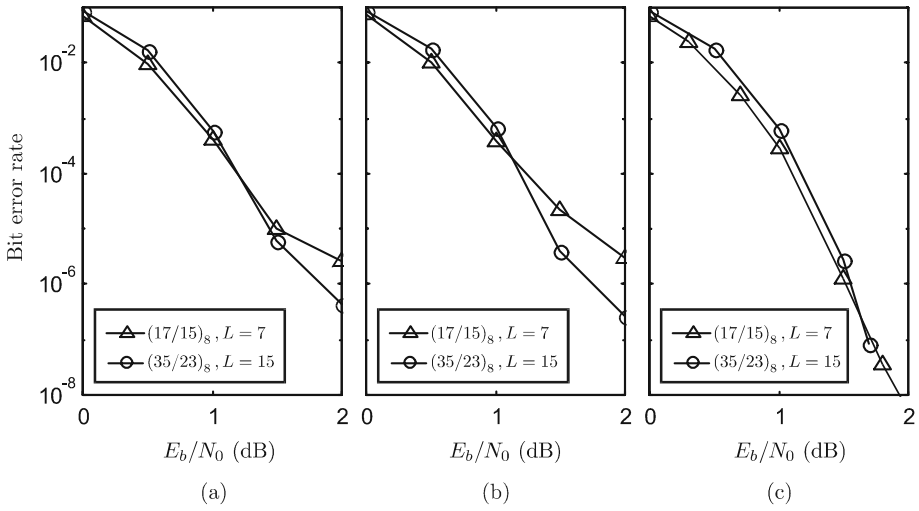


Figure 4.8: Simulated bit error rate performances of rate-1/3 Turbo codes using 500-bit interleavers and memory-3 and memory-4 constituent encoders with primitive feedback polynomials. (a) Golden interleaver, (b) random interleaving, and (c) designed interleavers.

4.3 Summary

In this chapter, the influence of the constituent encoders on the performance of Turbo codes has been reviewed, with focus on the feedback polynomials. In particular, their impact on the distance spectra and the iterative decoding performance is investigated. It is concluded that there is a single parameter that has a strong relation to both these issues: the period of the feedback polynomial. In essence, a long period L has a positive impact on the lower part of the Turbo code distance spectra. Thus, for asymptotic performances, *i.e.* at high SNRs, it is advantageous to use constituent encoders with long periods. For a given number of encoder memory elements ν, the longest L possible is obtained with primitive feedback polynomials, for which $L = 2^{\nu} - 1$.

For low SNRs, *i.e.* when the Turbo code is operating in the waterfall region, it is sometimes advantageous to use feedback polynomials with shorter periods. This is especially true when using designed interleavers, since these can mitigate the deteriorating effect that a short L has on the distance spectrum. For complexity reasons, it is natural to decrease the value of L by reducing the number of memory elements in the encoders. In our investigations, nearly identical error performances are obtained for Turbo codes using constituent encoders with identical feedback polynomial periods but with different number of memory elements.

Chapter 5

Conclusions

There are two purposes of this first part of the thesis. Firstly, it is an introduction to Turbo codes in general and, secondly, it is an overview of the research area, including a review of the major results and contributions to the field of designing Turbo codes.

There are different definitions on what 'Turbo codes' are. The codes investigated in this thesis are constructed via parallel concatenation of two *recursive convolutional codes* (RSC). The parallel concatenation is implemented by interleaving, *i.e.* re-ordering, the information sequence before it is input to the second constituent encoder. Two of the most central parts of a Turbo code encoder are thus the *interleaver* and the *constituent encoders*. Other central aspects of a Turbo encoder are *trellis termination* and *puncturing*. Trellis termination is an issue brought into attention as a consequence of the recursive property of the constituent encoders, in combination with the interleaver. Puncturing is the process of excluding bits from the outputs from the constituent encoders, so that the concatenated transmitted sequence is a decimated version of the encoder outputs.

One of the major breakthroughs related to the introduction of Turbo codes is the process of *iterative decoding*. With iterative decoding, two or more *constituent decoders* take turns in attempting to decode the received message. In every new attempt, each decoder make use of the outcome from the other decoder's last decoding attempt. Properly formulated, and with sufficient signal-to-noise ratio, the process of iterative decoding is remarkably successful in refining the decoder outputs until the two decoders converge to a joint decision.

The reason to use iterative decoding is related to the invincible complexity (still) associated with maximum likelihood decoding of Turbo codes. Iterative decoding offers efficient decoding of a complex and powerful code, however to the price of being suboptimal to maximum likelihood decoding.

The subject of this thesis is the design of Turbo codes, that is, how to choose and/or design the components in a Turbo code encoder to get the best possible

performances. Traditionally, the issue of designing codes is focused on the goal of creating the best possible *Hamming distance spectrum*. Indeed, this is a major issue also in the case of designing Turbo codes. However, since Turbo 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 Turbo code encoder.

In order to investigate the performance of iterative decoding, the concept of studying the correlation properties between decoder inputs is introduced. It is found that the correlation properties are strongly influenced by the interleaver choice. These findings can be exploited in two ways. Firstly, it is possible to investigate the properties of an interleaver, to determine wether it is suitable for an iterative decoding environment or not. For this purpose, the IDS-measure was introduced, as a means of assessing the *iterative decoding suitability* of an interleaver without performing extensive simulations. Secondly, it is possible to form an interleaver design criterion based on the correlation properties.

An extensive part of the presented work is on the issue of interleaver design. For the purpose of constructing high-performance interleavers, two design criteria are proposed in this thesis. The first criterion is based on Turbo code distance spectra. This criterion relies on the observation that the lower part of a Turbo code distance spectrum originates from input sequences of low weight, in particular of weight-2, 3 and 4. The second criterion targets the performance of iterative decoding, based on the correlation properties between decoder inputs. An important result of using correlation properties in the interleaver design is a faster convergence of the iterative decoding process. The performance of interleavers designed with both the code/distance- and decoding/correlation-criteria are very competitive compared to other interleaver designs reported in the literature. In fact, to our knowledge there are no other design methodologies that result in better performing interleavers. Finally, the attempt to design interleavers resulting in codes with both good distance properties and good iterative decoding performance is by no means counteracting. On the contrary, the correlation criterion has a beneficial influence also on the code distance spectrum.

The choice of the constituent encoders has also been reviewed. One of the important parameters of the recursive convolutional encoders that has a strong relation to the Turbo code performance is the period of the feedback polynomial. Again, both the distance spectrum and the performance of iterative decoding are influenced by this parameter. However, in contrast to the case of interleaver design, the choice of the feedback period involves a trade-off: A long period is beneficial for the distance spectrum, and therefore important for the performance at high SNRs. On the other hand, at low SNRs a short period results in a better iterative decoding performance. Thus, the choice of feedback period is dependent on the target operating SNR of the Turbo code. The use of interleavers designed with sophisticated methods increases the region for which the advantages of a short period dominate. Thus, properly designed interleavers not only improve Turbo code performance, but can also decrease the decoding

complexity. This follows since shorter feedback periods are associated with a smaller number of memory elements in the constituent encoders. In addition, interleavers with good correlation properties have faster iterative decoding convergence, and thus, require fewer decoding iterations.

The issue of trellis termination is not thoroughly described in this part of the thesis. However, trellis termination for Turbo codes is addressed in [Paper VI]. There, the distance spectra of Turbo codes using different termination methods are investigated, as well as the issue of designing interleavers customized to the termination method. As an important result thereof, it is demonstrated that the error-correcting performances of Turbo codes using different termination methods, including no termination at all, are very similar as long as custom designed interleavers are used.

The investigations in this thesis have been restricted to a specific rate 1/3 Turbo code structure and antipodal modulation over a memoryless AWGN channel. This has been favorable in the sense that the degree of freedom in the design has been kept at a reasonable level, allowing for more detailed studies of confined issues. On the other hand, it also leaves a number of interesting questions open for further investigation. This relates both to the Turbo code structure and the transmission environment, in the form of modulation and channel characteristics. This thesis shows that certain statistical dependencies between the inputs to the constituent decoders have a deteriorating effect on the performance of iterative decoding. Such statistical dependencies can, in addition to those addressed in this thesis, be introduced by channels containing memory, fading, and colored noise. One or more of these are present in most practical communication systems, making the understanding of Turbo codes in such environments an urgent extension to what is known today.

Appendix A

The BCJR Algorithm

A.1 Theoretical Description

The BCJR algorithm [3] 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 derivations in [3, 12, 59, 77].

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 $u_1^N = (u_1, u_2, \ldots, u_N)$ and the parity sequence generated by a rate-1/2 recursive convolutional encoder $c_1^N = (c_1, c_2, \ldots, c_N)$, where $u_n, c_n \in \{0, 1\}$. This notation is illustrated in Figure A.1, showing an encoder with generator polynomials $\begin{pmatrix} 1 & 7/5 \end{pmatrix}_8$. Using binary antipodal modulation, the u_n:s and c_n:s are converted to ± 1 before being transmitted on the channel. With the additive white Gaussian noise channel model, the received systematic and parity sequences, $x_1^N = (x_1, x_2, \ldots, x_N)$ and $y_1^N = (y_1, y_2, \ldots, y_N)$, are

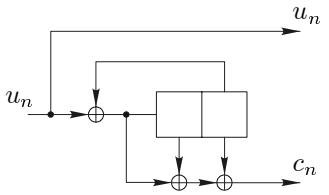


Figure A.1: Example of a systematic recursive convolutional encoder with generator polynomials $(1\ 7/5)_8$.

modeled as

$$x_n = (2u_n - 1) + w_{x,n} \tag{A.1}$$
$$y_n = (2c_n - 1) + w_{y,n}, \tag{A.2}$$

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

$$w_{x,n}, w_{y,n} \sim N\left(0, \sigma^2\right), \quad n = 1, \ldots, N. \tag{A.3}$$

Apart form the received code sequences x_1^N and y_1^N, the decoder has access to *a priori* information for each transmitted bit, denoted $\tilde{\Lambda}_n$. The *a priori* information is given as the logarithm of the prior probabilities that information bit u_n is 1and 0 respectively, that is,

$$\tilde{\Lambda}_n = \ln \frac{\Pr\left(u_n = 1\right)}{\Pr\left(u_n = 0\right)}. \tag{A.4}$$

The decoder inputs for an entire block of length N is denoted R_1^N, where $R_1^N = \left(x_1^N, y_1^N, \tilde{\Lambda}_1^N\right)$.

The objective of the decoder is to calculate the *a posteriori* probabilities that information symbol u_n is 1 or 0, based on the received code sequences and the available *a priori* information, that is, calculate $\Pr\left(u_n = 1|R_1^N\right)$ and $\Pr\left(u_n = 0|R_1^N\right)$. 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 in Figure A.1 is shown in Figure A.2. Each state transition is labeled with the systematic and parity bit generated by the encoder. In order to calculate $\Pr\left(u_n = 1|R_1^N\right)$ and $\Pr\left(u_n = 0|R_1^N\right)$, the probability of trellis transitions corresponding to $u_n = 1$ and $u_n = 0$, marked with solid and dashed lines in Figure A.2, is summed. Denoting the encoder state after the nth input symbol by $S_n \in \{0, 1, \ldots, 2^\nu - 1\}$, where ν is the number of memory elements in the encoder, we get

$$\Pr\left(u_n = i|R_1^N\right) = \sum_{s'=0}^{2^\nu-1} \sum_{s=0}^{2^\nu-1} \Pr\left(u_n = i, S_{n-1} = s', S_n = s|R_1^N\right), \quad u_n = 0, 1. \tag{A.5}$$

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

$$\sigma_n^i(s', s) \triangleq p\left(u_n = i, S_{n-1} = s', S_n = s, R_1^N\right), \tag{A.6}$$

through which (A.5) becomes

$$\Pr\left(u_n = i|R_1^N\right) = \sum_{s'=0}^{2^\nu-1} \sum_{s=0}^{2^\nu-1} \sigma_n^i(s', s)/p\left(R_1^N\right). \tag{A.7}$$

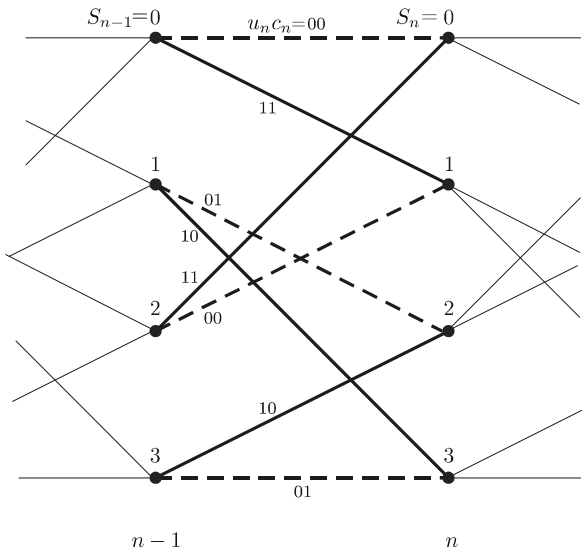


Figure A.2: A 4-state trellis diagram for an RSC encoder with generator polynomials described by $(1\ 7/5)_8$ .The trellis transitions corresponding to input 0's and 1's are marked with dashed and solid lines, respectively.

The probability density function (pdf) $\sigma_n^i(s', s)$ can be calculated in a recursive manner by factorizing it into three parts, denoted $\tilde{\alpha}_{n-1}(s')$, $\tilde{\gamma}_n^i(R_n, s', s)$ and $\tilde{\beta}_n(s)$. Firstly,

$$\begin{aligned}\sigma_n^i(s', s) &= p(u_n = i, S_{n-1} = s', S_n = s, R_1^n) \cdot \\ &\quad p\left(R_{n+1}^N | u_n = i, S_{n-1} = s', S_n = s, R_1^n\right) \\ &= p\left(S_{n-1} = s', R_1^{n-1}\right) p\left(u_n = i, S_n = s, R_n | S_{n-1} = s', R_1^{n-1}\right) \cdot \\ &\quad p\left(R_{n+1}^N | u_n = i, S_{n-1} = s', S_n = s, R_1^n\right). \end{aligned} \tag{A.8}$$

Here, $p\left(u_n = i, S_n = s, R_n | S_{n-1} = s', R_1^{n-1}\right) = p(u_n = i, S_n = s, R_n | S_{n-1} = s')$, since the information bit u_n, the state S_n and the decoder input R_n are all independent on R_1^{n-1} if the encoder state S_{n-1} is known. Similarly, $p\left(R_{n+1}^N | u_n = i, S_{n-1} = s', S_n = s, R_1^n\right) = p\left(R_{n+1}^N | S_n = s\right)$. Thus,

$$\begin{aligned}\sigma_n^i(s', s) &= p\left(S_{n-1} = s', R_1^{n-1}\right) p(u_n = i, S_n = s, R_n | S_{n-1} = s') \cdot \\ &\quad p\left(R_{n+1}^N | S_n = s\right) \\ &= \tilde{\alpha}_{n-1}(s')\, \tilde{\gamma}_n^i(R_n, s', s)\, \tilde{\beta}_n(s), \end{aligned} \tag{A.9}$$

where

$$\tilde{\alpha}_{n-1}(s') \triangleq p\left(S_{n-1}=s', R_1^{n-1}\right), \tag{A.10}$$

$$\tilde{\gamma}_n^i(R_n, s', s) \triangleq p\left(u_n=i, S_n=s, R_n|S_{n-1}=s'\right), \text{ and} \tag{A.11}$$

$$\tilde{\beta}_n(s) \triangleq p\left(R_{n+1}^N|S_n=s\right). \tag{A.12}$$

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 and succeeding counterparts, respectively. More precisely,

$$\begin{aligned}
\tilde{\alpha}_n(s) &= p\left(S_n=s, R_1^n\right) \\
&= \sum_{s'=0}^{2^\nu-1}\sum_{i=0}^{1} p\left(u_n=i, S_{n-1}=s', S_n=s, R_1^n\right) \\
&= \sum_{s'=0}^{2^\nu-1}\sum_{i=0}^{1} p\left(S_{n-1}=s', R_1^{n-1}\right)\cdot \\
&\quad\; p\left(u_n=i, S_n=s, R_n|S_{n-1}=s', R_1^{n-1}\right) \\
&= \sum_{s'=0}^{2^\nu-1}\sum_{i=0}^{1} p\left(S_{n-1}=s', R_1^{n-1}\right) p\left(u_n=i, S_n=s, R_n|S_{n-1}=s'\right) \\
&= \sum_{s'=0}^{2^\nu-1}\sum_{i=0}^{1} \tilde{\alpha}_{n-1}(s')\,\tilde{\gamma}_n^i(R_n, s', s),
\end{aligned} \tag{A.13}$$

and

$$\begin{aligned}
\tilde{\beta}_n(s') &= p\left(R_{n+1}^N|S_n=s'\right) \\
&= \sum_{s=0}^{2^\nu-1}\sum_{i=0}^{1} p\left(u_{n+1}=i, S_{n+1}=s, R_{n+1}^N|S_n=s'\right) \\
&= \sum_{s=0}^{2^\nu-1}\sum_{i=0}^{1} p\left(R_{n+2}^N|u_{n+1}=i, S_{n+1}=s, R_{n+1}, S_n=s'\right)\cdot \\
&\quad\; p\left(u_{n+1}=i, S_{n+1}=s, R_{n+1}, S_n=s'\right)/\Pr\left(S_n=s'\right) \\
&= \sum_{s=0}^{2^\nu-1}\sum_{i=0}^{1} p\left(R_{n+2}^N|S_{n+1}=s\right)\cdot \\
&\quad\; p\left(u_{n+1}=i, S_{n+1}=s, R_{n+1}|S_n=s'\right) \\
&= \sum_{s=0}^{2^\nu-1}\sum_{i=0}^{1} \tilde{\beta}_{n+1}(s)\,\tilde{\gamma}_{n+1}^i(R_{n+1}, s', s).
\end{aligned} \tag{A.14}$$

The calculations of the $\tilde{\alpha}$:s and the $\tilde{\beta}$:s according to (A.13) and (A.14) are referred to as the forward and the backward recursion.

It remains to calculate the density functions $\tilde{\gamma}_n^i(R_n, s', s)$. From its definition in (A.11), $\tilde{\gamma}_n^i(R_n, s', s)$ can be expressed

$$\begin{aligned}\tilde{\gamma}_n^i(R_n, s', s) &= p(u_n = i, S_n = s, R_n | S_{n-1} = s') \\ &= p(R_n | u_n = i, S_n = s, S_{n-1} = s') \cdot \\ &\quad \Pr(S_n = s | u_n = i, S_{n-1} = s') \Pr(u_n = i). \end{aligned} \tag{A.15}$$

Assuming that the encoder inputs are independent, conditioned on a certain trellis transition, and exchanging R_n for $\left(x_n, y_n, \tilde{\Lambda}_n\right)$, we get

$$\begin{aligned}\tilde{\gamma}_n^i(R_n, s', s) &= p\left(x_n, y_n, \tilde{\Lambda}_n | u_n = i, S_n = s, S_{n-1} = s'\right) \cdot \\ &\quad \Pr(S_n = s | u_n = i, S_{n-1} = s') \Pr(u_n = i) \\ &= p(x_n | u_n = i, S_n = s, S_{n-1} = s') \cdot \\ &\quad p(y_n | u_n = i, S_n = s, S_{n-1} = s') \cdot \\ &\quad p\left(\tilde{\Lambda}_n | u_n = i, S_n = s, S_{n-1} = s'\right) \cdot \\ &\quad \Pr(S_n = s | u_n = i, S_{n-1} = s') \Pr(u_n = i) \\ &= p(x_n | u_n = i) \cdot \\ &\quad p(y_n | u_n = i, S_{n-1} = s') \cdot \\ &\quad p\left(\tilde{\Lambda}_n | u_n = i\right) \Pr(u_n = i) \cdot \\ &\quad \Pr(S_n = s | u_n = i, S_{n-1} = s') \\ &= p(x_n | u_n = i) \cdot \\ &\quad p(y_n | u_n = i, S_{n-1} = s') \cdot \\ &\quad \Pr\left(u_n = i | \tilde{\Lambda}_n\right) p\left(\tilde{\Lambda}_n\right) \cdot \\ &\quad \Pr(S_n = s | u_n = i, S_{n-1} = s'). \end{aligned} \tag{A.16}$$

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

$$p(x_n | u_n = i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_n - (2i-1))^2}{2\sigma^2}}, \text{ and} \tag{A.17}$$

$$p(y_n | u_n = i, S_{n-1} = s') = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_n - (2c_n - 1))^2}{2\sigma^2}}. \tag{A.18}$$

Further, $\Pr\left(u_n = i | \tilde{\Lambda}_n\right)$ is the *a priori* probability that information symbol $u_n = i$. Since $\tilde{\Lambda}_n = \ln \frac{\Pr(u_n=1)}{\Pr(u_n=0)}$, the *a priori* probabilities are

$$\Pr\left(u_n = 1 | \tilde{\Lambda}_n\right) = \frac{e^{\tilde{\Lambda}_n}}{1 + e^{\tilde{\Lambda}_n}}, \text{ and} \tag{A.19}$$

$$\Pr\left(u_n = 0 | \tilde{\Lambda}_n\right) = \frac{1}{1 + e^{\tilde{\Lambda}_n}}. \tag{A.20}$$

The probability density function $p\left(\tilde{\Lambda}_n\right)$ in (A.16) is unknown; however, when forming the log-likelihood ratio Λ_n below, $p\left(\tilde{\Lambda}_n\right)$ is a constant factor common to both the numerator and denominator. Similarly, knowledge of $p\left(\tilde{\Lambda}_n\right)$ is in principle required in the recursive calculations of the $\tilde{\alpha}$:s and the $\tilde{\beta}$:s. However, since $p\left(\tilde{\Lambda}_n\right)$ is the same for all the transitions in a given section of the trellis (*i.e.* for a given n), it appears as a constant factor of all $\tilde{\alpha}_n(s)$ and $\tilde{\beta}_n(s)$, $s \in \{0, \ldots, 2^\nu - 1\}$. As discussed in further detail in Section A.2 below, factors common to all states need not be included in the calculations. Finally, the probability $\Pr(S_n = s | u_n = i, S_{n-1} = s')$ in (A.16) 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 i.

The *a posteriori* log-likelihood ratio, denoted Λ_n, can now be expressed as

$$\begin{aligned}
\Lambda_n &\triangleq \ln \frac{\Pr\left(u_n = 1 | R_1^N\right)}{\Pr\left(u_n = 0 | R_1^N\right)} \\
&= \ln \frac{\sum_{s'} \sum_s \tilde{\alpha}_{n-1}(s') \tilde{\gamma}_n^1(R_n, s', s) \tilde{\beta}_n(s)}{\sum_{s'} \sum_s \tilde{\alpha}_{n-1}(s') \tilde{\gamma}_n^0(R_n, s', s) \tilde{\beta}_n(s)} \\
&= \ln \frac{\sum_{s'} \sum_s \left(\tilde{\alpha}_{n-1}(s') p(x_n | u_n = 1) p(y_n | u_n = 1, S_{n-1} = s') \cdot \right.}{\sum_{s'} \sum_s \left(\tilde{\alpha}_{n-1}(s') p(x_n | u_n = 0) p(y_n | u_n = 0, S_{n-1} = s') \cdot \right.} \\
&\quad \left. \frac{\Pr\left(u_n = 1 | \tilde{\Lambda}_n\right) p\left(\tilde{\Lambda}_n\right) \Pr(S_n = s | u_n = 1, S_{n-1} = s') \tilde{\beta}_n(s)\right)}{\Pr\left(u_n = 0 | \tilde{\Lambda}_n\right) p\left(\tilde{\Lambda}_n\right) \Pr(S_n = s | u_n = 0, S_{n-1} = s') \tilde{\beta}_n(s)\right)} \\
&= \ln \frac{p(x_n | u_n = 1) \Pr\left(u_n = 1 | \tilde{\Lambda}_n\right) \sum_{s'} \sum_s \left(\tilde{\alpha}_{n-1}(s') \cdot \right.}{p(x_n | u_n = 0) \Pr\left(u_n = 0 | \tilde{\Lambda}_n\right) \sum_{s'} \sum_s \left(\tilde{\alpha}_{n-1}(s') \cdot \right.} \\
&\quad \left. \frac{p(y_n | u_n = 1, S_{n-1} = s') \Pr(S_n = s | u_n = 1, S_{n-1} = s') \tilde{\beta}_n(s)\right)}{p(y_n | u_n = 0, S_{n-1} = s') \Pr(S_n = s | u_n = 0, S_{n-1} = s') \tilde{\beta}_n(s)\right)} \\
&= \ln \frac{p(x_n | u_n = 1)}{p(x_n | u_n = 0)} + \ln \frac{\Pr\left(u_n = 1 | \tilde{\Lambda}_n\right)}{\Pr\left(u_n = 0 | \tilde{\Lambda}_n\right)} + \\
&\quad \ln \frac{\Pr\left(u_n = 1 | x_1^{n-1}, x_{n+1}^N, y_1^N, \tilde{\Lambda}_1^{n-1}, \tilde{\Lambda}_{n+1}^N\right)}{\Pr\left(u_n = 0 | x_1^{n-1}, x_{n+1}^N, y_1^N, \tilde{\Lambda}_1^{n-1}, \tilde{\Lambda}_{n+1}^N\right)} \\
&= \frac{2}{\sigma^2} x_n + \tilde{\Lambda}_n + Le_n, \qquad \text{(A.21)}
\end{aligned}$$

where Le_n denote the decoder extrinsic output. The partitioning of the *a posteriori* log-likelihood ratio Λ_n is useful in an iterative decoding environment,

where it is only the extrinsic output Le_n that is passed on to the next decoding step. Since the first and second terms in (A.21) are easily obtained from the encoder inputs, the most straightforward way to calculate the extrinsic output is through

$$\begin{aligned} Le_n &= \Lambda_n - \frac{2}{\sigma^2}x_n - \tilde{\Lambda}_n \\ &= \ln \frac{\sum_{s'}\sum_{s}\tilde{\alpha}_{n-1}(s')\,\tilde{\gamma}_n^1(R_n,s',s)\,\tilde{\beta}_n(s)}{\sum_{s'}\sum_{s}\tilde{\alpha}_{n-1}(s')\,\tilde{\gamma}_n^0(R_n,s',s)\,\tilde{\beta}_n(s)} - \frac{2}{\sigma^2}x_n - \tilde{\Lambda}_n, \end{aligned} \tag{A.22}$$

where $\tilde{\alpha}_{n-1}(s')$ and $\tilde{\beta}_n(s)$ are recursively calculated using (A.13) and (A.14).

A.2 Implementation Aspects

When forming the LLR in (A.22), 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 $\tilde{\gamma}_n^i(R_n,s',s)$:s,

$$\begin{aligned} \tilde{\gamma}_n^i(R_n,s',s) &= \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x_n-(2i-1))^2}{2\sigma^2}}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_n-(2c_n-1))^2}{2\sigma^2}}\frac{e^{i\tilde{\Lambda}_n}}{1+e^{\tilde{\Lambda}_n}}p\left(\tilde{\Lambda}_n\right)\cdot \\ &\quad \Pr(S_n=s|u_n=i,S_{n-1}=s') \\ &= \frac{p\left(\tilde{\Lambda}_n\right)}{2\pi\sigma^2\left(1+e^{\tilde{\Lambda}_n}\right)}e^{-\frac{x_n^2+(2i-1)^2+y_n^2+(2c_n-1)^2}{2\sigma^2}}\cdot \\ &\quad e^{\frac{x_n(2i-1)+y_n(2c_n-1)}{\sigma^2}+i\tilde{\Lambda}_n}\Pr(S_n=s|u_n=i,S_{n-1}=s'). \end{aligned} \tag{A.23}$$

Using $(2i-1)^2=1$ and $(2c_n-1)^2=1$, we define

$$\begin{aligned} C_{\gamma_n} &\triangleq \frac{p\left(\tilde{\Lambda}_n\right)}{2\pi\sigma^2\left(1+e^{\tilde{\Lambda}_n}\right)}e^{-\frac{x_n^2+y_n^2+2}{2\sigma^2}}, \text{ and} \\ \gamma_n^i(R_n,s',s) &\triangleq e^{\frac{x_n(2i-1)+y_n(2c_n-1)}{\sigma^2}+i\tilde{\Lambda}_n}\Pr(S_n=s|u_n=i,S_{n-1}=s'), \end{aligned}$$

which leads to

$$\tilde{\gamma}_n^i(R_n,s',s) = C_{\gamma_n}\gamma_n^i(R_n,s',s). \tag{A.24}$$

Since C_{γ_n} is independent on both s and s', $\tilde{\gamma}_n^i(R_n,s',s)$ may be replaced by $\gamma_n^i(R_n,s',s)$ in (A.22). Further, the forward recursion can be written

$$\begin{aligned} \tilde{\alpha}_n(s) &= \sum_{s'=0}^{2^\nu-1}\sum_{i=0}^{1}\tilde{\alpha}_{n-1}(s')\,\tilde{\gamma}_n^i(R_n,s',s) \\ &= C_{\gamma_n}\sum_{s'=0}^{2^\nu-1}\sum_{i=0}^{1}\tilde{\alpha}_{n-1}(s')\,\gamma_n^i(R_n,s',s), \end{aligned} \tag{A.25}$$

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

$$\begin{aligned} \alpha_n(s) &\triangleq \sum_{s'=0}^{2^\nu-1} \sum_{i=0}^{1} \alpha_{n-1}(s')\, \gamma_n^i(R_n, s', s), \text{ and} \\ \beta_n(s') &\triangleq \sum_{s'=0}^{2^\nu-1} \sum_{i=0}^{1} \beta_{n+1}(s)\, \gamma_{n+1}^i(R_n, s', s). \end{aligned} \tag{A.26}$$

The initialization of the first set of α-values, *i.e.* $\alpha_0(s), s \in \{0, \ldots, 2^\nu - 1\}$, 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

$$\alpha_0(s) = \begin{cases} 1 & s = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{A.27}$$

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

$$\beta_N(s) = \begin{cases} 1 & s = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{A.28}$$

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

$$\beta_N(s) = \frac{1}{2^\nu}, \qquad s = 0, \ldots, 2^\nu - 1. \tag{A.29}$$

In each step in the recursive calculation of the α:s and the β:s 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 α:a and the β:s for a specific time n can be scaled so that their sums are 1, that is, $\sum_s \alpha_n(s) = 1$ and $\sum_s \beta_n(s) = 1$.

The implementation complexity of the BCJR-algorithm is further reduced by performing the operations in the log-domain [78]. Then, a suboptimal but high performing algorithm referred to as *Max-Log-MAP* is obtained using the approximation $\ln\left(e^{\delta_1} + e^{\delta_2}\right) \approx \max(\delta_1, \delta_2)$, which corresponds to

$$\ln\left(1 + e^{-|\delta_2 - \delta_1|}\right) \approx 0$$

since

$$\ln\left(e^{\delta_1} + e^{\delta_2}\right) = \max(\delta_1, \delta_2) + \ln\left(1 + e^{-|\delta_2 - \delta_1|}\right).$$

In [78], a correction of the above approximation is proposed, where pre-calculated values of $\ln\left(1 + e^{-|\delta_2 - \delta_1|}\right) = f_c(|\delta_2 - \delta_1|)$ is stored in a one-dimensional look-up table. This low-complexity algorithm is shown to perform very close to the original BCJR-algorithm, also when the look-up table consist of only a few entries.

Appendix B

Comments on the Interleaver Design Algorithm

The interleaver design algorithm described in Section 3.2 has certain aspects that deserve to be commented. Firstly, the concept of using a "non-global" design approach, regarding the correlation properties, needs to be motivated. Further, the algorithm offers a number of possible variations and parameters that can be varied. The influence of such variations are discussed briefly here, including:

B.1: The use of a non-global optimization criterion.

B.2: The design order, *i.e.* the order in which the positions is chosen from the set $\mathcal{M}$ (*cf.* Section 3.2).

B.3: The modelled correlation decay rate, *i.e.* the constant c in (3.4) and (3.5).

B.4: The effect of distance spectrum design restrictions, d_{design}.

In this appendix, the impact of the above issues are investigated, primarily by illustrating permutation matrices (*cf.* Section 2.1.2) and comparing IDS-values of different design variations. For each interleaver size, the reported IDS-values are normalized to the IDS value of a golden interleaver [21] of the same size. This is because golden interleavers have very good correlation properties, and they can easily be generated for every interleaver size. With this normalization, the resulting IDS-values give an indication on how much the correlation properties of the resulting interleaver is degraded, compared to what is known to be achievable.

B.1 Non-Global Correlation Criterion

In this section we compare the correlation properties of interleavers designed with the algorithm presented in Section 3.2 with those of golden interleavers. This is done in order to motivate the use of a non-global optimization criterion.

Golden interleavers are based on the golden section. The golden section divides a segment of length 1 into two parts with lengths g and $1-g$, such that the ratio of the longer segment to the entire segment is the same as the ratio of the shorter segment to the longer segment. Thus, $g/1 = (1-g)/g$, which yields $g = \left(\sqrt{5}-1\right)/2$. To generate a golden interleaver, consider the points obtained then starting with 0 and adding increments of length g, using modulo-1 arithmetic. The golden interleaver is obtained as the order in which these points should be read, if sorted in ascending order. In Matlab, golden interleavers can for example be generated by the following function:

```
function Interleaver = GoldenInterleaver(IntSize)

g = (sqrt(5)-1)/2;
RealPositions = mod((1:IntSize)*g, 1);
[temp, Interleaver] = sort(RealPositions);
```

Figure B.1 illustrates the permutation matrices of both golden-, correlation designed[1]- and pseudo-random interleavers with lengths 100, 500 and 1000 bits. Also shown are the respective IDS-values, normalized to the IDS of the golden interleavers. The effect of using a non-global correlation criterion is discernible in the upper left- and right corners of the correlation-designed permutation matrices, especially for the smaller interleaver sizes. Towards the end of the design, there are fewer and fewer positions to choose from in the design. For the very last element in $\mathcal{M}$ (*cf.* Section 3.2), the set $\mathcal{L}$ contains only one element, and there is no design freedom at all. It is therefore not guaranteed that suitable design choices can be made. However, from inspection of the permutation matrices and through comparison of IDS values, it is concluded that the non-global design algorithm succeeds in designing interleavers with desirable correlation properties. Note that these interleavers were designed without any distance spectrum criterion, that is, $d_{\text{design}} = 0$. The effect of using too restricting distance spectrum criteria is illustrated in Section B.4.

B.2 Design Order

The order in which the interleaver is designed, *i.e.* the one in which the positions are chosen from the set $\mathcal{M}$, is referred to as the *design order*. The result of using three different design orders is illustrated here: (1) design in straight order, from

[1] The correlation-designed interleavers were designed with the following paramters: $d_{\text{design}} = 0$, $c = 0.18$, design order: last to first.

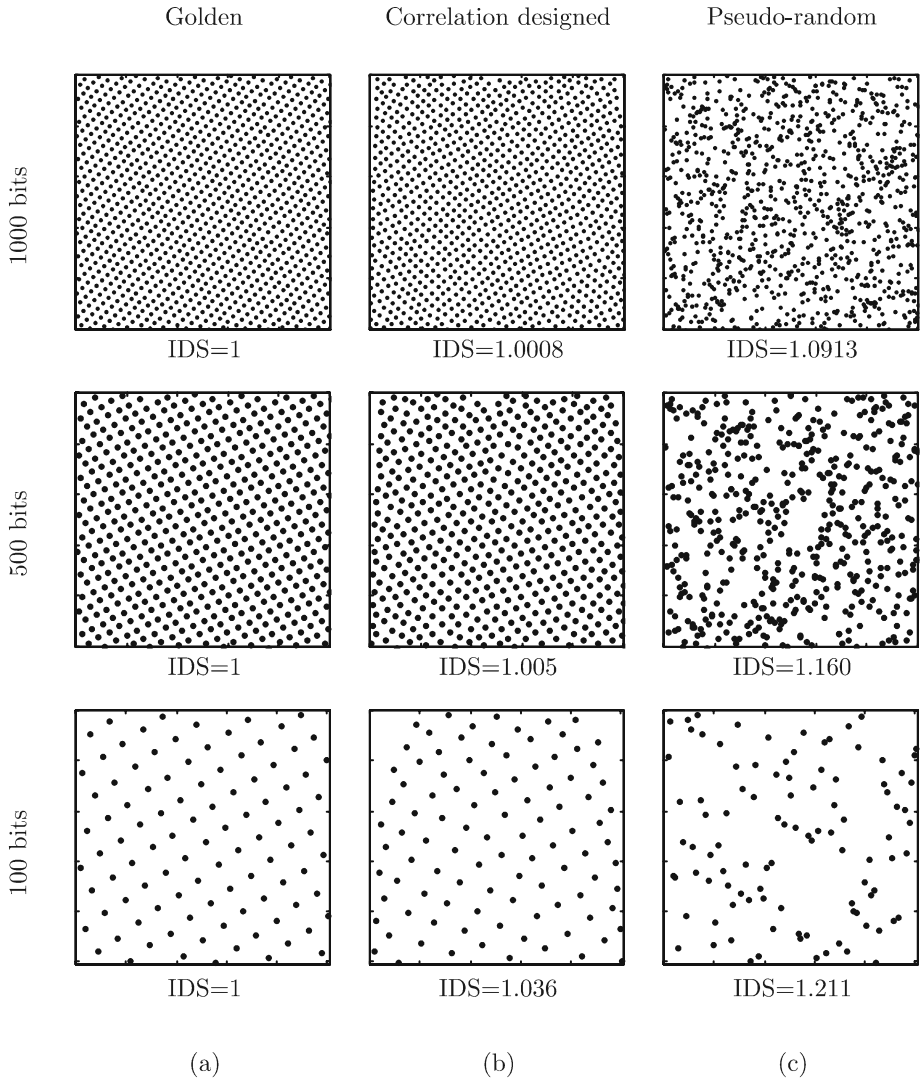


Figure B.1: Permutation matrices and IDS-values for golden-, correlation designed- and pseudo-random interleavers, 1000 bits (top) 500 bits (middle) and 100 bits (bottom) long. The correlation-designed interleaver have good correlation properties even though they are not designed with a global correlation criterion. The IDS-values are normalized to the values corresponding to the golden interleavers, for each interleaver size.

one end to the other, (2) alternatingly choosing the smallest and the largest element in $\mathcal{M}$, and (3) choosing the elements from $\mathcal{M}$ in random order. For the case of straight order design, the interleavers are designed from last to first, *i.e.* in reversed order. The interleavers illustrated here are designed without distance spectrum criteria, that is, $d_{\text{design}} = 0$.

The permutation matrices and the corresponding IDS-values are shown in Figure B.2, for 100-, 500- and 1000-bit interleavers. Judging from the IDS-values, designing the interleavers in straight order succeeds the best for the larger interleavers. For small interleavers, *i.e.* around 100 bits long, it might be advantageous to design interleavers in random order. However, since the differences compared to the straight-order design is small, the use of straight-order design is adopted for the entire range of interleaver sizes (interleavers smaller than 100 bits has not been considered).

B.3 Correlation Decay Rate

The correlation criterion (3.7) in the interleaver design algorithm includes the empirically determined parameter c. Using Monte-Carlo simulations, it was found that $c = 0.18$ approximately describes the correlation decay for the generator polynomial $(17/15)_8$. In this section, the influence of using faster and slower decay in the correlation model is illustrated.

As before, interleavers of sizes 100, 500 and 1000 bits are used, for which interleavers were designed using three different decay rates corresponding to $c = \{0.09, 0.18, 0.36\}$. All designs were performed without the distance spectrum criterion, *i.e.* $d_{\text{design}} = 0$. The resulting permutation matrices are shown in Figure B.3.

The result of using a correlation model that decays too slowly is obvious from inspecting the three left-most plots in Figure B.3, *i.e.* those obtained with $c = 0.09$. The positions near the edges are 'used up' too fast, with the consequence that towards the end of the design there are no edge positions left to choose from. The influence of this on the iterative decoding suitability is reflected in the IDS-values. As seen, the degradation is larger the smaller the interleaver.

The effect of using a correlation model that decays too fast is not as obvious. On the contrary, a fast decay generates interleavers with properties very similar to those designed with the 'normal' decay rate. Nevertheless, for large interleaver sizes the best IDS-values are achieved with the correlation decay that is in accordance with the empirically found decay rate.

B.4 Distance Spectrum Design Restriction

In the interleaver design examples presented so far in this appendix, the correlation-designed interleavers were all designed without distance spectra criteria,

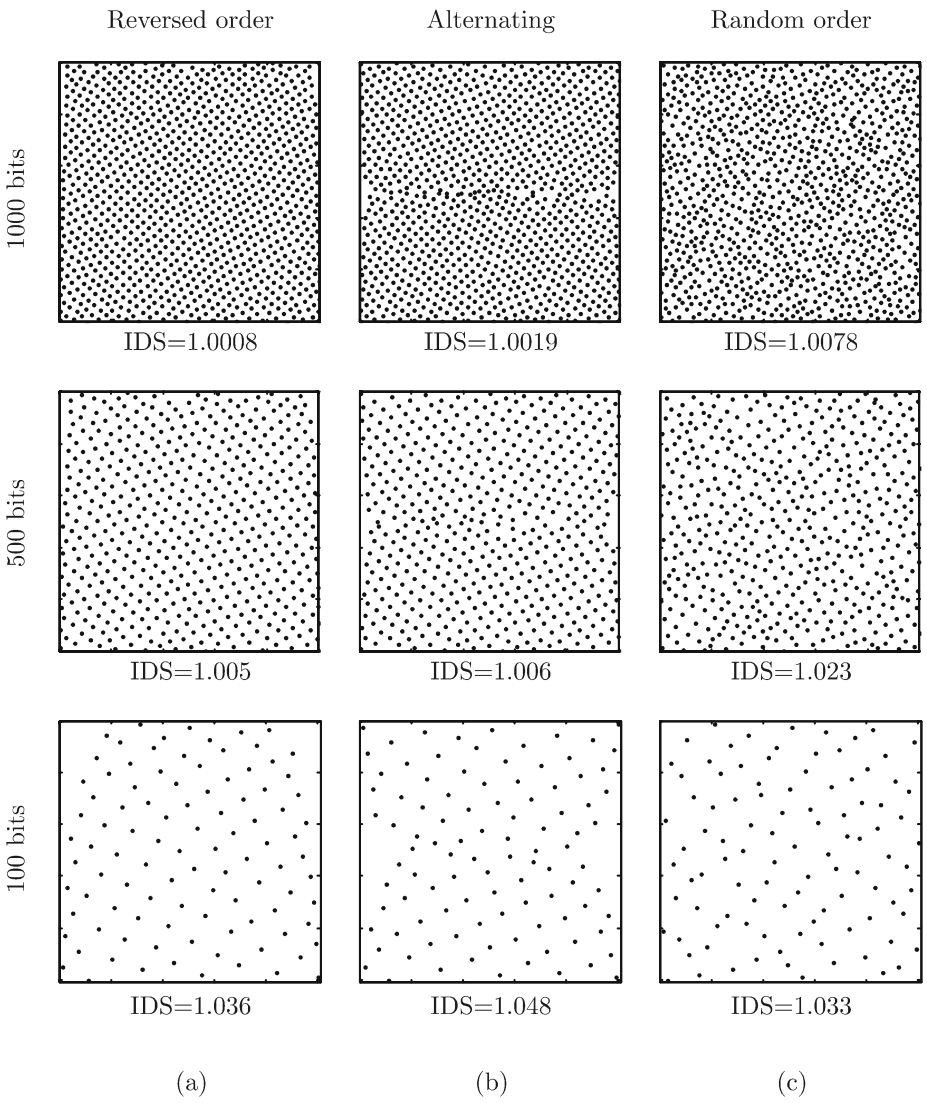


Figure B.2: Permutation matrices and IDS-values for correlation designed interleavers, designed using three different design orders: (a) reversed order, *i.e.* from last to first, (b) alternatingly last and first, and (c) random order. Three interleavers sizes are shown: 1000 bits (top), 500 bits (middle) and 100 bits (bottom).

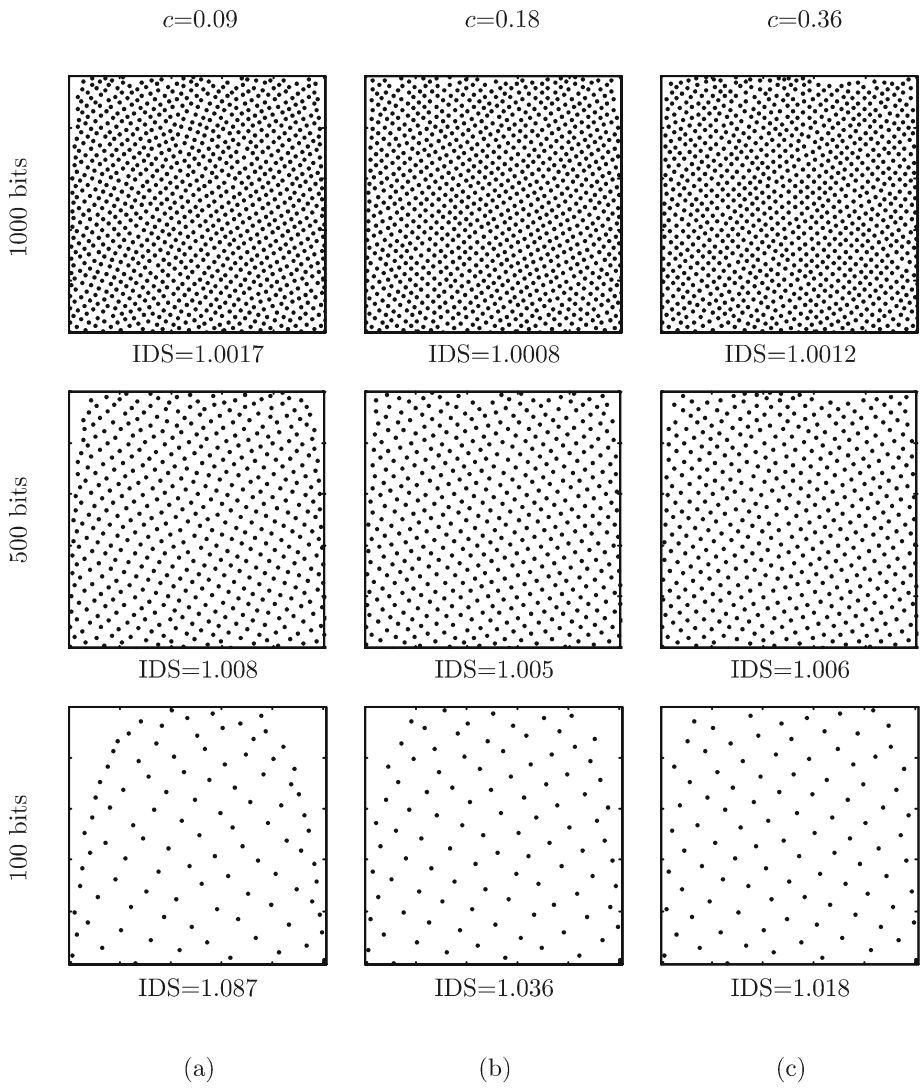


Figure B.3: Permutation matrices and IDS-values of correlation-designed interleavers of length 1000 (top), 500 (middle) and 100 bits (bottom), designed with various values of the correlation-decay coefficient c. (a) $c = 0.09$, (b) $c = 0.18$, and (c) $c = 0.36$.

i.e. with $d_{\text{design}} = 0$. In this section, the result of including a distance spectrum criterion is demonstrated; especially, the balance between the correlation criterion and a distance spectrum criterion is illustrated.

Typically, using a d_{design} that is too large will prevent the correlation criterion from influencing the design and, as a consequence, the resulting interleavers will be primarily designed based on distance spectrum considerations. Figure B.4 shows the result of using different d_{design} values, for 100-, 500- and 1000-bit interleavers. The value of d_{design} increases from left to right and at the left-hand side, the values are too low. They are too low since the distance spectrum criterion hardly influences the design at all. Thus, the criterion is not helpful in preventing mappings that generate low-weight codewords. On the other hand, as d_{design} is increased, eventually the distance spectrum criterion becomes too restrictive, so that there is no room for correlation considerations, as seen to the right in Figure B.4.

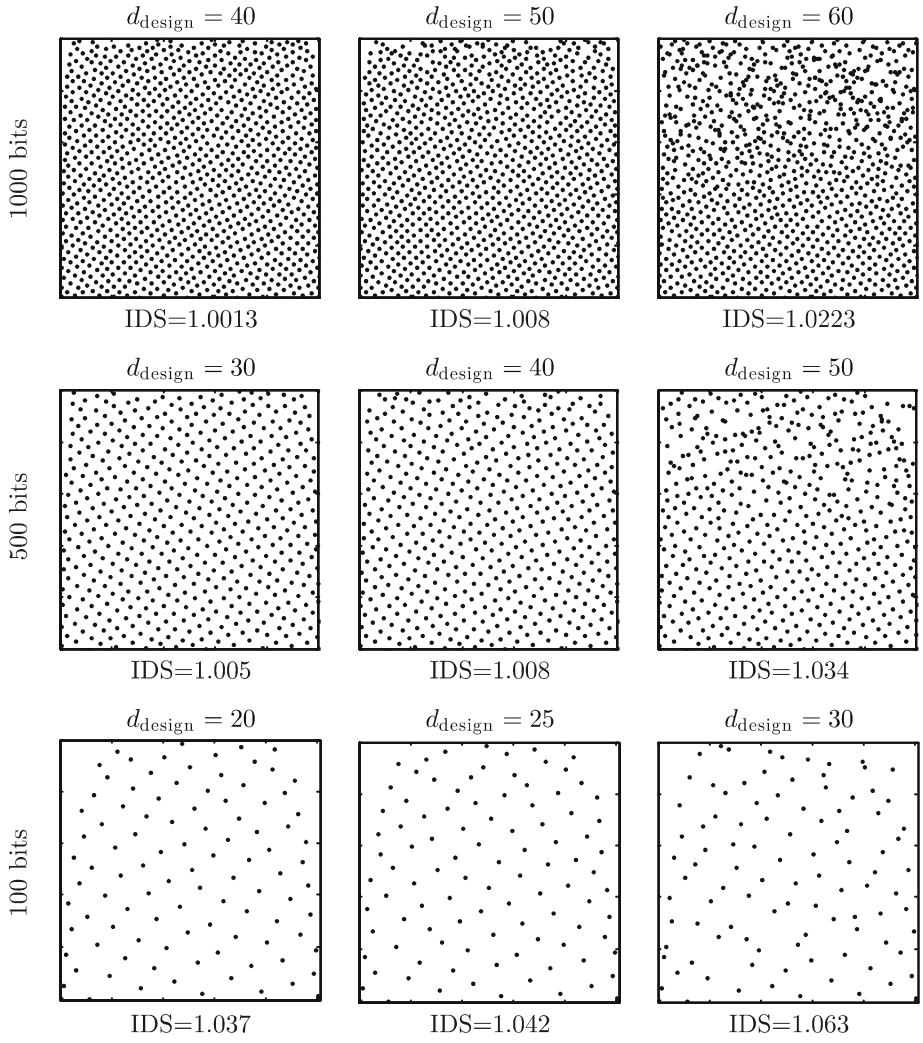


Figure B.4: Permutation matrices and IDS-values of correlation-designed interleavers of length 1000 (top), 500 (middle) and 100 (bottom) bits, designed with various values of the d_{design} parameter.

Bibliography

[1] J. Andersen and V. Zyablov. Interleaver design for turbo coding. In *International Symposium on Turbo Codes & Related Topics*. Brest, France, September 1997.

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

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

[4] A. S. Barbulescu and S. S. Pietrobon. Interleaver design for three dimensional turbo-codes. In *Proceedings of 1995 IEEE International Symposium on Information Theory*. Whistler, BC, Canada, September 1995.

[5] A. S. Barbulescu and S. S. Pietrobon. Terminating the trellis of turbo-codes in the same state. *Electronics Letters*, 31(1):22–23, January 1995.

[6] B. Bartolomé. *Utilisation des turbo codes pour un système de communications multimédia par satellite*. PhD thesis, Ecole Nationale Supérieure de Télécommunications, Paris, France, 1999.

[7] G. Battail. Pondération des symboles décodés par l'algorithme de Viterbi. *Annales Télécommunic*, 42(1–2):31–38, January 1987.

[8] S. Benedetto, R. Garello, and G. Montorsi. A search for good convolutional codes to be used in the construction of turbo codes. *IEEE Transactions on Communications*, 46(9):1101–1105, Sep. 1998.

[9] S. Benedetto and G. Montorsi. Average performance of parallel concatenated block codes. *IEEE Electronics Letters*, 31(3):156–158, February 1995.

[10] S. Benedetto and G. Montorsi. Unveiling turbo codes: Some results on parallel concatenated coding schemes. *IEEE Transactions on Information Theory*, 42(2):409–428, March 1996.

[11] E. R. Berlekamp. The technology of error correcting codes. *Proc. IEEE*, pages 564–593, May 1980.

[12] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding: Turbo-Codes. *IEEE Transactions on Communications*, 44(10):1261–1271, October 1996.

[13] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo Codes. In *Proc. 1993 IEEE International Conference on Communication (ICC)*, pages 1064–1070. Geneva, Switzerland, May 1993.

[14] W. J. Blackert, E. K. Hall, and S. G. Wilson. Turbo code termination and interleaver conditions. *Electronics Letters*, 31(24):2082–2084, November 1995.

[15] W. J. Blackert, E. K. Hall, and S. G. Wilson. An upper bound on turbo code free distance. In *IEEE Int. Conf. on Communications*, pages 957–961, 1996.

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

[17] G. C. Clark and J. B. Cain. *Error-Correction Coding for Digital Communications*. Plenum press, New York, 1988.

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

[19] S. Crozier. New high-spread high-distance interleavers for turbo-codes. In *20-th biennial Symposium on Communications*, pages 3–7. Kingston, Canada, May 2000.

[20] S. Crozier, P. Guinand, J. Lodge, and A. Hunt. Construction and performance of new tail-biting turbo codes. In *6-th International Workshop on Digital Signal Processing Techniques for Space Applications (DSP '98)*. Noordwijk, The Netherlands, September 1998.

[21] S. Crozier, J. Lodge, P. Guinand, and A. Hunt. Performance of turbo codes with relative prime and golden interleaving strategies. In *Sixth International Mobile Satellite Conference*, pages 268–275. Ottawa, Canada, June 1999.

[22] F. Daneshgaran and M. Mondin. Design of interleavers for turbo codes based on a cost function. In *International Symposium on Turbo Codes & Related Topics*, pages 255–258. Brest, France, September 1997.

[23] D. Divsalar, S. Dolinar, R. J. McEliece, and F. Pollara. Transfer function bounds on the performance of turbo codes. TDA Progress Report 42-122, JPL, Aug. 1995.

[24] D. Divsalar and R. McEliece. Effective free distance of turbo codes. *Electronic Letters*, 32(5):445, May 1996.

[25] D. Divsalar and F. Pollara. On the design of turbo codes. *TDA Progress Report 42-123*, pages 99–121, Nov. 1995.

[26] D. Divsalar and F. Pollara. Turbo codes for PCS applications. In *IEEE International Conference on Communications*. New York, USA, 1995.

[27] S. Dolinar and D. Divsalar. Weight distributions for turbo codes using random and nonrandom permutations. TDA progress report 42-122, Jet propulsion Lab., Pasadena, CA, August 1995.

[28] S. Dolinar, D. Divsalar, and F. Pollara. Turbo code performance as a function of code block size. In *International Symposium on Information Theory*. Boston, USA, August 1998.

[29] M. Eroz and A. R. Hammons. On the design of prunable interleavers for turbo codes. In *Vehicular Technology Conference*. Houston, USA, May 1999.

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

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

[32] V. Franz and J. Anderson. Reduced-search BCJR algorithms. In *IEEE International Symposium on Information Theory*, page 230, 1997.

[33] V. Franz and J. Anderson. Concatenated decoding with a reduced-search BCJR algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):186–195, Feb. 1998.

[34] P. Guinand and J. Lodge. Trellis termination for turbo encoders. In *17th Biennial Symp. On Communications*, pages 389–392. Kingston, Canada, May 1994.

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

[36] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, 42(2):429–445, March 1996.

[37] J. Hagenauer and L. Papke. Decoding "turbo"-codes with the soft output Viterbi algorithm (SOVA). In *IEEE International Symposium on Information Theory, 1994*, page 164, June 1994.

[38] M. Hattori, J. Murayama, and R. J. McEliece. Pseudo-random and self-terminating interleavers for turbo codes. In *Winter 1998 Information Theory Workshop*, pages 9–10. San Diego, USA, February 1998.

[39] A. Henriksson, J. Hokfelt, and O. Edfors. Evaluation of a trellis termination strategy for turbo codes in UMTS. Tdoc 432/98, ETSI SMG2 UMTS L1 Expert Group, Meeting number 7, Sweden, Oct 1998.

[40] A. Henriksson, J. Hokfelt, and O. Edfors. Evaluation of an interleaver design algorithm for turbo codes in UMTS. Tdoc 419/98, ETSI SMG2 UMTS L1 Expert Group, Meeting no 7, Sweden, Oct. 1998.

[41] A. Henriksson, J. Hokfelt, and O. Edfors. Interleaver design for turbo codes with reduced memory requirement. Tdoc 510/98, ETSI SMG2 UMTS L1 Expert Group, Meeting no 8, Nov. 1998.

[42] H. Herzberg. Multilevel turbo coding with short interleavers. *IEEE Journal on Selected Areas in Communications*, pages 303–309, February 1998.

[43] H. Herzberg and G. Poltyrev. The error probability of M-ary PSK block coded modulation schemes. *IEEE Transactions on Communications*, (4):427–433, April 1996.

[44] M. S. C. Ho, S. S. Pietrobon, and T. Giles. Improving the constituent codes of turbo encoders. In *IEEE Global Telecommunications Conference*, pages 3525–3529. Sydney, Australia, 1998.

[45] J. Hokfelt, O. Edfors, and T. Maseng. Assessing interleaver suitability for turbo codes. In *Nordic Radio Symposium*, pages 195–202. Saltsjöbaden, Sweden, October 1998.

[46] J. Hokfelt, O. Edfors, and T. Maseng. Interleaver design for turbo codes based on the performance of iterative decoding. In *IEEE International Conference on Communications*, pages 93–97. Vancouver, BC, Canada, June 1999.

[47] J. Hokfelt, O. Edfors, and T. Maseng. Interleaver structures for turbo codes with reduced storage memory requirement. In *Vehicular Technology Conference*, pages 1585–1589. Amsterdam, Netherlands, September 1999.

[48] J. Hokfelt, O. Edfors, and T. Maseng. A survey on trellis termination alternatives for turbo codes. In *IEEE Vehicular Technology Conference*, pages 2225–2229. Houston, Texas, USA, May 1999.

[49] J. Hokfelt, O. Edfors, and T. Maseng. Turbo codes: Interaction between trellis termination method and interleaver design. In *Radio Science and Communications Conference*, pages 571–576. Karlskrona, Sweden, June 1999.

[50] J. Hokfelt, O. Edfors, and T. Maseng. A turbo code interleaver design criterion based on the performance of iterative decoding. *IEEE Communications Letters*, In press.

[51] J. Hokfelt, C. F. Leanderson, O. Edfors, and T. Maseng. Distance spectrum of turbo codes using different trellis termination methods. In *International Symposium on Turbo codes & Related Topics*. Brest, France, September 2000.

[52] J. Hokfelt and T. Maseng. Optimizing the energy of different bitstreams of turbo codes. In *International Turbo Coding Seminar*, pages 59–63. Lund, Sweden, August 1996.

[53] J. Hokfelt and T. Maseng. Methodical interleaver design for turbo codes. In *International Symposium on Turbo Codes & Related Topics*, pages 212–215. Brest, France, September 1997.

[54] T. W. Hungerford. *Abstract Algebra: An Introduction*. Saunders College Publishing, Philadelphia, 1990.

[55] O. Joerssen and H. Meyr. Terminating the trellis of turbo codes. *Electronics Letters*, 30(16):1285–1286, August 1994.

[56] R. Johannesson. *Informationsteori – grundvalen för (tele-)kommunikation*. Studentliteratur, Lund, 1988. In Swedish.

[57] R. Johannesson and J. B. Anderson. A condition for feedback tailbiting convolutional encoders and a short list of allowed feedback polynomial. In *Proc. 1998 Princeton Conf. on Information Sciences and Systems*. Princeton, NJ, march 1998.

[58] R. Johannesson and K. Zigangirov. *Fundamentals of Convolutional Coding*. IEEE Press, 1999.

[59] P. Jung and M. M. Naßhan. Comprehensive comparison of turbo-code decoders. In *IEEE Vehicular Technology Conference*, pages 624–628. Chicago, USA, July 1995.

[60] S. M. Kay. *Fundamentals of Signal Processing: Detection Theory*, volume 2. Prentice-Hall, Inc., New Jersey, 1998.

[61] S. M. Kay. *Fundamentals of Signal Processing: Estimation Theory*, volume 1. Prentice-Hall, Inc., New Jersey, 1998.

[62] P. Komulainen and K. Pehkonen. Performance evaluation of superorthogonal turbo codes in AWGN and flat rayleigh fading channels. *IEEE Journal on Selected Areas in Communications*, 16(2):196–205, Feb. 1998.

[63] C. F. Leanderson. Low rate Turbo codes – design aspects and applications. Licentiate Thesis, Lund University, Lund, Sweden, Aug. 2000.

[64] C. F. Leanderson, J. Hokfelt, O. Edfors, and T. Maseng. A note on generator reuse in the component codes of low rate turbo codes. In *International Symposium on Turbo codes & Related Topics*. Brest, France, September 2000.

[65] C. F. Leanderson, J. Hokfelt, O. Edfors, and T. Maseng. On the design of low rate turbo codes. In *Vehicular Technology Conference*. Tokyo, Japan, May 2000.

[66] L. Lin and 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.

[67] S. Lin and D. J. Costello, JR. *Error Control Coding*. Prentice-Hall, Inc., New Jersey, 1983.

[68] J. Lodge, R. Young, P. Hoeher, and J. Hagenauer. Separable MAP "filters" for the decoding of product and concatenated codes. In *IEEE International Conference on Communications*, pages 1740–1745, May 1993.

[69] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North Holland, Amsterdam, 1977.

[70] M. Öberg. *Turbo Coding and Decoding for Signal Transmission and Recording Systems*. PhD thesis, University of California, San Diego, CA, USA, 2000.

[71] L. Perez, J. Seghers, and D. J. Costello, Jr. A distance spectrum interpretation of turbo codes. *IEEE Transactions on Information Theory*, 42(6):1698–1709, November 1996.

[72] R. Podemski, W. Holubowicz, and C. B. ans A. Glavieux. Distance spectrum of the turbo code. In *IEEE International Symposium on Information Theory*. Whistler, British Columbia, Canada, 1995.

[73] G. Poltyrev. Bounds on the decoding error probability of binary linear codes via their spectra. *IEEE Transactions on Information Theory*, (4):1284–1292, July 1994.

[74] J. G. Proakis. *Digital Communications*. McGraw-Hill, Inc., New York, 1989.

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

[76] P. Robertson. Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes. In *Proceedings Globecom '94*, pages 1298–1303, Dec. 1994.

[77] 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.

[78] P. Robertson, E. Villebrun, and P. Hoeher. A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain. In *IEEE International Conference on Communications*, pages 1009–1013. Seattle, USA, 1995.

[79] D. Ruyet and H. V. Thien. Design of cycle optimized interleavers for turbo codes. In *International Symposium on Turbo codes & Related Topics*. Brest, France, September 2000.

[80] H. R. Sadjapour, M. Salehi, N. J. A. Sloane, and G. Nebe. Interleaver design for short block length turbo codes. In *IEEE International Conference on Communications*. New Orleans, USA, June 2000.

[81] I. Sason and S. Shamai. Improved upper bounds on the ML decoding error probability of parallel and serial concatenated turbo codes via their ensemble distance spectrum. *IEEE Transactions on Information Theory*, (1):24–47, January 2000.

[82] S. Schäffler and J. Hagenauer. Soft decision MAP decoding of binary linear block codes via global optimization. In *IEEE International Symposium on Information Theory*, page 231, June 1997.

[83] C. Schlegel. *Trellis Coding*. IEEE Press, New York, 1997.

[84] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 (Part I) and 623–656 (Part II), 1948.

[85] C. E. Shannon. Probability of error for optimal codes in a gaussian channel. *Bell System Technical Journal*, 38:611–656, 1959.

[86] A. Shibutani, H. Suda, and F. Adachi. Multi-stage interleaver for turbo codes in DS-CDMA mobile radio. In *Asia-Pasific Conference on Communications*, November 1998.

[87] P. Ståhl, J. B. Anderson, and R. Johannesson. New tailbiting encoders. In *IEEE International Symposium on Information Theory*, 1998.

[88] N. A. Van Stralen, J. A. F. Ross, and J. B. Anderson. Tailbiting and decoding recursive systematic codes. *Electronics Letters*, 35(17):1461–1462, Aug. 1999.

[89] Y. Svirid. Weight distributions of turbo-codes. In *IEEE International Symposium on Information Theory*, page 38. Whistler, British Columbia, Canada, 1995.

[90] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello, Jr. A note on asymmetric Turbo-codes. *Communications Letters*, 3(3):2082–2084, March 1999.

[91] O. Y. Takeshita and D. J. Costello, Jr. New classes of algebraic interleavers for turbo-codes. In *IEEE International Symposium on Information Theory*, page 419. Cambridge, MA, USA, August 1998.

[92] G. Ungerboeck. Trellis-coded modulation with redundant signal sets. Part I: Introduction. *IEEE Communications magazine*, 25(2):5–11, February 1987.

[93] G. Ungerboeck. Trellis-coded modulation with redundant signal sets. Part II: State of the art. *IEEE Communications magazine*, 25(2):12–21, February 1987.

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

[95] Y.-P. Wang, R. Ramesh, A. Hassan, and H. Koorapaty. On MAP decoding for tail-biting convolutional codes. In *IEEE International Symposium on Information Theory*, page 225, June 1997.

[96] C. Weiß, C. Bettstetter, S. Riedel, and D. J. Costello, Jr. Turbo decoding with tail-biting trellises. In *1998 URSI International Symposium on Signals, Systems and Electronics*, pages 343–348, September 1998.

[97] N. Wiberg. *Codes and Decoding on General Graphs*. PhD thesis, Linköping University, Linköping, Sweden, 1996.

[98] N. Wiberg, H.-A. Loeliger, and R. Kötter. Codes and iterative decoding on general graphs. In *IEEE International Symposium on Information Theory*, page 468, September 1995.

Part II

Included Papers

Contributions

This part of the thesis consists of six papers, whose respective contributions to the research field are described below.

I *Methodical Interleaver Design for Turbo Codes.*
This paper presents an interleaver design algorithm based on Turbo code Hamming distance spectra. In essence, the algorithm strives to design interleavers for the highest possible minimum distance, with the lowest possible multiplicity. Interleavers produced with the presented algorithm yield Turbo codes with very good distance properties. However, the complexity of the algorithm limits its use to reasonably small interleavers, typically less than 500 bits.
Although the algorithm presented in this paper yields very competitive Turbo codes, it is observed that Turbo code performances do not always rank as anticipated by the distance spectra, especially after only a few number of decoding iterations. As a consequence, the performance of iterative decoding was investigated, with results reported in [Papers II-IV].

II *On the Convergence Rate of Iterative Decoding.*
In this paper, the influence of the interleaver on the convergence rate of iterative decoding is investigated. In particular, the approach of investigating correlation properties of encoder inputs and outputs, estimated using Monte Carlo simulations, are proposed. It is found that these correlation properties are clearly related to the convergence rate of iterative decoding. It is also concluded that it is possible to design interleavers that promote a faster decoding convergence.

III *Turbo Codes: Correlated Extrinsic Information and its Impact on Iterative Decoding Performance.*
This paper further investigates the correlation properties of the extrinsic information, and the influence on the iterative decoding performance. An analytical approximation of relevant correlation coefficients is proposed. With this approximation, it is possible to assess the correlation properties that a specific interleaver results in. Based on these observations, an *iterative decoding suitability* (IDS) measure is presented. The IDS-value

is a scalar that measures the correlation properties of an interleaver. The IDS-measure has later been adopted and further investigated by others, for example in [79, 80].

IV *A Turbo Code Interleaver Design Criterion Based on the Performance of Iterative Decoding.*
In this paper, an interleaver design criterion based on the performance of iterative decoding is proposed. The criterion uses the approximation of the correlation coefficients presented in [Paper III]. This paper is the first proposal[2] of an interleaver design criterion that targets the performance of iterative decoding. Since then, similar proposals have followed, for example in [19, 79, 80].

V *Interleaver Structures for Turbo Codes with Reduced Storage Memory Requirement.*
This paper deals with the implementation complexity of interleaver rules stored as lookup tables. Two interleaver structures are proposed, which decrease the amount of memory required for interleaver rule storage. The first structure is useful for a single interleaver, while the second is useful when a range of interleaver sizes are to be stored. Both structures offer memory savings of approximately 50%, with an overall saving of approximately 75%. Simulations indicate that these interleaver restrictions do not imply any significant loss in error-correcting performance.

VI *On the Theory and Performance of Trellis Termination Methods for Turbo Codes.*
This paper investigates different trellis termination methods for Turbo codes. A method for calculating the distance spectrum of the ensemble of Turbo codes using a specific trellis termination method is presented. Using this method, it is shown that the performance differences between different termination methods are small, except when *no termination* at all is used. Further, the issue of custom design of interleavers for different termination methods is addressed. It is shown that with proper interleaver design, the performance degradation associated with *no termination* can be significantly reduced, achieving performances comparable to codes using termination. At the writing of this thesis, the best non-terminated Turbo code in this paper is still under simulation. It has reached over 6 billion frames at $E_b/N_0 = 2$ dB with only one decoding error. There is still no evidence of an error floor. However, achieving statistical significance at these error rates will require several months of simulations.

[2] The contents of this paper was originally presented in [46].

Paper I

Methodical Interleaver Design for Turbo Codes

Johan Hokfelt and Torleiv Maseng

International Symposium on Turbo Codes & Related Topics
Brest, France, September 1997

Methodical Interleaver Design for Turbo Codes

Johan Hokfelt and Torleiv Maseng

Abstract

An interleaver design algorithm for parallel concatenated coding schemes which utilizes the weight distribution as design criterion is presented. The qualities of the algorithm is demonstrated by comparing the weight distributions of various turbo codes using a number of previously proposed interleaver techniques (pseudo random, non uniform, block and block helical simile).

1 Introduction

The error correcting capability of a linear code is closely related to the Hamming weights of its set of codewords, *i.e.* the weight distribution of the code. In the turbo coding scheme proposed by Berrou et. al. [1], depicted in Figure1, the Hamming weights of the codewords are composed of three parts; the weight of the input word w and the weights of the parity words from each constituent encoder, z^1 and z^2. The objective when designing interleavers for such parallel concatenated coding systems is to match the parity sequences from the constituent encoders so that overall codewords with low Hamming weights (originating from non-zero input sequences) are avoided. It has previously been argued [2, 3] that it is primarily low weight input words (*i.e.* $w = 2, 3, 4, ...$) that result in low weight overall codewords. The way in which the parallel concatenated coding scheme responds to such input words is thus critical for the code performance. This paper presents an interleaver design algorithm which uses the weight distribution resulting from these low weight input words as design criterion.

2 Role of Interleaver

In general, the error correcting capability of a linear code is related to the weights of its set of codewords. The role of the interleaver in a turbo coding scheme is to map the set of parity words from the second constituent encoder (set C^2) to the combined set of words consisting of the input words together with their corresponding parity words from the first constituent encoder (set

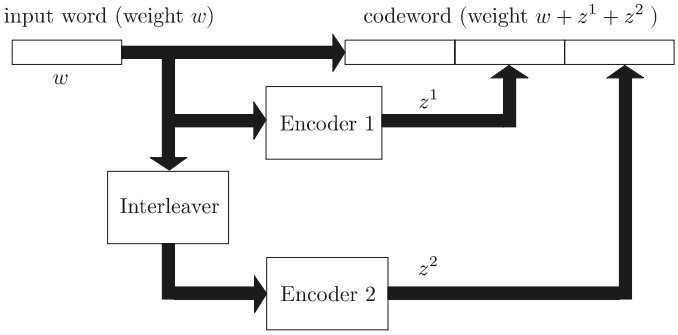


Figure 1: Basic principle of studied Turbo coding scheme

C^1). If no precautions are taken, an element from set C^1 with low Hamming weight ($w + z^1$) might be combined with an element from set C^2 which also has low weight (z^2). The resulting codeword will therefore have a low overall weight and a deteriorating influence on code performance. These unfavorable mappings may for example occur when a *pseudo-random interleaver* is used, which maps words from C^1 and C^2 in a seemingly random fashion (with certain limitations). The *block interleaver*, in which bits are written in rows and read by columns, also exhibit such unfavorable mappings.

The constituent encoders in Figure 1 are recursive convolutional encoders, examples of which are shown in Figure 2. For every feedback generator polynomial g1, there exists input words that produce constituent parity words of low weight. For g_1=$15_{(\mathrm{oct})}$ as in Figure 2, examples of such words are 00...0100000010...00, 00...01000110...00 and 00...011010...00. These input words all have one property in common; they return the recursive encoders to their original state (*i.e.* the zero state). Words with this property are said to be *self terminating*. This property is essential since an input sequence which is not self terminating will produce a parity sequence of infinite weight (if no truncation is performed), and will therefore not produce an overall codeword of low weight. In practice, the input sequences are truncated to the size of the interleaver, which results in *interleaver edge effects*. The consequence of the truncation is that also non self terminating input words might produce parity sequences of low weight, and thereby degrade code performance.

Interleaver structures that strives to avoid mappings of self terminating input words to other self terminating words has been proposed, for example the *non-uniform interleaver* [4] and the *block helical simile interleaver* [5]. These interleavers are based on the ordinary block interleaver, but the output bits are read in some modified fashion instead of just column by column. The basic principle of the block helical simile interleaver is to read bits diagonally instead of column by column, with certain limitations on the number of rows

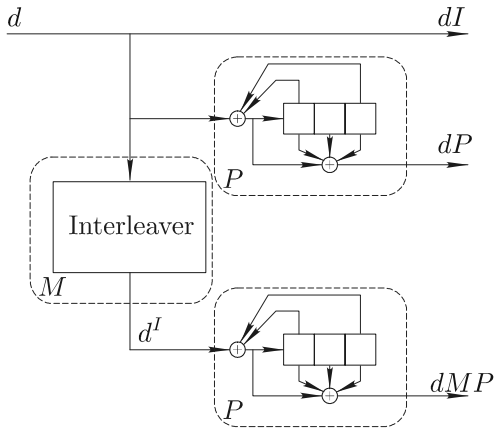


Figure 2: A Turbo encoder with feedback polynomials $15_{(\mathrm{oct})}$ and $17_{(\mathrm{oct})}$ respectively. No puncturing is performed.

and columns in the matrix. The resulting interleaver is a so called *simile* interleaver, which terminates both constituent encoders in the same state. The non-uniform interleaver also read output bits diagonally, but with certain row- and column jumps between each reading.

These more sophisticated interleavers succeed in avoiding several unfavorable mappings. They do however exhibit residual low weight codewords, and there is reason to believe that there exists interleavers which results in still better weight distributions.

2.1 Block code form of turbo codes

The interleaver design algorithm is best explained when considering the turbo coding scheme in its equivalent block form. This text will follow the notations used in [6], which are recited here in brief.

The generator matrix of a parallel concatenated code is obtained by merging the generator matrices of each of the concatenated branches. The generator matrices of the constituent encoders and the interleaver matrix are denoted P and M respectively, both of size $N \times N$, where N is the size of the interleaver. M can be any permutation matrix, *i.e.* any matrix having exactly one 1 in each row and column, all other elements being 0. For the systematic branch, the generator matrix is simply the identity matrix I, also of size $N \times N$. See Figure 2 for reference. The generator matrix of the parallel concatenated code, G_T, is then given by

$$G_T = [I|P|MP]\,,$$

where it is assumed that the constituent encoders are identical.

$$P = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3: Generator matrix P and interleaver matrix M for given example of Turbo encoder.

Example 1 *A turbo encoder with blocklength 12 with feedback and parity polynomials $15_{(oct)}$ and $17_{(oct)}$ respectively, as depicted in Figure 2, have constituent generator and interleaver matrices as shown in Figure 3. A 1 in row i and column j in the interleaver matrix results in that bit number i in the input word will be bit number j in the interleaved word.*

3 Design Algorithm

The basic idea of the algorithm is to make sure that at least one of the parity words have "non-low" weight whenever the weight of the information word is low. This ensures that the total weight of the codeword is kept "non-low", which has the desired impact on the weight distribution of the overall turbo code. The design process consists of row by row choosing the positions of the 1's in the interleaver matrix M, whose elements are all zero from the beginning. Once a 1 has been placed at a given position in a row, it is possible to locate certain "bad" positions for the remaining rows, positions that would result in a low weight codeword if chosen. Keeping track of all of the resulting codeword Hamming weights of these bad positions in a *cost matrix* (denoted J) makes it feasible to avoid undesired mappings between the sets C^1 and C^2. Due to the fast increase in complexity encountered when taking input words of larger and larger weight into account, and to the fact that it is primarily low weight input words that produce codewords with low weight, the algorithm has been limited to take input words with weight smaller or equal to 4 into account.

Since the latter rows in the interleaver matrix have fewer and fewer columns to choose among, its is plausible that a position resulting in a low weight codeword is forced to be chosen towards the end of the design process. However, as the cost matrix contains the lowest weight codeword that would exist in the turbo code for every choice of position in the interleaver matrix (given the positions already chosen), it is possible to locate appropriate swapping

positions. After performing such a swap the entire cost matrix is updated which allows for identification of new swapping alternatives, a process that is continued until no further swaps can be found. The algorithm is clarified by an example.

3.1 Algorithm example

Assume we are to design an interleaver of size 16 bits, to be used in the turbo encoder depicted in Figure 2. The algorithm is initialized with an all-zero interleaver matrix and an empty cost matrix.

A column in which to place the 1 in the first row in the interleaver matrix is chosen randomly, after which an update of the cost matrix is performed. Figure 4a shows the cost matrix after this step, where the first 1 was placed in the $14th$ column, indicated by a circle in the cost matrix. The cost matrix tells us that if a future 1 is placed in position (8, 7), a codeword with Hamming weight 14 will exist in the overall turbo code. This low weight codeword would arise because the corresponding interleaver would permute the input word $d = 1000000100000000$ to the interleaved word $d^I = 0000001000000100$, which are both self terminating sequences for the constituent encoders in Figure 2, and thus produce low weight parity words. When interleaver edge effects are taken into account, the cost matrix shown in Figure 4c arises. In this small size interleaver the edge effect impacts a major portion of the cost matrix, which is not the case for large interleavers. Interleaver edge effects are not taken into account in the rest of this example, since they conceal the role of the self terminating sequences.

The next step is to place a 1 in the second row of the interleaver matrix. The eligible columns are the still unoccupied ones with the highest calculated Hamming weights in the cost matrix. Since the second row of the cost matrix is still empty, we choose among all the columns not yet occupied. Figure 4b shows the updated cost matrix after choosing the third column in the second row in the interleaver matrix.

Figure 5 shows the cost matrix after all the positions in the interleaver has been chosen. The very last position to be chosen where forced to be in column 6, which resulted in a codeword with Hamming weight 11. This codeword is a result of the positions in the interleaver matrix indicated by the thicker circles, at rows 13, 14 and 16. These positions correspond to the permutation of the input word $d = 0000000000001101$ to $d^I = 1000110000000000$, which both are self terminating and thus result in an overall codeword of low weight. However, this low weight codeword can be avoided by performing the swap indicated by the shaded circles. By moving any of the thicker circles the above permutation is eliminated, in the suggested case replaced by $d^I = 1000100100000000$ which is no longer a self terminating word.

After performing a suitable swap an update of the cost matrix is performed, so that further swap candidates can be identified. The updated cost matrix

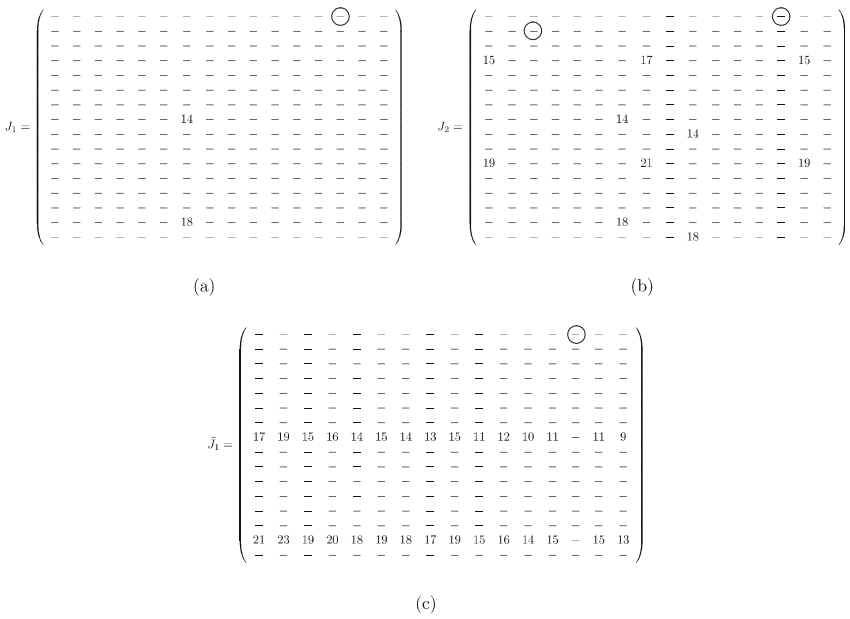


Figure 4: (a) and (b): Cost matrices after choosing positions for the first and second row in the interleaver matrix. Chosen positions are indicated by circles. (c): Same as (a), except that interleaver edge effects are taken into account.

is shown in Figure 6, where it can be seen that the weight of the minimum weight codeword has increased from 11 to 15. Again, further improvements through swaps of positions are investigated, for example as again indicated by grey circles. However, updating the cost matrix after performing these swaps reveals a minimum distance of 14, *i.e.* a deterioration of the weight distribution. This is a result of the fact that the calculation of the cost matrix is carried out for specific positions of the 1's in the interleaver matrix; if any of these 1's are moved, the cost matrix is no longer completely valid. This is why the existence of the codewords with Hamming weight 14 were not discovered until after the update of the cost matrix. In this particular example no further suitable swaps were found, and the design process is terminated. The minimum weight of the resulting turbo code is 15 (for input words ranging from 2 to 4). However, in this example the interleaver edge effects were not taken into account, which may very well result in lower weight codewords in this particular case. Edge effects are taken into account in the full implementation of the algorithm.

$$J_{16} = \begin{pmatrix}
11 & 14 & 14 & 10 & 14 & 11 & 16 & 11 & 13 & 10 & 14 & 16 & 11 & 16 & 15 & 13 \\
15 & 16 & 15 & 13 & 13 & 14 & 14 & 11 & 10 & 14 & 11 & 14 & 14 & 14 & 11 & 14 \\
14 & 14 & 15 & 12 & 11 & 11 & 11 & 13 & 14 & 15 & 14 & 11 & 11 & 11 & 16 & 12 \\
15 & 15 & 14 & 12 & 11 & 10 & 16 & 13 & 13 & 11 & 11 & 11 & 8 & 16 & 14 & 15 \\
14 & 16 & 12 & 12 & 11 & 12 & 8 & 14 & 14 & 11 & 11 & 11 & 10 & 10 & 14 & 16 \\
13 & 15 & 15 & 10 & 14 & 11 & 14 & 11 & 15 & 13 & 12 & 12 & 11 & 8 & 11 & 15 \\
11 & 11 & 12 & 17 & 11 & 11 & 14 & 10 & 11 & 12 & 15 & 13 & 11 & 14 & 10 & 11 \\
11 & 14 & 10 & 11 & 16 & 11 & 12 & 11 & 11 & 8 & 11 & 16 & 11 & 14 & 15 & 11 \\
14 & 14 & 13 & 13 & 10 & 11 & 10 & 11 & 12 & 10 & 11 & 12 & 12 & 10 & 11 & 16 \\
13 & 11 & 12 & 14 & 11 & 11 & 11 & 11 & 13 & 12 & 13 & 11 & 11 & 11 & 15 & 16 \\
15 & 13 & 13 & 16 & 12 & 11 & 13 & 13 & 11 & 11 & 11 & 12 & 10 & 11 & 14 & 13 \\
16 & 16 & 13 & 15 & 12 & 16 & 11 & 16 & 14 & 11 & 11 & 12 & 12 & 13 & 18 & 18 \\
11 & 12 & 15 & 14 & 12 & 11 & 10 & 11 & 12 & 13 & 12 & 11 & 11 & 10 & 11 & 14 \\
13 & 11 & 14 & 14 & 11 & 12 & 15 & 13 & 11 & 12 & 15 & 11 & 11 & 14 & 12 & 14 \\
15 & 15 & 8 & 12 & 14 & 15 & 14 & 12 & 13 & 11 & 13 & 16 & 15 & 16 & 15 & 13 \\
16 & 16 & 15 & 16 & 12 & 11 & 15 & 15 & 14 & 12 & 14 & 14 & 15 & 15 & 15 & 17
\end{pmatrix}$$

Figure 5: Cost matrix when every position in the interleaver matrix has been chosen. Thicker circles marks positions that together results in a low weight codeword, which can be eliminated by performing the swap indicated by the grey circles.

4 Results

The described algorithm was evaluated by designing a 105 bit interleaver, to be used with the component encoders depicted in Figure 2 (g_1=$15_{(\mathrm{oct})}$, g_2=$17_{(\mathrm{oct})}$). The designed interleaver were compared to other interleavers by means of their weight distributions for input words of weight 6 and less (restricted by complexity considerations), which are reported in Table 1 and illustrated in Figure 7. It is apparent that the weight distribution achieved with the constructed interleaver is preferable to the other weight distributions. The block helical simile interleaver has however the advantage that both constituent encoders are terminated in the zero state. A description of the designed interleaver is given in Table 2.

$$J'_{16} = \begin{pmatrix}
11 & 14 & 14 & 10 & 14 & 11 & 16 & 11 & 13 & 10 & 14 & 16 & 11 & 16 & 15 & 13 \\
15 & 16 & 15 & 13 & 13 & 14 & 14 & 11 & 10 & 15 & 11 & 14 & 14 & 14 & 11 & 14 \\
14 & 14 & 15 & 12 & 11 & 11 & 11 & 13 & 14 & 15 & 14 & 11 & 11 & 11 & 16 & 12 \\
15 & 14 & 14 & 12 & 11 & 10 & 16 & 13 & 13 & 11 & 11 & 11 & 8 & 16 & 15 & 15 \\
14 & 16 & 12 & 12 & 11 & 13 & 8 & 12 & 14 & 11 & 11 & 11 & 10 & 10 & 14 & 16 \\
13 & 15 & 14 & 10 & 14 & 13 & 14 & 11 & 15 & 13 & 12 & 14 & 15 & 8 & 11 & 15 \\
11 & 13 & 11 & 12 & 11 & 11 & 16 & 10 & 11 & 11 & 15 & 13 & 11 & 14 & 10 & 11 \\
11 & 13 & 10 & 12 & 11 & 11 & 11 & 11 & 11 & 8 & 16 & 16 & 11 & 12 & 15 & 11 \\
14 & 14 & 13 & 13 & 12 & 11 & 10 & 11 & 12 & 13 & 11 & 12 & 12 & 10 & 11 & 16 \\
13 & 11 & 13 & 14 & 11 & 11 & 11 & 11 & 10 & 13 & 14 & 11 & 11 & 11 & 15 & 12 \\
11 & 14 & 13 & 16 & 13 & 12 & 13 & 11 & 14 & 11 & 11 & 11 & 10 & 11 & 15 & 14 \\
16 & 16 & 13 & 15 & 12 & 16 & 11 & 16 & 14 & 11 & 11 & 12 & 12 & 13 & 16 & 18 \\
17 & 12 & 15 & 14 & 12 & 11 & 10 & 11 & 12 & 14 & 12 & 12 & 13 & 10 & 11 & 14 \\
13 & 15 & 11 & 14 & 15 & 12 & 15 & 13 & 12 & 11 & 15 & 11 & 11 & 14 & 12 & 14 \\
15 & 11 & 8 & 15 & 11 & 14 & 13 & 12 & 11 & 12 & 16 & 16 & 15 & 14 & 15 & 13 \\
16 & 16 & 15 & 15 & 15 & 11 & 12 & 15 & 14 & 13 & 14 & 12 & 15 & 16 & 15 & 17
\end{pmatrix}$$

Figure 6: Cost matrix after performing the suggested swap, which increased the $d_{\min}$ from 11 to 15. A new swapping candidate is again indicated by grey circles.

Hamming distance:	10	11	12	13	14	15	16	17	18
1. Block (5×21)	1	3	1	2	4	6	2	2	5
2. Pseudo random (105 bits)	-	1	0	0	3	2	4	9	7
3. Non uniform (10×10)	-	-	-	-	1	0	1	8	2
4. Block Helical Simile (5×21)	-	-	-	-	-	-	-	-	-
5. New (105 bits)	-	-	-	-	-	-	-	-	-

	19	20	21	22	23	24	25	26	27	28	29
1.	12	11	16	191	45	123	74	268	301	474	603
2.	13	11	22	41	36	52	94	170	204	328	466
3.	7	1	31	55	31	29	69	155	181	304	409
4.	24	0	0	71	206	36	183	169	132	905	574
5.	-	-	-	8	20	36	87	122	177	241	286

Table 1: Weight distributions of the sample interleavers, for input words with weights ranging from 2 to 6.

[103 99 11 36 75 64 65 34 55 69 7 17 98 84 88 18 19 49 37 8 43
61 89 81 47 6 72 96 33 27 24 60 90 66 22 52 1 32 93 3 101 53
79 5 26 13 46 71 76 42 87 68 31 58 83 21 102 48 51 74 38 20 91
12 25 86 57 97 23 4 45 73 78 94 28 40 16 29 82 59 70 2 92 41
100 15 54 14 56 39 85 62 9 35 50 30 77 10 80 44 63 95 67 105 104]

Table 2: Representation of the designed 105 bit interleaver. The number in

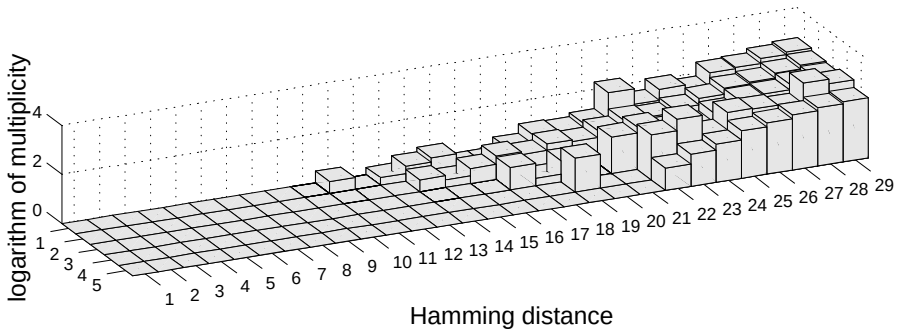


Figure 7: Weight distribution (logarithm of multiplicity) of 5 sample interleavers, for codewords stemming from input words with weight ranging from 2 to 6. The interleaver numbers (1–5) correspond to the interleavers reported in Table 1.

position i holds the position that input bit nr i will have in the interleaved sequence. For example will input bit number 3 be permuted to bit number 11 in the interleaved sequence.

5 Conclusions

A methodical interleaver design algorithm has been presented, and its advantages in terms of resulting weight distribution for a turbo code has been confirmed by constructing a 105 bit interleaver. The size were chosen to match the block helical interleaver, which were the second best interleaver considered. The described algorithm has no limitations on the interleaver size, which is the case for the block helical interleaver. The disadvantages of the algorithm is its rapid increase in complexity as the interleaver size grows. It is therefore believed that the algorithm is primarily significant for shorter interleaver sizes, suitable for *e.g.* speech communication.

References

[1] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-Codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, October 1996.

[2] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations." TDA progress report 42-122, Jet propulsion Lab., Pasadena, CA, August 1995.

[3] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, pp. 409–428, March 1996.

[4] C. Berrou and A. Glavieux, "Turbo-codes: General principles and applications," in *6th Int. Tirrenia Workshop on Digital Communications*, pp. 215–226, Pisa, Italy, September 1993.

[5] A. S. Barbulescu and S. S. Pietrobon, "Terminating the trellis of turbo-codes in the same state," *Electronics Letters*, vol. 31, pp. 22–23, January 1995.

[6] W. J. Blackert, E. K. Hall, and S. G. Wilson, "An upper bound on turbo code free distance," in *IEEE International Conference on Communications*, pp. 957–961, Dallas, TX, USA, 1996.

Paper II

On the Convergence Rate of Iterative Decoding

Johan Hokfelt and Torleiv Maseng

IEEE Communications Theory Mini-Conference
Sydney, Australia, November 1998

On the Convergence Rate of Iterative Decoding

Johan Hokfelt and Torleiv Maseng

Abstract

The convergence rate of iterative decoding of turbo codes for various interleaver strategies is studied. It is found that the convergence rate is related to the correlation between the extrinsic inputs and outputs of the component decoders in the iterative decoding scheme. These correlation properties are in turn dependent on the specific interleaver used, which explains why the interleaver influence the convergence rate. It is found that it is possible to design interleavers which achieve a faster convergence of the iterative decoding than achieved using a variety of other commonly used interleaver strategies.

1 Introduction

Since the introduction of turbo codes, quite an effort has been made to find good interleaver design strategies. Most of these efforts have been directed towards finding interleavers which results in codes with better weight distribution, *e.g.* [1, 2, 3, 4]. There are also design strategies that include termination on the trellis of the second encoder, *e.g.* [5]. Few of the proposed strategies employs the fact that turbo codes are decoded iteratively, and that the weight distribution of a code is a sufficient design criterion only for codes which are decoded with a maximum likelihood decoding algorithm. In order to achieve the best performance for turbo codes it is necessary to optimize with respect to the behavior of the iterative decoding as well.

In this paper the interleaver influence on the convergence rate of the iterative decoding is investigated. This investigation is based upon input-output correlation properties of the constituent decoders. It turns out that these correlation properties affect the performance of iterative decoding, both in terms of convergence rate and degree of non-maximum likelihood decisions. It is furthermore exemplified that it is possible to design interleavers which improve the convergence rate *and* the error correcting capabilities of turbo codes, by taking the correlation properties into account in the interleaver design.

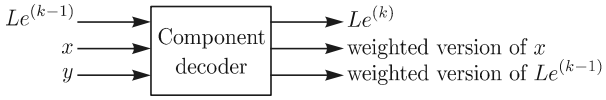


Figure 1: Inputs and outputs of component decoder performing decoding step k.

2 Decoder Correlation Properties

2.1 The First Decoding Step

This section treats the correlation properties between the inputs and outputs of a single decoder producing *a posteriori probabilities*. The component decoders considered have three inputs and three outputs as depicted in Figure 1. The three inputs are the extrinsic outputs from decoding step $k-1$, $Le^{(k-1)}$, the systematic inputs, x, and the parity inputs, y. The received sequences x and y have been transmitted on an additive white Gaussian noise channel. The three outputs are the extrinsic outputs, $Le^{(k)}$, a weighted version of the systematic inputs and a weighted version of the extrinsic inputs [6]. The decoder decision variables are the sum of the three outputs.

In the iterative decoding scheme, the sequence of extrinsic outputs are passed on to the next component decoder after a permutation defined by the interleaver. The interleaver is defined by N elements, $d(m), m = 0, 1, \ldots, N-1$, which point to the position in the original sequence which is to be permuted to position m in the interleaved sequence. The deinterleaver is correspondingly defined by the elements $d^{-1}(m), m = 0, 1, \ldots, N-1$. The size of the interleaver is N, which is also the length of all the input and output sequences to the component decoders.

Since it is only the extrinsic part of the decoder outputs that is passed on to the next decoder in the iterative decoding scheme, this presentation is focused on the correlation between these extrinsic outputs and the decoder input sequences. Let $\rho_{Le_i^{(k)},x_j}$ denote the correlation coefficient between extrinsic output i after decoding step k and systematic input j, *i.e.*

$$\rho_{Le_i^{(k)},x_j} = \frac{\mathrm{Cov}[Le_i^{(k)}, x_j]}{\sqrt{\mathrm{Var}[Le_i^{(k)}]\mathrm{Var}[x_j]}}. \tag{1}$$

The above entity can be estimated empirically by repeatedly transmitting the same codeword. Example of such estimations after the first decoding step for a recursive convolutional encoder with generator polynomials $(g_{feedback}, g_{parity}) = (15_{oct}, 17_{oct})$ are shown in Figure 2. The decay of $\rho_{Le_i^{(1)},x_j}$ as $|j-i|$ grows is roughly exponential, as indicated in Figure 2. The rate of decay is weakly dependent on the number of memory elements in the encoder: additional memory

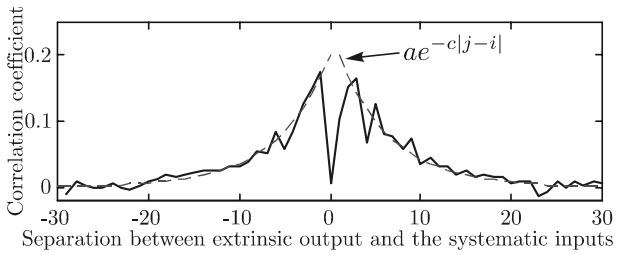


Figure 2: Empirically found correlation coefficients $\rho_{Le_i^{(1)},x_j}$ between extrinsic output i and systematic inputs $j = i - 30, \ldots, i + 30$.

elements result in slightly lower rate of decay.

Note that $\rho_{Le_i^{(k)},x_i} = 0$. This is a result of the implementation of the component decoders, which have a weighted version of the systematic input as output. Some decoder implementations include the weighted systematic output in the extrinsic output, and for those implementations there would be a strong correlation between extrinsic output i and systematic input i, *i.e.* $\rho_{Le_i^{(k)},x_i} \neq 0$.

2.2 The Second Decoding Step

The difference between the first and second decoding step is the presence of the extrinsic inputs. Without these, each extrinsic output from the second decoder would be correlated to the systematic input sequence in the same way as after the first decoder, expect for the permutation of the input sequence performed by the interleaver.

The inputs to the second decoder in the vicinity of time i are the sequence of triplets $\ldots$, $\{Le^{(1)}_{d(i-1)}, x_{d(i-1)}, y_{2,i-1}\}$, $\{Le^{(1)}_{d(i)}, x_{d(i)}, y_{2,i}\}$, $\{Le^{(1)}_{d(i+1)}, x_{d(i+1)}, y_{2,i+1}\}$ etc. As in the case of the first decoder, extrinsic output i is correlated to the permuted systematic inputs $x_{d(i\pm1)}$, $x_{d(i\pm2)}$, $x_{d(i\pm3)}$, etc. in an exponentially decaying fashion. In addition, $Le_i^{(2)}$ will also be correlated to the extrinsic inputs $Le^{(1)}_{d(i\pm1)}$, $Le^{(1)}_{d(i\pm2)}$, $Le^{(1)}_{d(i\pm3)}$, etc. in a similar fashion. These extrinsic inputs are in turn correlated to the systematic sequence according to Figure 2. They will thus provide extrinsic output $Le_i^{(2)}$ with correlation to parts of the systematic sequence to which the second component code itself is not capable of providing. Examples of the correlation coefficients $\rho_{Le^{(2)}_{250},x_j}$, $j = 0, 1, \ldots, 499$, for a rate1/3 turbo code composed by a block interleaver and the same generator polynomials as before is shown in Figure 3b. The black line shows the correlation between extrinsic output 250 and the original systematic sequence, while the grey line shows the correlation to the permuted systematic sequence. The latter illustrates the high correlation to the permuted systematic inputs in

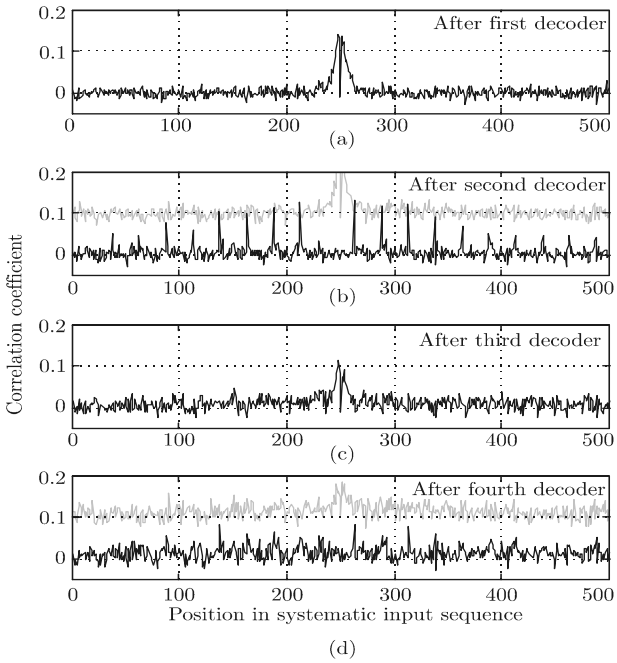


Figure 3: Empirically found correlation coefficients between extrinsic output 250 and the systematic sequence after 1, 2, 3 and 4 decoding steps. The grey lines are offset by 0.1 in the plots in order to not visually interfere with the black lines.

the vicinity of bit 250, *i.e.* $x_{d(i\pm1)}$, $x_{d(i\pm2)}$, $x_{d(i\pm3)}$, etc. Similarly, the dark line illustrates how the extrinsic inputs to the second decoder provide information about parts of the systematic sequence which the second component code itself is not capable of providing. This is seen as a slight increase in the correlation coefficients in the vicinity of each spike. The grey line is offset by 0.1 in order to not visually interfere with the black line.

2.3 Further Decoding Steps

The reasoning above applies also to the decoding steps to follow. The difference compared to the second decoding step is again in the extrinsic inputs. The extrinsic inputs to the second decoder were found to be roughly exponentially correlated to the systematic sequence. The previous subsection showed that the extrinsic outputs from the second decoder, which serves as inputs to the third, are correlated to a larger part of the systematic sequence. This will allow each new extrinsic output from the third decoder to be correlated to an even

larger part of the systematic sequence. This is verified in Figure 3c, showing the correlation coefficients after the third decoding step. The peak in the vicinity of bit 250 is slightly lower than it was after the second decoder, while the correlation to parts further away from bit 250 has increased. This tendency will proceed as further decoding steps are carried out, which is indicated in Figure 3d, showing the correlation coefficients after the fourth decoding step.

3 Convergence

When the iterative decoding converges, each extrinsic output $Le_i^{(k)}$ will become more and more correlated to the sequence of extrinsic inputs, $Le^{(k-1)}$, and less and less correlated to the systematic sequence x. This is a natural consequence of the concept of convergence; the entities being passed around in the loop should after convergence not alter significantly when further decoding steps are carried out. Therefore the correlation between the extrinsic outputs and inputs will become high, while the correlation to the systematic and parity sequences becomes low. Thus it is natural to investigate how the correlation between the extrinsic outputs and inputs develops as the iterative decoding proceeds. Let $\rho_{Le_i^{(k)},Le_j^{(k-1)}}$ denote the correlation coefficient between extrinsic output i after decoding step k and extrinsic output j after decoding step $k-1$. The extrinsic outputs are ordered in the order they leave each decoder, *i.e.* for odd k's in the original order and for even k's in permuted order. After the first decoder $\rho_{Le_i^{(1)},Le_j^{(0)}} = 0$, since $Le_j^{(0)} = 0, \forall j$. Extrinsic output i from the second decoder will be mainly correlated to the extrinsic outputs from the first decoder which serve as inputs in the vicinity of bit i. However, as in the case as for $\rho_{Le_i^{(1)},x_j}$, extrinsic output $Le_i^{(2)}$ will not be correlated to extrinsic input $Le_{d(i)}^{(1)}$, for the same reasons as given for $\rho_{Le_i^{(1)},x_i}$ earlier. There will however be a decaying correlation to the extrinsic inputs $Le_{d(i\pm1)}^{(1)}$, $Le_{d(i\pm2)}^{(1)}$, $Le_{d(i\pm3)}^{(1)}$, etc. Each of these inputs are in themselves correlated to their neighbors, *i.e.* $Le_{d(i+1)}^{(1)}$ is correlated to $Le_{d(i+1)\pm1}^{(1)}$, $Le_{d(i+1)\pm2}^{(1)}$, $Le_{d(i+1)\pm3}^{(1)}$, etc. This follows from the fact that $Le_{d(i+1)}^{(1)}$ and $Le_{d(i+1)+1}^{(1)}$ are both correlated to the same part of the channel inputs, *i.e.* the sequences x and y. Thus, they will also be correlated to each other. A consequence of this is that since extrinsic output $Le_i^{(2)}$ is correlated to $Le_{d(i+1)}^{(1)}$, it will also be correlated to $Le_{d(i+1)\pm1}^{(1)}$, $Le_{d(i+1)\pm2}^{(1)}$, $Le_{d(i+1)\pm3}^{(1)}$, etc.

It is obvious that the choice of interleaver will influence the correlation properties of the extrinsic output from the second decoder, since the interleaver defines the values of $d(m)$, $m = 0, 1, \ldots, N-1$. To clarify this influence, we present a comparison of four different interleavers, all of size 500 bits: a block interleaver, a pseudo-random interleaver, an S-random interleaver [1], and finally an interleaver designed based on the discussed correlation proper-

	Block		**Random**	
m	$d(m)$	$d^{-1}(m)$	$d(m)$	$d^{-1}(m)$
246	138	410	382	270
247	163	430	61	185
248	188	450	332	87
249	213	470	130	314
250	238	490	493	376
251	263	11	46	175
252	288	31	403	3
253	313	51	393	197
254	338	71	282	384

	S-random		**Correlation design**	
m	$d(m)$	$d^{-1}(m)$	$d(m)$	$d^{-1}(m)$
246	429	160	261	261
247	326	260	104	104
248	386	18	168	168
249	368	232	48	48
250	232	400	354	354
251	170	305	409	409
252	146	323	291	291
253	341	36	202	202
254	127	77	22	22

Table 1: Interleaver and deinterelaver rules in the vicinity of bit 250 for the compared interleavers.

ties [7]. For these interleavers we compare the correlation between extrinsic output 250 and the sequence of extrinsic outputs from the previous decoder, *i.e.* $\rho_{Le_{250}^{(k)}, Le_j^{(k-1)}}, j = 0, 1, \ldots, 499$.

Study first the block interleaver, which has 20 rows and 25 columns. For this interleaver $d(250) = 238$, $d(250 \pm 1) = 213$ and 263 and so on, as shown in the first column for the block interleaver in Table 1. Following the reasoning in the previous paragraphs, we confirm in Figure 4a that $Le_{250}^{(2)}$ is not particularly correlated to $Le_{238}^{(1)}$, nor to the extrinsic outputs in the vicinity of $Le_{238}^{(1)}$. Furthermore, we note a high correlation to $Le_{213}^{(1)}$, $Le_{263}^{(1)}$, $Le_{188}^{(1)}$, $Le_{288}^{(1)}$, etc., and to the outputs in the vicinity of these.

Moving on to the correlation coefficients for the random interleaver, shown in Figure 4b, we observe the same behavior: low correlation to extrinsic output $Le_{493}^{(1)}$ ($d(250) = 493$) and high correlation to extrinsic outputs $Le_{d(250\pm1)}^{(1)}$, $Le_{d(250\pm2)}^{(1)}$, etc. Note the pronounced peak around extrinsic output $Le_{390}^{(1)}$. The origin of this peak is clear: Table 1 reveals that extrinsic outputs 382, 403

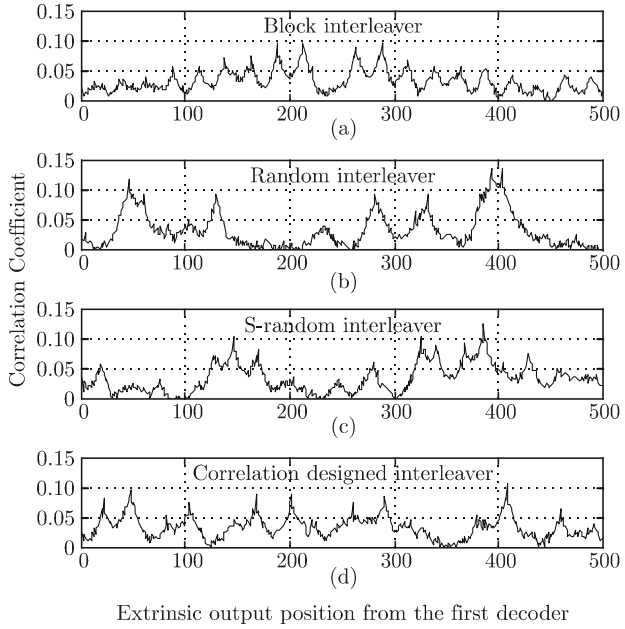


Figure 4: Correlation coefficients between extrinsic output 250 from the second decoder and the sequence of extrinsic outputs from the first decoder, $\rho_{Le_{250}^{(2)},Le_i^{(1)}}, i = 0, 1, \ldots, N-1$, for four different interleavers.

and 393 from the first decoder are used as inputs to the second decoder at positions 246, 252 and 253 respectively, *i.e.* all very close to position 250. It is thus consistent with the previous reasoning that extrinsic output 250 has high correlation to the extrinsic outputs from the first decoder around positions $370 - 410$. The effect seen here is the result of so called *short cycles*. Short cycles occur when two bits are close to each other both before and after interleaving. For example, bit 252 and 253 constitute a cycle of length 11, since $|252 - 253| + |d(252) - d(253)| = 11$.

The S-random interleaver is effectively removing all cycles shorter than a certain value, namely $S + 2$. Naturally there is an upper limit on how large S can be, dependent on the size of the interleaver. In [1], $S < \sqrt{N/2}$ was given as a reasonable choice for S, N being the interleaver size. In this interleaver comparison we used an S-interleaver with $S = 14$, which was the highest value of S for which we found a valid S-interleaver. The empirically found correlation coefficients for this interleaver are shown in Figure 4c. The main characteristics of these coefficients are rather similar to those of the random interleaver, except that there are no peaks on a distance smaller than 16, *i.e.* $S + 2$. It is again confirmed that the peaks appear at the positions given by $d(m)$ as shown in

Table 1.

The last compared interleaver is designed based on an algorithm described in [7]. The basic principle of this algorithm is to spread out the correlation peaks as much as possible, instead of just making sure that they are at least $S+2$ bits apart as in the case of the S-random interleaver. Furthermore, Table 1 shows that $d(m) = d^{-1}(m)$ for this particular interleaver, implying that the deinterleaving rule is identical with the interleaving rule. This property has no general impact the performance of the turbo code, but it has implementation advantages since it require only half the memory to store the interleaver/deinterleaver rules. The correlation coefficients $\rho_{Le_{250}^{(k)}, Le_j^{(k-1)}}, j = 0, 1, \ldots, N-1$, for this interleaver are shown in Figure 4d.

Figure 5 shows the same correlation coefficients as Figure 4 for decoding steps 2–7, marked on the right hand side of each plot. Observe first the correlation coefficients after the third decoding step. The extrinsic outputs from the second decoder serving as input close to bit 250 are shown in Table 1, in the columns headed $d^{-1}(m)$. For the block interleaver, for example, extrinsic output $Le_{490}^{(2)}$ serves as input at position 250, while $Le_{470}^{(2)}$ and $Le_{11}^{(2)}$ are inputs at positions 249 and 251 respectively. It is thus expected that extrinsic output $Le_{250}^{(3)}$ will have high correlation to the positions in the vicinity of 470, 450, 430, 410, ..., as well as 11, 31, 51, 71, etc. That this really is the case is seen in Figure 5a.

Consider the random interleaver, whose correlation coefficients are shown in Figure 5b. The set of correlation coefficients $\rho_{Le_{250}^{(3)}, Le_i^{(2)}}, i = 0, 1, \ldots, 499$ exhibit an extra high and wide peak near position 185. From Table 1 it is clear that this peak origins from the extrinsic outputs $Le_{185}^{(2)}$, $Le_{175}^{(2)}$ and $Le_{197}^{(2)}$, which serve as inputs at position $247, 251$ and 253 respectively. What is the drawback of having such a peak in the set of correlation coefficients? To answer this question, observe the correlation coefficients as the iterative decoding converges. For all compared interleavers the correlation plots become more and more smooth as the iterative decoding proceeds. It is thus reasonable to expect that the faster this smoothness is achieved, the faster will the iterative decoding converge. It would then be favorable to avoid unnecessarily high and wide peaks in the correlation coefficients, peaks which again are a result of short cycles created by the interleaver.

Both the S-random and the correlation designed interleaver avoid the short cycles which has been concluded to produce extra high and wide peaks in the correlation coefficients. The basic difference between the two interleavers are that while the S-random type guarantees that two peaks are separated by at least $S + 2$ bits, the correlation designed seeks to place peaks at positions with the lowest (previous) correlation. This is an attempt to make the set of correlation coefficients as smooth as possible, even after only two and three decoding steps. That this is somewhat achieved is seen in Figure 5c and d.

In order to compare of the smoothness of the correlation coefficients of the

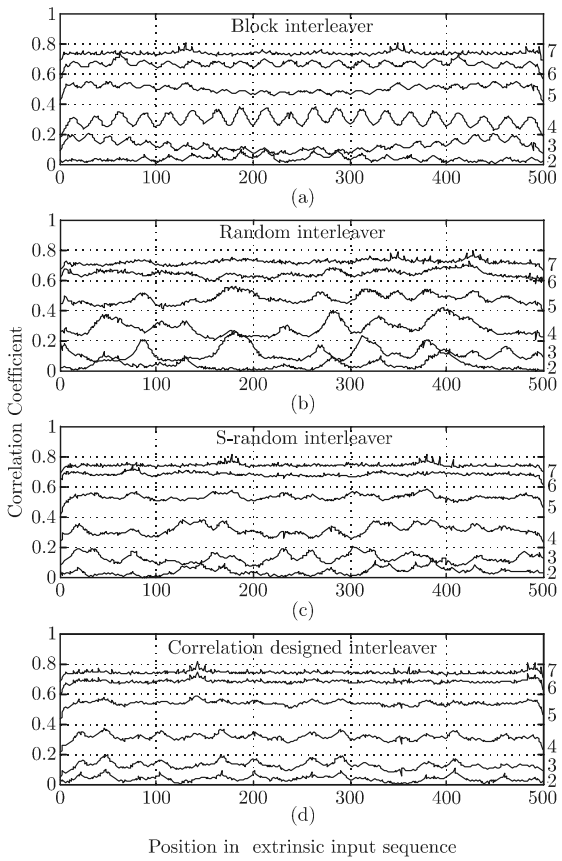


Figure 5: Correlation coefficients between extrinsic output 250 and the sequence of extrinsic inputs after 2–7 decoding steps, for four different interleavers.

studied interleavers, we compare the standard deviation of the correlation coefficients showed in Figure 5. These standard deviations, denoted $\sigma^{(k)}_{Le_{250}}$, are reported in Table 2 together with the percentage of successfully decoded frames for decoding steps 2–7. The percentage of successfully decoded frames is related to the number of correctly decoded frames after 30 decoding steps (15 iterations). We note a clear agreement between low values on the standard deviations and fast convergence. The interleavers resulting in the fastest convergence are, as expected, the S-random and correlation designed interleavers. Also shown in Table 2 are the frame error rates after fifteen decoding iterations and the minimum Hamming distances of the compared codes. Fortunately, the

	Block		Random		S-random		Correlation design	
k	$\sigma^{(k)}_{Le_{250}}$	Dec. frames	$\sigma^{(k)}_{Le_{250}}$	Dec. frames	$\sigma^{(k)}_{Le_{250}}$	Dec. frames	$\sigma^{(k)}_{Le_{250}}$	Dec. frames
2	0.018	2.5%	0.029	2.3%	0.024	2.6%	0.018	2.8%
3	0.034	48.2%	0.047	43.7%	0.030	50.0%	0.024	50.4%
4	0.038	84.2%	0.049	80.2%	0.032	85.1%	0.021	86.1%
5	0.024	96.2%	0.033	94.8%	0.020	97.3%	0.016	97.5%
6	0.020	98.5%	0.023	98.2%	0.012	99.2%	0.012	99.2%
7	0.012	99.4%	0.016	99.3%	0.012	99.8%	0.011	99.7%
FER	$10.4 \cdot 10^{-4}$		$12.4 \cdot 10^{-4}$		$1.2 \cdot 10^{-4}$		$0.7 \cdot 10^{-4}$	
$d_{\min}$	28		14		19		24	

Table 2: Comparison of $\sigma^{(k)}_{Le_{250}}$, k=2–7, with the percentage of successfully decoded frames (at $E_b/N_0 = 1.5dB$). Also listed are the final frame error rates (after 15 iterations) and minimum Hamming distances of the compared codes.

interleaver resulting in the fastest convergence also result in the code with the best error correcting performance. The rate of convergence for the different interleavers are also illustrated in Figure 6.

Note that the reported values in Table 2 are for extrinsic output 250 only. A complete investigation of the iterative decoding performance requires studying all of the extrinsic outputs, *i.e.* $\sigma^{(k)}_{Le_i}$ for $i = 0, 1, \ldots, N-1$, not only output 250 as exemplified here. However, due to the design rules of the block-, S-random and correlation based interleavers, these quantities are rather similar for the majority of the extrinsic outputs. This is however not necessarily the case for the pseudo-random interleaver, which exhibits larger variations in the standard deviations $\sigma^{(k)}_{Le_i}$. These variations stem from the fact that each position in the interleaver is involved in different number of short cycles, which of course also vary in lengths.

The 20×25 block interleaver used in the comparison has no cycles shorter than 21. Although these cycles are longer than the shortest cycles of the S-random interleaver, which has no cycles shorter than 16, the S-random interleaver converges faster. This may seem contradictive at first. However, the block interleaver has a very large number of cycles of length 21 while the S-random interleaver is probable to have only few of its short length cycles. Furthermore, the block interleaver has also a very large number of short *secondary cycles*, cycles which require more than one interleaving or deinterleaving to complete. These secondary cycles exist also for the other interleavers in the comparison, but in very small numbers. They are therefore not believed to have a significant influence on the iterative decoding performance of these interleavers. For the block interleaver however, there are approximately N secondary cycles of length 4, and even more of length $5, 6, 7$ etc. These cycles may be a part in the explanation of the slower convergence rate of the block

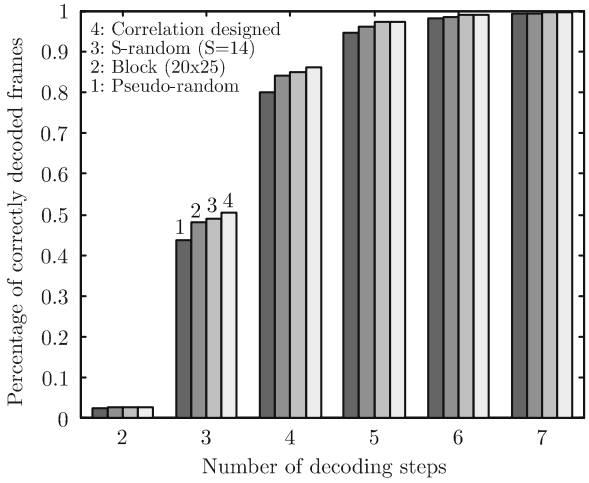


Figure 6: Percentage of correctly decoded frames as a function of the number of completed decoding steps ($E_b/N_0 = 1.5$ dB).

interleaver.

4 Conclusions

The interleaver influence on the convergence rate of iterative decoding of turbo codes has been investigated. The intuitively appealing approach of studying the correlation between the extrinsic inputs and outputs of the component decoders reveal interesting properties of the iterative decoding process. Specifically, the standard deviation of these correlation coefficients show upon a relationship with the convergence rate of the iterative decoding. Furthermore, it is found that it is possible to design interleavers which achieve a faster convergence than achieved with for example pseudo-random interleavers. The fastest converging interleavers found are S-random interleavers and interleavers designed with a related algorithm based on the discussed correlation properties [7]. These interleavers are also competitive in terms of error correcting performance.

References

[1] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations." TDA progress report 42-122, Jet propulsion Lab., Pasadena, CA, August 1995.

[2] F. Daneshgaran and M. Mondin, "Design of interleavers for turbo codes based on a cost function," in *International Symposium on Turbo Codes & Related Topics*, Brest, France, September 1997.

[3] J. Andersen and V. Zyablov, "Interleaver design for turbo coding," in *International Symposium on Turbo Codes & Related Topics*, Brest, France, September 1997.

[4] J. Hokfelt and T. Maseng, "Methodical interleaver design for turbo codes," in *International Symposium on Turbo Codes & Related Topics*, Brest, France, September 1997.

[5] A. S. Barbulescu and S. S. Pietrobon, "Terminating the trellis of turbo-codes in the same state," *Electronics Letters*, vol. 31, pp. 22–23, January 1995.

[6] P. Jung and M. M. Nasshan, "Comprehensive comparison of turbo-code decoders," in *IEEE Vehicular Technology Conference*, Chicago, USA, July 1995.

[7] J. Hokfelt, O. Edfors, and T. Maseng, "A Turbo code interleaver design algorithm based on the performance of iterative decoding." submitted to IEEE Communications Letters[1].

[1] This paper was later accepted for publication in May 2000 [Paper IV], with the title "A Turbo code interleaver design criterion based on the performance of iterative decoding."

Paper III

Turbo Codes: Correlated Extrinsic Information and its Impact on Iterative Decoding Performance

Johan Hokfelt, Ove Edfors and Torleiv Maseng

IEEE Vehicular Technology Conference
Houston, Texas, May 1999

Turbo Codes: Correlated Extrinsic Information and its Impact on Iterative Decoding Performance

Johan Hokfelt, Ove Edfors and Torleiv Maseng

Abstract

The performance of a turbo code is dependent on two properties of the code: its distance spectrum and its suitability to be iteratively decoded. The performance of iterative decoding depends on the quality of the extrinsic inputs; badly correlated extrinsic inputs can deteriorate the performance. While most turbo coding literature assumes that the extrinsic information is uncorrelated, we investigate these correlation properties. An iterative decoding suitability *measure is presented, intended to serve as an indication on the degree of correlation between extrinsic inputs. The suitability measure can be used as a complement to the weight distribution when ranking interleavers.*

1 Introduction

The interleaver used in turbo codes [1] has two major tasks. The first is to ensure a good distance spectrum of the code by breaking up so called self-terminating input sequences, see *e.g.* [2]. In the case of maximum likelihood decoding this would be the natural and single task of the interleaver. However, turbo codes are decoded iteratively which is not optimal in the sense of making maximum likelihood decisions. The performance of iterative decoding compared to maximum likelihood decoding is dependent on the quality of the extrinsic information, which is the information being exchanged between the constituent decoders in the iterative decoding scheme. The choice of interleaver affects the degree of correlation between extrinsic inputs, and thereby the performance of iterative decoding. In this paper we present and investigate an *iterative decoding suitability* (IDS) measure, which indicates the degree of correlation between nearby extrinsic inputs.

The outline of the paper is as follows. In Section II the correlation properties of the extrinsic information are studied. Based upon this investigation, an iterative decoding suitability measure is proposed. In Section III, this measure is applied on a large number of interleavers, in order to evaluate its usefulness.

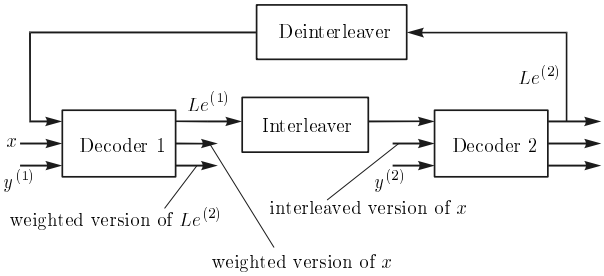


Figure 1: Structure of the iterative decoder.

2 Correlated Extrinsic Inputs

The iterative decoding scheme studied in this paper is depicted in Figure 1. Each soft-input/soft-output constituent decoder has three inputs and three outputs, according to *e.g.* [3]. The three inputs at decoding step k are the systematic input, x, the parity input, $y^{(1)}$ or $y^{(2)}$, and the extrinsic input from the previous decoder. The three outputs are a weighted version of the systematic input, a weighted version of the extrinsic input, and finally the new extrinsic output. These extrinsic outputs are used as input *a priori* probabilities in the next decoding step. The decision variables after each decoder is achieved as the sum of the three outputs, for each time instant. All input and output sequences are N bits long, which is also the size of the interleaver. Each constituent decoder employs the maximum *a posteriori* probability (MAP) algorithm as described in [3].

The systematic and parity inputs for each decoding step depend only on the received values for each specific bit. The extrinsic inputs, however, are dependent on a range of symbols in the received systematic and parity sequences. The following subsections investigate the nature of these dependencies.

2.1 The First Decoding Step

We start by investigating the dependencies between the outputs from the first constituent decoder and the input sequences. Since the extrinsic inputs to the first decoder are all zero, the decoder outputs are only dependent on the systematic and parity input sequences. Consider, for example, the decision variable at position i. This output is naturally dependent on the decoder inputs at position i, *i.e.* x_i and $y_i^{(1)}$. Furthermore, as a result of the trellis code, it is also (decreasingly) dependent on the inputs at time $i \pm 1$, $i \pm 2$, $i \pm 3$ and so on. In the following, these dependencies are investigated by the corresponding correlation coefficients. Examples of such correlation coefficients are depicted in Figure 2, showing the correlation between the systematic inputs and both

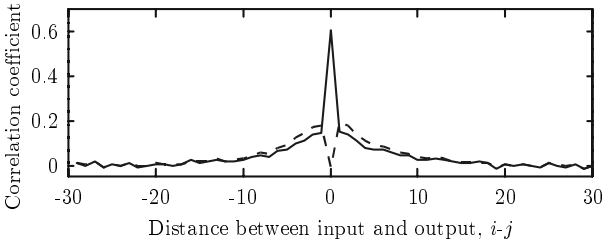


Figure 2: Correlation coefficients between the inputs and outputs of the first decoder. Solid: correlation between decision variable i and systematic bit j. Dashed: correlation between extrinsic output $Le_i^{(1)}$ and systematic bit j.

a decision variable (solid) and an extrinsic output (dashed). These coefficients were empirically obtained by simulating the decoding of a recursive convolutional code with feedback and parity polynomials 15_{oct} and 17_{oct}, respectively, repeatedly transmitting the all-zero code word.

The correlation coefficients between the extrinsic outputs from the first decoder and the sequence of systematic inputs are in the following represented by the matrix $\rho^{(1)}_{Le,x}$. Element (i,j) of $\rho^{(1)}_{Le,x}$, denoted $\rho^{(1)}_{Le_i,x_j}$, is the correlation coefficient between extrinsic output i after the first decoding step and systematic input j. Note that $\rho^{(1)}_{Le_i,x_j}$ is primarily dependent on the distance between i and j, and not on their absolute values, except for edge effects near the beginning and end of the sequences.

2.2 The Second Decoding Step

The difference between the first and second decoding step is the presence of extrinsic inputs. Output i from the second decoder is not only dependent on the sequence of systematic and parity inputs in the vicinity of position i, but also on the extrinsic inputs in the same vicinity. Each of these extrinsic inputs are in turn correlated to the channel inputs in the vicinity of its origin, before interleaving. Therefore, these extrinsic inputs provide output i from the second decoder with correlation to various parts of the systematic and parity sequences. Provided that the interleaver is suitably chosen, output i from the second decoder will thus be correlated to a wide range of channel inputs, not only to those in the vicinity of i. Intuitively, the decoding performance will be positively influenced by an increase in the number of channel inputs that affect each decoder output.

The dashed line in Figure 3 shows an example of the correlation between extrinsic output 50 from the second decoder and the entire sequence of systematic inputs, for a turbo code constructed with a 105-bit interleaver. The solid line shows the corresponding correlation coefficients between decision variable

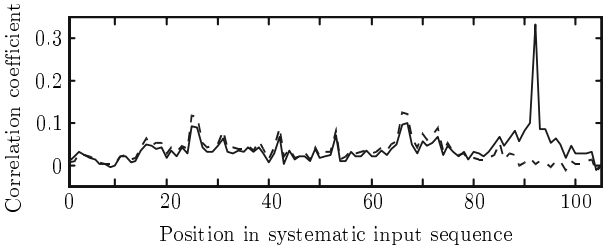


Figure 3: Correlation coefficients between the inputs and outputs of the second decoder. Solid: correlation between decision variable 50 and the sequence of systematic inputs. Dashed: correlation between extrinsic output 50 and the sequence of systematic inputs.

50 and the same sequence of systematic inputs. The peak around position 92 stems from the fact that extrinsic output 92 from the first decoder, for this particular interleaver, is interleaved to position 50 in the input sequence to the second decoder.

When it comes to investigating the performance of iterative decoding, we are primarily interested in the correlation coefficients between the extrinsic outputs and the input sequences, since it is the extrinsic outputs that are passed on to the next decoder in the iterative decoder. As in the case for the first decoder, the correlation coefficients between the extrinsic output from the second decoder and the systematic input sequence are represented by the matrix $\rho^{(2)}_{Le,x}$.

2.3 Iterative Decoding Suitability

Above we have discussed the correlation between the outputs of the constituent decoders and the received sequences, exemplified by the correlation coefficients to the systematic sequence. These correlation coefficients after the second decoder are dependent on the specific interleaver choice; by changing the interleaver, we change the correlation properties of the extrinsic information. Especially, interleavers with short cycles[1] result in high correlation to some parts of the received sequence, and lower correlation to other parts. Intuitively, we would like to spread this correlation as evenly as possible, since an output highly correlated to a few positions is very sensitive to channel noise at these positions.

Following the above discussion, we use the standard deviation of the correlation coefficients between extrinsic outputs and the sequence of systematic inputs as a quality measure. Denoting this standard deviation for extrinsic

[1] A short cycle occurs when two bits in a sequence are close to each other both before and after interleaving.

output i with V_i, we get

$$V_i = \frac{1}{N-1} \sum_{j=1}^{N} \left(\rho^{(2)}_{Le_i, x_j} - \overline{\rho^{(2)}_{Le_i, x}} \right)^2, \tag{1}$$

where $\overline{\rho^{(2)}_{Le_i,x}} = \frac{1}{N} \sum_{j=1}^{N} \rho^{(2)}_{Le_i,x_j}$, *i.e.* the average value of the correlation coefficients on row i of $\rho^{(2)}_{Le,x}$. A low value on V_i indicates a good quality of extrinsic output $Le^{(2)}_i$, since it is then evenly correlated to the sequence of systematic inputs. An interleaver well suited for iterative decoding should thus have low values on all V_i:s.

The above standard deviations indicate the quality of the extrinsic inputs to the second decoder. However, they do not reveal anything about the quality of the extrinsic information used as inputs in the third decoding step. The correlation properties of these extrinsic inputs are instead influenced by the deinterleaving rule. Therefore, the above calculations are performed for both the interleaver and the deinterleaver[2]. The corresponding correlation matrix for the deinterleaver is denoted $\rho^{(2)'}_{Le,x}$, which is substituted for $\rho^{(2)}_{Le,x}$ in (1) to calculate the standard deviations V'_i. We now have two sets of measures, the V_i:s and the V'_i:s, representing the quality of the extrinsic inputs to each of the two constituent decoders. To obtain a single measure on the iterative decoding suitability for an interleaver, we calculate the average of all the V_i:s and V'_i:s[3]. The *iterative decoding suitability* (IDS) measure is thus achieved as

$$\text{IDS} = \frac{1}{2N} \left(\sum_{i=1}^{N} V_i + \sum_{i=1}^{N} V'_i \right). \tag{2}$$

2.4 Analytical Approximation of Correlation Coefficients

It is not straightforward to derive analytical expressions of the correlation matrices $\rho^{(2)}_{Le,x}$ and $\rho^{(2)'}_{Le,x}$, which are needed to form the iterative decoding suitability measure. However, we have investigated a simple model in which the desired coefficients are expressed as a linear combination of the correlation after the first decoder, $\rho^{(1)}_{Le,x}$. The correlation coefficients after the first decoder are in turn approximated by an exponentially decaying function given by

$$\hat{\rho}^{(1)}_{Le_i,x_j} = \begin{cases} ae^{-c|j-i|} & \text{if } j \neq i, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

[2] A block interleaver with a large unbalance in the number of rows and columns is an example of an interleaver that result in large differences in the quality of the extrinsic inputs to the constituent decoders.

[3] In an earlier investigation [4], IDS was defined as $\max(\overline{V}, \overline{V'})$. Later, we have found that the average is more conceptually appealing.

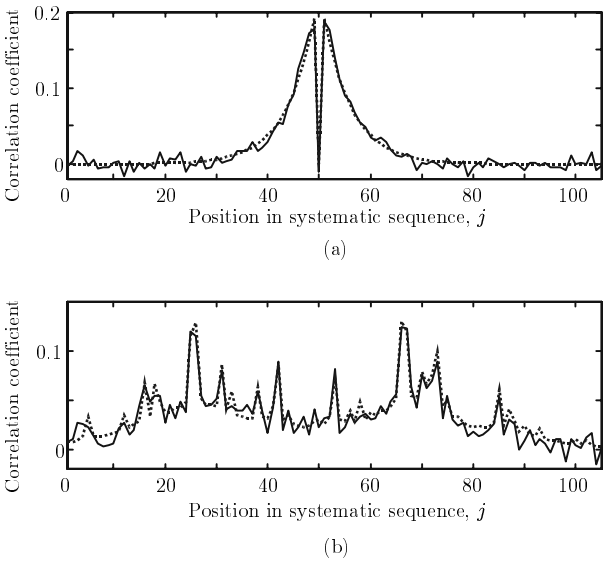


Figure 4: Comparison of approximated (dotted) and empirically found (solid) correlation coefficients between extrinsic output 50 and the sequence of systematic inputs, after a) the first decoder and b) the second decoder.

The constants a and c are adjusted so that the height and rate of decay of the approximation matches the empirically found coefficients. Figure 4a shows the approximated values (dotted line) with $a = 0.23$ and $c = 0.18$, together with the empirically found coefficients (solid line). The correlation coefficients between the extrinsic outputs of the second decoder and the received sequence x is in turn approximated as

$$\hat{\rho}^{(2)}_{Le,x} = \frac{1}{2}\hat{\rho}^{(1)}_{Le,x} P \left(I + \hat{\rho}^{(1)}_{Le,x} \right), \tag{4}$$

where P is a permutation matrix representing the interleaver. The interleaved sequence is obtained by multiplying the original sequence by P, *i.e.* as xP. The two terms in (4) stems from the two input sequences to the second decoder that result in correlation between the new extrinsic outputs and the systematic inputs, *i.e.* the systematic and extrinsic input sequences. The relevance of (4) is evaluated by comparing the approximated values with empirically found correlation coefficients, both shown in Figure 4b. The plot depicts the correlation coefficients between extrinsic output 50 from the second decoder and the sequence of systematic inputs. The proposed approximation is naturally not an exact description of the decoding process, but it captures the main features of the empirically found correlation coefficients.

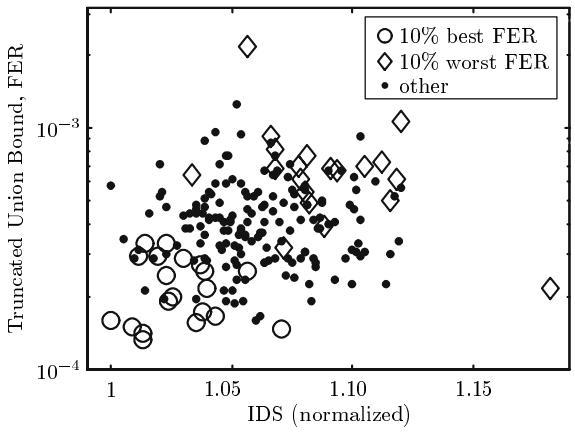


Figure 5: Scatter plot of the IDS values and truncated Union Bounds for 200 105-bit pseudo-random interleavers. The truncated union bound and the simulations are performed at $E_b/N_0 = 2.5$ dB. The 10% best and worst performing interleavers, during simulation, are marked with rings and diamonds respectively.

3 Evaluation of IDS

To evaluate the iterative decoding suitability measure we compared simulated error rate performances (on an AWGN channel) of a large number of pseudo-random interleavers to their IDS values. In addition, truncated union bounds on the frame-error rate probability for each interleaver was calculated. Two interleaver sizes were used, 105 and 500 bits.

Figure 5 shows a scatter plot of the truncated union bounds[4] and IDS values for 200 105-bit interleavers. The IDS values are normalized with the lowest found value, since the IDS is a relative comparison. It is observed that the best performing interleavers have low values on *both* the IDS measure *and* the truncated union bound. It is further noted that interleavers that have a low value on one measure but high on the other, perform, in this case, among the 10% worst performing interleavers. This result stress the fact that both the distance spectrum of a turbo code *and* its suitability to be iteratively decoded are important issues when it comes to ranking the performance of interleavers used in turbo codes. The frame-error rates of the 200 turbo codes range from $1{\cdot}10^{-3}$ to $4{\cdot}10^{-3}$ at a signal-to-noise ratio of 2.5 dB, after 20 decoding iterations. The constituent encoders have generator polynomials $(1, 17/15)_{oct}$.

The convergence rate of the iterative decoding is also affected by the correlation properties of the extrinsic information. Figure 6 shows a scatter plot of

[4] The bound is truncated after Hamming distance 22.

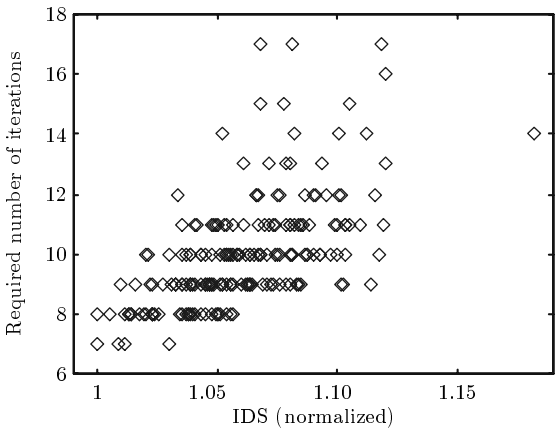


Figure 6: Scatter plot of the IDS values and the required number of decoding iterations for a frame-error rate of $5 \cdot 10^{-3}$, for 200 105-bit pseudo-random interleavers.

the normalized IDS values and the required number of decoding iterations for a frame-error rate of $5 \cdot 10^{-3}$.

As pseudo-random interleavers increase in size, the variation in their IDS values tend to diminish. Therefore, the IDS measure is mostly suitable for small pseudo-random interleavers, in the range of 100 to 500 bits. The corresponding scatter plots in Figures 5 and 6 for the 500-bit pseudo-random interleavers are similar to the ones shown for the 105-bit interleavers, but with a lower significance of the IDS measure.

Figure 7 shows the frame-error rate performances and IDS values of four designed 640-bit interleavers evaluated for the UMTS standardization [5] (Int. #1 is a pseudo-random interleaver, the others are designed). Also in this case, there is a correspondence between low IDS values and good error correcting performances. The turbo codes in this example use 8-state encoders and 8 decoding iterations.

4 Conclusions

The correlation properties of the extrinsic information in an iterative decoder have been studied. These correlation properties depends on the particular choice of interleaver, which therefore influence the performance of iterative decoding. An *iterative decoding suitability* measure was presented and investigated, intended as a complement to the distance spectrum when ranking interleaver performances. Simulation results indicate that the IDS is related to both the convergence rate and final frame-error rate of turbo codes. It is given

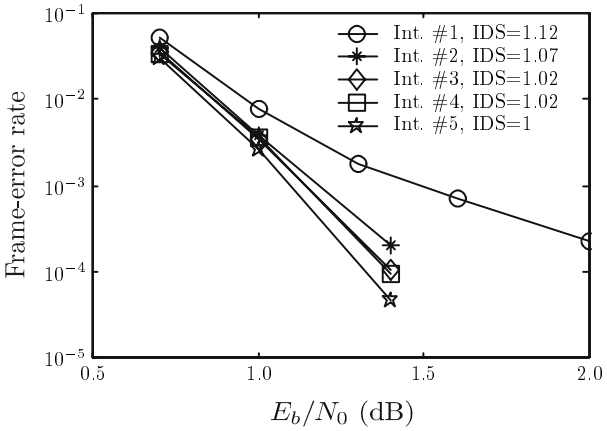


Figure 7: Simulated frame-error rates of designed 640-bit interleavers, and their IDS values.

in a closed form expression and requires no simulations which makes it useful for interleaver design and ranking.

References

[1] Claude Berrou and Alain Glavieux. Near optimum error correcting coding and decoding: Turbo-Codes. *IEEE Transactions on Communications*, 44(10):1261–1271, October 1996.

[2] S. Dolinar and D. Divsalar. Weight distributions for turbo codes using random and nonrandom permutations. TDA progress report 42-122, Jet propulsion Lab., Pasadena, CA, August 1995.

[3] P. Jung and M. M. Nasshan. Comprehensive comparison of turbo-code decoders. In *IEEE Vehicular Technology Conference*. Chicago, USA, July 1995.

[4] J. Hokfelt, O. Edfors, and T. Maseng. Assessing interleaver suitability for turbo codes. In *Nordic Radio Symposium*. Saltsjöbaden, Sweden, October 1998.

[5] Anders Henriksson, Johan Hokfelt, and Ove Edfors. Evaluation of an interleaver design algorithm for turbo codes in UMTS. Tdoc 419/98, ETSI SMG2 UMTS L1 Expert Group, Meeting no 7, Sweden, Oct. 1998.

[6] L. Perez, J. Seghers, and D. J. Costello Jr. A distance spectrum interpretation of turbo codes. *IEEE Transactions on Information Theory*, 42(6):1698–1709, November 1996.

[7] N. Wiberg. Codes and decoding on general graphs, April 1996. Ph.D. thesis, Linköping University, Sweden (ISSN 0345-7524, ISBN 91-7871-729-9).

Paper IV

A Turbo Code Interleaver Design Criterion Based on the Performance of Iterative Decoding

Johan Hokfelt, Ove Edfors and Torleiv Maseng

Accepted for publication, IEEE Communications Letters

A Turbo Code Interleaver Design Criterion Based on the Performance of Iterative Decoding

Johan Hokfelt, Ove Edfors and Torleiv Maseng

Abstract

The performance of a Turbo code is dependent on two code properties: its distance spectrum and its suitability to be iteratively decoded. Both these properties are influenced by the choice of interleaver used in the turbo encoder. This paper presents an interleaver design criterion that focuses on the performance of iterative decoding, based on the correlation properties of the extrinsic inputs. Interleavers designed with the proposed criterion achieve very competitive performances, both in terms of convergence rates and error correcting capabilities.

1 Introduction

Since the introduction of turbo codes [1], quite an effort has been made to find good interleaver design methodologies. Most of these efforts have been directed towards finding interleavers that yield codes with improved distance spectra, a natural design criterion when maximum likelihood decoding is employed. Turbo codes are however decoded iteratively, which is suboptimal to maximum likelihood decoding. This prompts for an additional design criterion: minimization of the performance deterioration due to iterative decoding.

Some previously proposed interleaver design criteria result in good iterative decoding performance, for example S-random interleavers [2]. These are semi-random interleavers for which permutations resulting in 'short cycles' are avoided. A short cycle occurs when two bits are close to each other both before and after interleaving. It was concluded in [3] that these short cycles impair the performance of iterative decoding. In this paper we investigate the influence of short cycles using correlation properties of the extrinsic information. These correlation properties are then exploited to form an interleaver design criterion that strives to make nearby extrinsic inputs as uncorrelated as possible.

2 Correlation Properties of the Extrinsic Information

The soft output from each constituent decoder can be divided into three parts; the extrinsic output, a weighted version of the systematic input, and a copy of the input a priori information [1, 4]. In this context we focus on the properties of the extrinsic output, since this is the part that is used as a priori information in the next constituent decoder. The correlation properties of these extrinsic outputs have received little attention in the literature. Most often, the extrinsic outputs/inputs are assumed to be independent of each other. In this section we investigate the correlation properties of the extrinsic information, and motivate their use as part of a design criterion in an interleaver design algorithm. In the following, we use the notation Le_i for extrinsic output i, x_i for information symbol i ($x_i \in \{\pm 1\}$), and n_i for the noise sample added to systematic symbol i.

Due to the structure of a trellis encoder/decoder, each decoder output depends on a range of input noise samples. These dependencies after the first decoding step are illustrated in Fig. 1(a), where the solid line shows simulated[1] correlation coefficients between extrinsic output 50 (out of 105) and the sequence of noise samples added to the systematic sequence. Since nearby extrinsic outputs depend on partially the same inputs, they are likely to be correlated. This is confirmed in Fig. 1(b), showing the correlation between extrinsic output 50 and the entire sequence of extrinsic outputs.

Now consider the second decoding step. The extrinsic outputs from the second decoder becomes correlated with the noise samples through two of the input sequences: the systematic inputs and the extrinsic inputs. Consequently, the correlation between the extrinsic output from the second decoder and the input noise samples is more complicated than after the first decoder. The solid line in Fig. 1(c) depicts examples of such correlation coefficients, achieved with a pseudo-random interleaver. The systematic inputs give rise to the correlation spikes, while each extrinsic input increases the correlation to positions in the neighborhood of each spike. Two spikes that are close to each other indicate a short cycle, resulting in increased correlation to the noise samples near these spikes, at the expense of lower correlation to other positions. If short cycles are avoided, the correlation between extrinsic outputs and the noise samples is instead distributed among a larger part of the sequence of noise samples. This will be used in the following to form an interleaver design criterion.

[1] The sought correlation coefficients are proportional to the expectations $E[Le_i x_i \cdot n_j x_j]$, which were estimated by averaging simulated values of $Le_i x_i \cdot n_j x_j$ over a large number of frames.

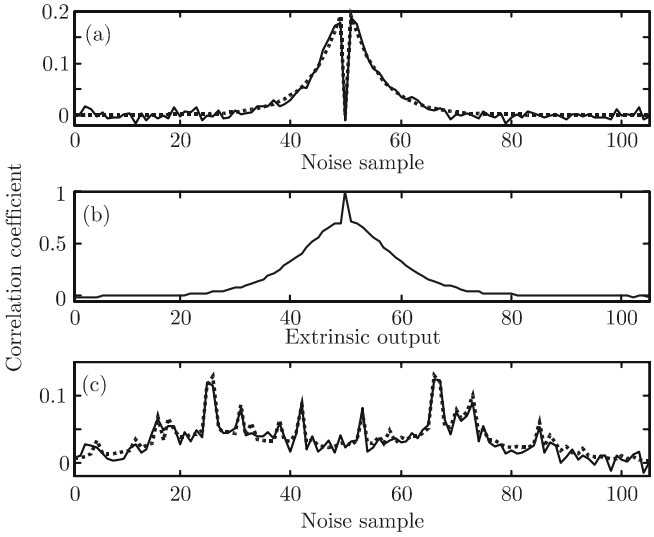


Figure 1: (a) Correlation between extrinsic output 50 from the first decoder and the noise samples added to the systematic symbols. (b) Autocorrelation of the extrinsic output sequence after the first decoder. (c) Correlation between extrinsic output 50 from the second decoder and the noise samples added to the systematic symbols. The solid lines correspond to empirically found coefficients, while the dotted lines are closed form approximations. All were obtained for a rate-1/3 turbo code with generator polynomials $(17/15)_{oct}$ and a 105-bit interleaver, transmitting over an AWGN channel at $E_b/N_0 = 1.0$ dB.

3 Interleaver Design

The investigation in the previous section indicates a useful relationship between short interleaver cycles and the correlation properties of the extrinsic outputs. For the purpose of interleaver design, we desire a short form expression for the correlation between extrinsic outputs (from the second decoder) and the sequence of noise samples. After studying the simulated correlation, we have chosen to use the following approximation[2]

$$\rho^{(1)}_{i,j} = \begin{cases} ae^{-c|j-i|} & \text{if } j \neq i \\ 0 & \text{otherwise} \end{cases} \qquad i,j = 1,2,\ldots,N, \tag{1}$$

for the correlation between $Le_i x_i$ and $n_j x_j$ after the first decoder. The dotted line in Fig. 1(a) shows $\rho^{(1)}_{50,j}, j = 1,2,\ldots,105$, with $a = 0.23$ and $c = 0.18$, which

[2] Compare with the exponentially decaying autocorrelation function of an Auto-Regressive process driven by white Gaussian noise.

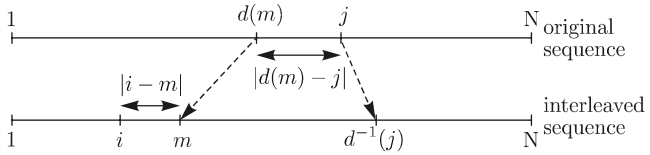


Figure 2: Illustration of the interleaver mappings $d(m)$ and $d^{-1}(j)$, contributing to the correlation between noise sample j and extrinsic output i.

is to be compared to the solid line showing the simulated correlation coefficients. For the corresponding coefficients after the second decoder, a linear model of the decoder in which the correlation from the two relevant inputs are averaged yields the approximation

$$\rho_{i,j}^{(2)} = \frac{1}{2}\overbrace{\left(1-\delta_{d^{-1}(j)-i}\right)ae^{-c|d^{-1}(j)-i|}}^{\text{systematic input}} + \underbrace{\frac{1}{2}\sum_{\substack{m=1\\ m\neq i, d(m)\neq j}}^{N} a^2 e^{-c(|d(m)-j|+|i-m|)}}_{\text{extrinsic inputs}}, \tag{2}$$

where δ is the Kronecker delta-function. The interleaver is present in the form of $d(m)$, as illustrated in Fig. 2. $\rho_{50,j}^{(2)}$ is plotted in Fig. 1(c) with a dotted line, again to be compared to the solid line representing simulated correlation coefficients. Even though (2) is a heuristic approximation of the true correlation coefficients, it is reasonably accurate and proves to be very useful for the purpose of interleaver design.

A straightforward interleaver design procedure is to assign the elements $d(i), i \in \{1, 2, \ldots, N\}$, element by element until the whole interleaver is defined. When choosing each new interleaver mapping $d(i)$, a position in the original sequence is chosen for which the correlation stemming from the already defined mappings is low. This correlation is approximated using (2), which need only be calculated for positions j not already chosen. For these positions, $d^{-1}(j)$ is not yet defined, and hence the first term of (2), $ae^{-c|d^{-1}(j)-i|}$, does not exist. As a consequence, the minimization is performed as

$$d(i) = \arg\min_{j\in\mathcal{L}} \sum e^{-c(|d(m)-j|+|i-m|)}, \tag{3}$$

where the summation is carried out over the already defined interleaver elements $d(m)$. The set $\mathcal{L}$ is the permissible positions to choose from. It contains the still not assigned positions in the original sequence, except for those eliminated by other (optional) design criteria. Other design criteria can include

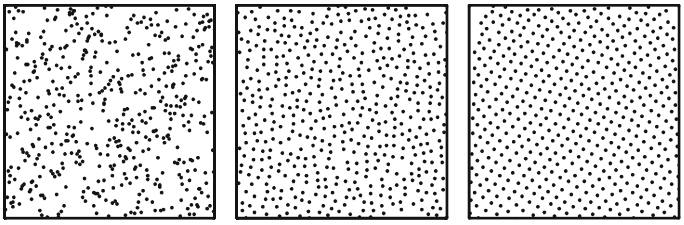


Figure 3: Illustration of permutation matrices for three types of interleavers (500 bits). Each dot corresponds to a one in the permutation matrix. (a) Pseudo-random, (b) S-random, and (c) correlation designed interleaver.

specific distance spectrum restrictions and/or trellis termination considerations [5]. The use of such criteria is however beyond the scope of this paper.

The result of using the correlation criterion in the design is illustrated in Fig. 3, which shows the permutation matrices for a pseudo-random, an S-random and a correlation designed interleaver. Each dot in these plots indicate a one in the permutation matrix; two dots that are near each other correspond to a short cycle, which in turn indicates that two nearby extrinsic inputs are correlated. In general, pseudo-random interleavers create a large amount of short cycles, whereas these are avoided with S-random and correlation-designed interleavers.

4 Performance Evaluation

The proposed design criterion was evaluated by comparing bit and frame error rate performances of rate-1/3 turbo codes using various well known types of interleavers, such as pseudo-random, S-random [2], block helical simile [6], ordinary block interleavers, as well as interleavers designed for optimized distance spectrum. The correlation-designed interleavers resulted in both the fastest decoding convergence (also studied in [7]) and the best asymptotic performance. This is exemplified in Fig. 4(a), showing the frame-error rate performances of two 512-bit interleavers: one correlation-designed and one distance spectrum designed. Note that the correlation-designed interleaver yields a significantly lower error floor, despite its lower minimum Hamming distance (27 versus 32).

While Fig. 4(a) presents results for 512-bit interleavers only, it is of course interesting to investigate the performances for a wide range of interleaver sizes. This is done in Fig. 4(b), where we show required SNRs for interleaver sizes ranging from 105 to 4096 bits, at a frame error rate of 10^{-4}. The performance gains achieved with correlation designed interleavers over the S-random[3] in-

[3] $(N, S) \in \{(105, 7), (200, 9), (500, 12), (1000, 17), (4096, 42)\}$

terleavers are approximately 0.1 dB for small and medium size interleavers. Since good interleavers tend to perform close to the lower bound[4], the room for improvements is small.

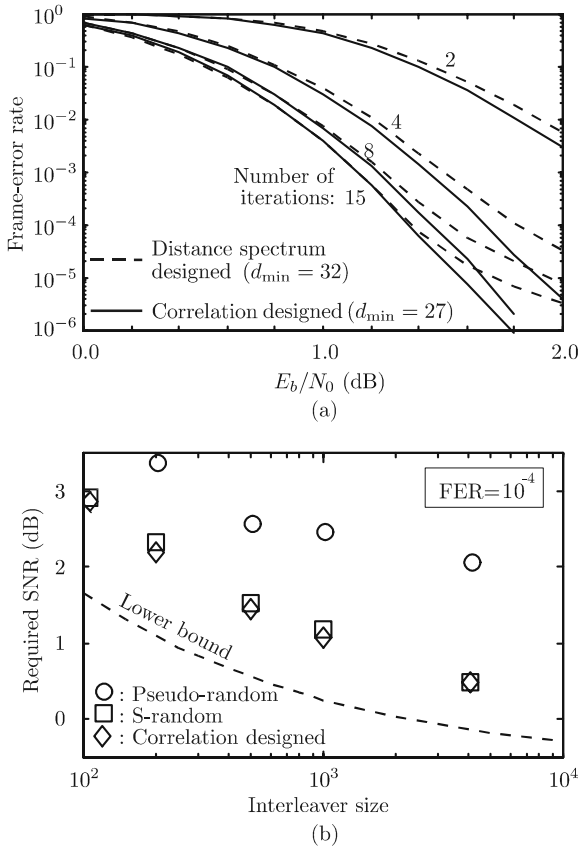


Figure 4: (a) Comparison of convergence rates and frame error rates for a correlation designed and a distance spectrum designed interleaver, both 512 bits long. (b) Performance comparison of turbo codes using three types of interleavers. The plot shows the minimum required signal-to-noise ratio for a frame error rate of 10^{-4} using 8-state constituent encoders and 10 decoding iterations, as well as a sphere-packing lower bound.

[4] Sphere-packing lower bounds for unconstrained-input, unconstrained-output channels [8].

5 Conclusion

A new interleaver design criterion for turbo codes has been presented. In contrast to previous design criteria, which concentrate on the distance spectra, the basis of the proposed criterion is the correlation properties of the extrinsic information. The correlation criterion can be thought of as a soft-version of the S-random design criterion. The spreading associated with these design criteria results in two favorable properties: competitive distance spectra and low correlation between nearby extrinsic inputs. Simulation results indicate a faster decoding convergence as well as an improved asymptotic performance for interleavers designed with the presented criterion. The computational complexity of the proposed criterion is very low, which allows for quick design of large interleavers.

References

[1] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-Codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, October 1996.

[2] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations." TDA progress report 42-122, Jet propulsion Lab., Pasadena, CA, August 1995.

[3] N. Wiberg, "Codes and decoding on general graphs," April 1996. Ph.D. thesis, Linköping University, Sweden (ISSN 0345-7524, ISBN 91-7871-729-9).

[4] P. Jung and M. M. Nasshan, "Comprehensive comparison of turbo-code decoders," in *IEEE Vehicular Technology Conference*, Chicago, USA, July 1995.

[5] J. Hokfelt, O. Edfors, and T. Maseng, "A survey on trellis termination alternatives for turbo codes," in *IEEE Vehicular Technology Conference*, Houston, Texas, USA, May 1999.

[6] A. S. Barbulescu and S. S. Pietrobon, "Terminating the trellis of turbo-codes in the same state," *Electronics Letters*, vol. 31, pp. 22–23, January 1995.

[7] J. Hokfelt, O. Edfors, and T. Maseng, "On the convergence rate of iterative decoding," in *IEEE Global Telecommunicatons Conference*, Sydney, Australia, November 1998.

[8] S. Dolinar, D. Divsalar, and F. Pollara, "Turbo code performance as a function of code block size," in *International Symposium on Information Theory*, Boston, USA, August 1998.

Paper V

Interleaver Structures for Turbo Codes with Reduced Storage Memory Requirement

Johan Hokfelt, Ove Edfors and Torleiv Maseng

IEEE Vehicular Technology Conference
Amsterdam, Holland, 1999

Interleaver Structures for Turbo Codes with Reduced Storage Memory Requirement

Johan Hokfelt, Ove Edfors and Torleiv Maseng

Abstract

This paper describes two interleaver structures that reduce the memory requirement for the storage of interleaver rules. The first structure offer memory reductions of more than 50%, compared to storing the entire interleaver vector. The second structure is useful when a range of interleaver sizes are to be stored, offering memory reductions asymptotically approaching 50%. By combining the two structures, a total memory reduction of approximately 75% is achieved. This is obtained without any significant reduction of the error correcting performance of the codes.

1 Introduction

A turbo code typically consists of two recursive encoders in parallel, separated by an interleaver [1] as shown in Fig. 1. The design parameters of a turbo code are primarily the generator polynomials of the constituent encoders, normally chosen to be identical, and the particular choice of interleaver mapping. The issue of choosing generator polynomials is discussed in for example [2, 3, 4], while for interleaver design strategies and interleaver structures, see for example [5, 6, 7, 8, 9].

In some applications, when many, and possibly large, interleavers are used, the implementation complexity to generate the interleaver rules is of importance. For example, in the standardization of UMTS, such considerations have achieved significant attention. There are basically two approaches that can be used to lower the implementation complexity of interleaver rules: either by using algebraic interleavers that can be generated in real time, or by imposing structures on the interleavers, which reduce the storage requirement of the interleaver rules.

This paper presents two interleaver structures that reduce the amount of memory required to store interleaver rules. The first structure is referred to as an *odd-even symmetric* structure, which reduces the memory requirement with slightly more than 50% compared to storing the entire interleaver vector. The

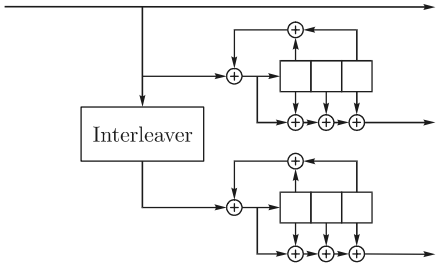


Figure 1: Example of a turbo code encoder, using constituent encoders with generator polynomials $(17/15)_{oct}$.

second structure, *expanded* interleavers, is useful when a series of interleavers with sizes of the form $2^k x, k = 0, 1, 2, \ldots$, where x is the size of the smallest desired interleaver, are to be stored. When a series of interleavers are designed with the expansion criterion, it is sufficient to store the largest interleaver only, offering memory savings approaching 50%. Fortunately, the interleaver structures can be combined, with a total memory requirement reduction of approximately 75%, compared to storing all the interleaver vectors.

Naturally, every interleaver restriction reduces the design freedom, and hereby, possibly, deteriorates the performance of the interleavers. However, simulations show that interleavers with the structures presented in this paper perform essentially the same as unconstrained interleavers, if designed appropriately.

2 Interleaver structures

Let the interleaver rule be represented by a vector of N integers, π=[$\pi(1)$ $\pi(2) \ldots$ $\pi(N)$], where $\pi(i) = j$ indicates that input position i is interleaved to position j, and N is the size of the interleaver. The interleaver structures presented in this paper impose certain restrictions on the permissible choices of the mappings $\pi(i)$. First, we describe two different restrictions that individually offer implementation simplifications, and thereafter these restrictions are merged into a combined structure, offering total storage savings of as much as 75%.

2.1 Odd-even symmetric interleaver structure

Consider an interleaver rule that swaps pairs of positions, i.e. a symmetric interleaver. If all the pairs are known, the interleaver rule is known. Since there are only $N/2$ such pairs, if organized properly, the storage of these pairs requires less memory than storing an entire interleaver vector with N addresses.

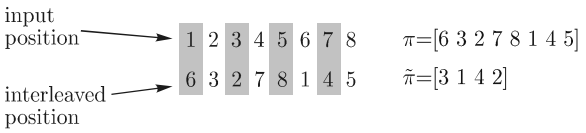


Figure 2: Example of an 8-bit odd-even symmetric interleaver. Each odd position in the input sequence is mapped to an even position, and vice versa. Further, if input i is mapped to position j, then input j is mapped to position i (symmetry).

One possible organization strategy is to require every position in the first half of the input sequence to be swapped with a position in the second half. However, this restriction severely reduces the design freedom of the interleaver, notably deteriorating the error correcting performance of the code. There are however other sequence partitions that yield a simple organization of the swapping pairs, without degrading the interleaver performance. One such partition is to swap every odd position with an even position, and vice versa. This interleaver structure, denoted odd-even symmetric, is thus achieved with the following two restrictions:

1. $i \bmod 2 \neq \pi(i) \bmod 2, \forall i$ (odd to even),
2. $\pi(i) = j \Rightarrow \pi(j) = i$ (symmetry).

With these restrictions, it is sufficient to store the interleaver rules for all the odd positions, since by performing swaps, the even positioned bits are automatically interleaved.

Assume that only the odd positions in the interleaver vector are stored. All the stored addresses are then even integers, implying that the least significant bit (LSB) in the binary representation of each address is always zero. Thus, the LSB need not be stored, which offers additional memory savings if the interleaver rule is stored with custom made memory cells. This shift of the binary representation corresponds to dividing each number by 2, so that the stored vector consists of $N/2$ integers ranging from 1 to $N/2$. This vector will in the following be denoted $\tilde{\pi}$, and it is given by $\tilde{\pi}(i) = \pi(2i-1)/2, i \in \{1, 2, \ldots, N/2\}$.

As an example, the swapping pairs of an 8-bit odd-even symmetric interleaver are illustrated in Fig. 2. The shown interleaver vector is π=[6 3 2 7 8 1 4 5], and the reduced memory-requirement vector is $\tilde{\pi}$=[3 1 4 2].

The implementation of the interleaving rule of an odd-even symmetric interleaver is straightforward: elements at even positions are interleaved by storing them sequentially and reading them in the order specified by $\tilde{\pi}$; elements at odd

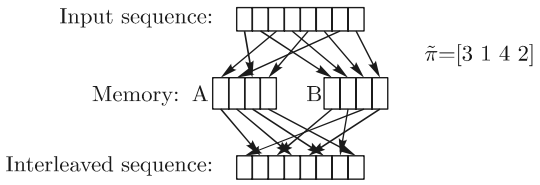


Figure 3: Implementation example of an 8-bit odd-even symmetric interleaver. The interleaver rule is stored by the 4-element vector $\tilde{\pi}$=[3 1 4 2].

positions are interleaved by storing them in the order specified by $\tilde{\pi}$ and reading them sequentially. As an example, we study the interleaving of the extrinsic outputs produced by the first constituent decoder. For illustrative purposes, it is suitable to partition the memory used to store the extrinsic information between the decoders into two logically separated memory areas, A and B. With these, odd extrinsic outputs on the form $2n-1$, $n \in \{1, 2, \ldots, N/2\}$ are stored at address $\tilde{\pi}(n)$ in memory A, while even outputs, $2n$, $n \in \{1, 2, \ldots, N/2\}$, are stored at address n in memory B. The second constituent decoder performs a similar action when reading its extrinsic inputs: odd inputs are read from memory B at address $\tilde{\pi}(n)$, and even inputs are read from memory A at address n. Such an interleaver implementation is illustrated in Fig. 3. The deinterleaver implementation is identical, due to the symmetric property.

2.2 Expanded interleaver structure

The expanded interleaver is also a structure where the mappings are restricted to two separate partitions of the input sequence. In the case of the odd-even partitioning in the previous section, the mappings were constrained to be from one set to the other. We will now study the benefits of instead requiring the mappings to be within each of the two sets. The advantage of this restriction is that the resulting interleaver can be viewed as a combination of two separate interleavers, each half the size of the combined interleaver. Assume that the two sets consist of the odd and even positions respectively. Then, by using every second element in the stored interleaver vector, an interleaver half as large as the original interleaver is achieved. Naturally, it is possible to require that also the mappings of the new, smaller interleaver are restricted to additional, similar, partitions. As long as this requirement is fulfilled, smaller and smaller interleavers can be produced, by simply using every second element in the nearest larger interleaver.

A straightforward way of constructing a series of expanded interleavers is to start with the smallest desired interleaver size, and expand this by inserting undefined elements between the already existing positions in the interleaver. The new, undefined positions are then assigned with mappings according to

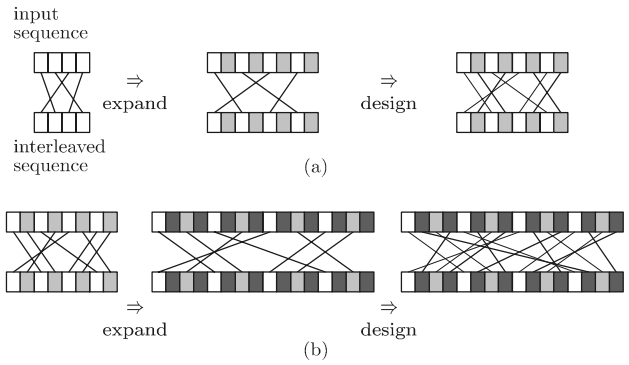


Figure 4: (a) Expansion of a 4-bit interleaver to an 8-bit interleaver, and (b) Expansion of the 8-bit interleaver to a 16-bit interleaver. Both the original 4-bit and the intermediate 8-bit interleavers can be retrieved from the final 16-bit interleaver.

some interleaver design criteria, independent of the restrictions imposed by the expansion structure. After the expanded interleaver is designed, further expansions can be performed until the largest desired interleaver size is obtained. Two such expansions are shown in Fig. 4, first from a 4-bit interleaver to an 8-bit, and then from the 8-bit to a 16-bit interleaver.

The advantage of expanding interleavers as described is the reduction of storage memory required for the interleaver rules; the expanded 16-bit interleaver in Fig. 2(b) can be used to interleave 4-bit, 8-bit and 16-bit sequences. This interleaver structure is thus useful when a range of interleaver sizes are to be stored. The amount of storage memory saved approaches 50%, as the number of interleaver sizes increases.

2.3 Expanded odd-even symmetric interleavers

The described structures can be merged into a combined interleaver structure which is both expanded and odd-even symmetric. Due to the contradictory constraints on the odd/even subsets, the expansion need to be modified compared to the description above. One way to maintain the odd-even property in an expanded interleaver is to insert *two* undefined elements after *every second* entry in the original interleaver. This modification ensures that each element on an even position in the original interleaver remains on an even position after expansion, and vice versa. Thus, if the original interleaver meets the odd-even symmetric constraints, and if the assignment of the inserted elements is performed with these restrictions, the new interleaver will be an *expanded, odd-even symmetric interleaver*. The expansion of an 8-bit odd-even symmetric interleaver to a 16-bit odd-even symmetric interleaver is illustrated in Fig. 5.

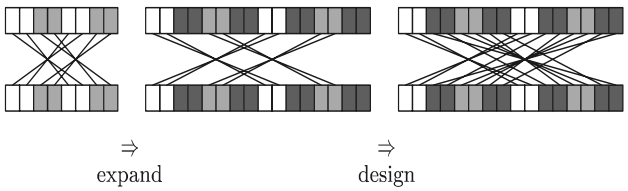


Figure 5: An interleaver expansion that preserves the odd-even symmetric properties of an interleaver. Instead of inserting one unassigned element after every original interleaver entry, two unassigned elements are inserted after every second original entry.

The expanded and designed 16-bit interleaver in Fig. 5 can be stored on hardware as $\tilde{\pi}_{16}$=[5 8 7 6 1 2 3 4]. For these small interleaver sizes, the restrictions imposed on the interleavers reduce the design freedom substantially. However, when designing interleavers with more reasonable sizes (at least one hundred bits), the imposed restrictions do not severely reduce the design freedom of the interleavers.

The implementation of expanded odd-even symmetric interleavers is identical to the odd-even symmetric implementation – with the addition that the same hardware can be used for a range of interleaver sizes. Interleaving of 16-, 8- and 4-bit sequences with the interleaver in Fig. 5 are illustrated in Fig. 6. The only difference between interleaving an 8-bit sequence instead of a 16-bit sequence is that the interleaver index-counter is clocked twice between each new bit to be stored/read. This holds also for the index-counter for the sequentially stored/read bits, which now use addresses $1, 3, 5, \ldots$, instead of $1, 2, 3, \ldots$.

3 Simulation results

The error correcting performances of turbo codes using interleavers with the described structures have been simulated for AWGN channels. We use rate-1/3 turbo codes with up to 15 decoding iterations, where each constituent decoder employ the log-MAP decoding algorithm. The generator polynomials of the constituent encoders were $(1, 17/15)_{\text{oct}}$. The first encoder was terminated with tail bits, while the second encoder was truncated in an unknown state.

Simulations were performed with interleavers of sizes 320, 640, 1280, 2560 and 5120 bits. The 320-bit interleaver is an odd-even symmetric interleaver, while all the rest are expanded, odd-even symmetric. Aside from these interleaver restrictions, the interleavers were designed to yield good correlation properties of the extrinsic information [9], as well as good distance properties of the resulting turbo codes. The bit- and frame-error rate results after 8 and 15 decoding iterations are shown in Fig. 7 and Fig. 8 respectively. In these

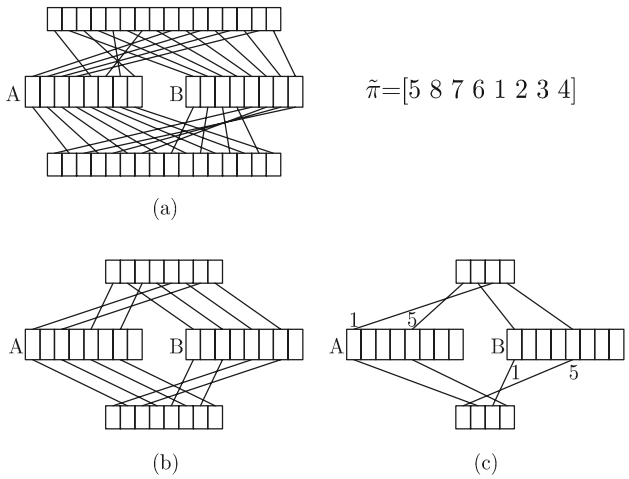


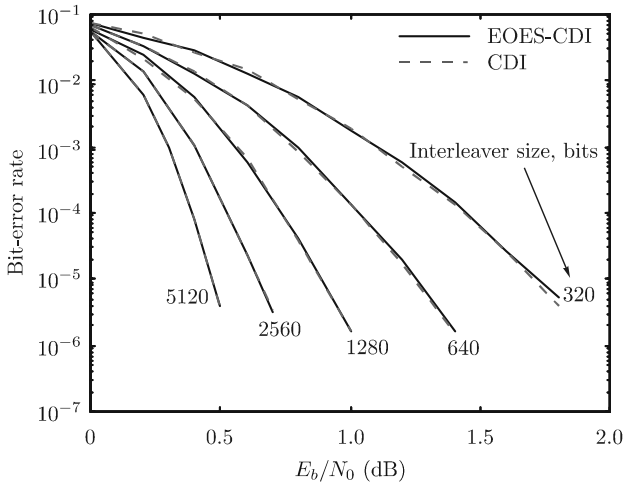
Figure 6: Implementation of a 16-bit expanded odd-even symmetric interleaver, used to interleave (a)16-bit-, (b) 8-bit- and (c) 4-bit input sequences. For each step down in input sequence length, the clock speed of the index counter for $\tilde{\pi}$ is doubled. For example, when interleaving the 4-bit sequence, the only addresses used in $\tilde{\pi}$ is 5 and 1.

figures, the structured interleavers are denoted EOES-CDI; expanded odd-even symmetric, correlation designed interleaver. As a performance reference, we use interleavers designed completely without structure restrictions, denoted CDI. This type of interleavers have shown very competitive performances in comparison to interleavers designed by many other design strategies, see for example [9, 10]. When only one line is visible in the performance plots, it is because the performances are right on top of each other. The results indicate that the presented interleaver structures have essentially no influence on the performance of the compared turbo codes, as long as the interleavers are properly designed.

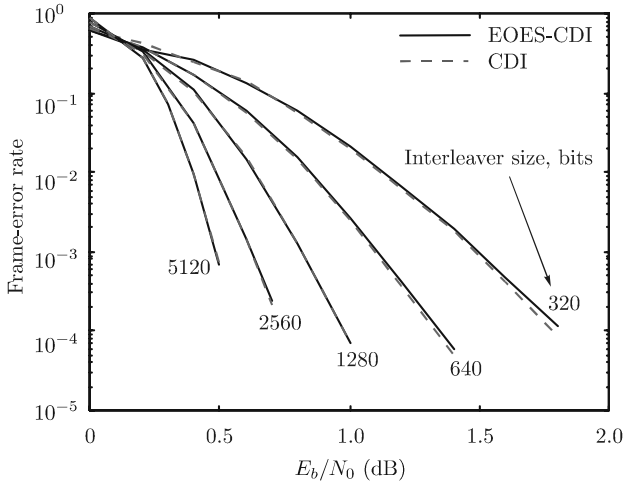
We present here only the performances of interleavers designed with both the expanded and the odd-even symmetric restrictions. However, interleavers designed with only one of the constraints, as presented in Section 2.1 and 2.2 respectively, perform the same as the performances shown in Fig. 7 and Fig. 8.

4 Discussion and conclusions

The concept of pruning interleavers were discussed in [7]. When pruning interleavers, very high interleaver size granularity is achieved by disregarding the interleaver mappings to positions above the size of the desired interleaver. For example, a 570-bit interleaver is obtained by disregarding all the elements that



(a)



(b)

Figure 7: Simulated bit- and frame-error rates after 8 decoding iterations of rate-1/3 turbo codes of various interleaver sizes, on an AWGN channel. The expanded odd-even symmetric (EOES-CDI) interleavers perform essentially as well as the interleavers designed entirely without the structure restrictions (CDI).

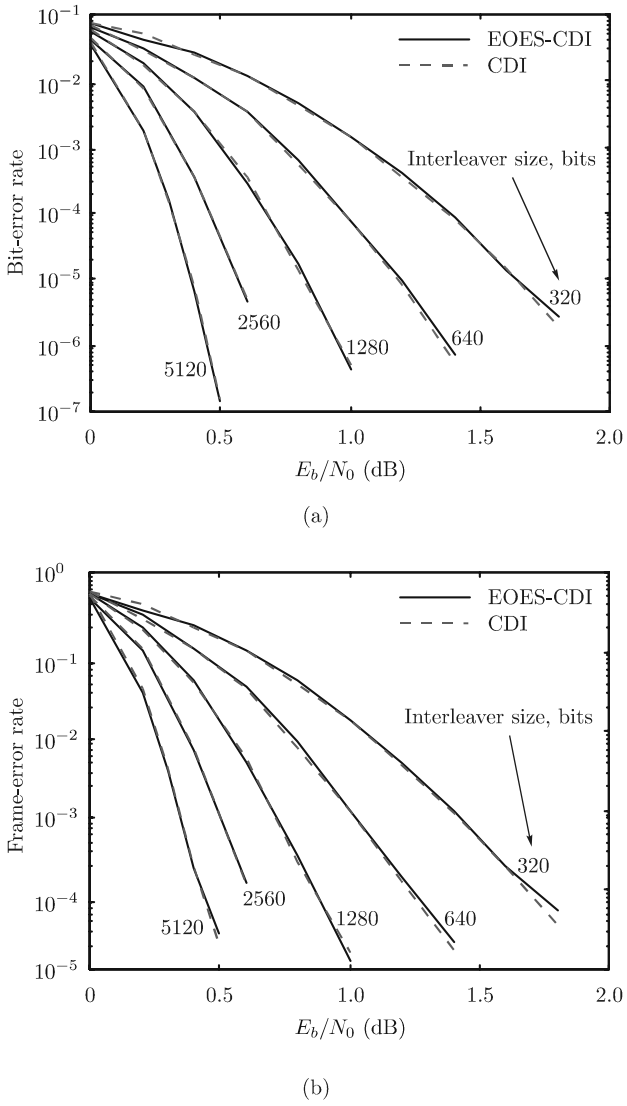


Figure 8: Simulated bit- and frame-error rates after 15 decoding iterations, of rate-1/3 turbo codes of various interleaver sizes, on an AWGN channel. The expanded odd-even symmetric (EOES-CDI) interleavers perform essentially as well as the interleavers designed entirely without the structure restrictions (CDI).

are larger than 570 in the nearest larger interleaver, i.e. the 640-bit interleaver in our case. However, for the interleavers evaluated in this paper we found that the correlation properties of the extrinsic information are better preserved, if the pruning is performed at both ends of an interleaver. This means that elements both below and above a certain number, depending on the desired interleaver size, are disregarded.

The storage of an expanded odd-even symmetric interleaver of size 5120 bits requires 2560 addresses to be stored, each address represented by 12 bits. This amounts to approximately $3.1 \cdot 10^4$ memory cells. As described, this interleaver can be used to interleave all sequences with lengths on the form $5120/2^k$, $k = 0, 1, 2, \ldots$, down to the size of the original interleaver used for the first expansion. Assuming instead that interleavers with sizes ranging from 320 to 5120 bits are to be stored using unstructured interleavers, these interleavers require a total of $\sum_{k=0}^{4} 320 \cdot 2^k \left\lceil \log_2 \left(320 \cdot 2^k \right) \right\rceil \approx 1.2 \cdot 10^5$ memory cells for storage. The reduction using the expanded odd-even symmetric interleaver is thus 74% in terms of storage area. Furthermore, switching between interleaver sizes is very easily implemented by shifting the bits in the interleaver index counter to the left, one position for each step down in interleaver size. Simulations show that the presented interleaver constraints have essentially no influence on the error correcting performances of the codes.

References

[1] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-Codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, October 1996.

[2] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Transactions on Communications*, vol. 44, pp. 591–600, May 1996.

[3] S. Benedetto, R. Garello, and G. Montorsi, "A search for good convolutional codes to be used in the construction of turbo codes," *IEEE Transactions on Communications*, vol. 46, Sep. 1998.

[4] D. Divsalar and R. J. McEliece, "Effective free distance of turbo codes," *Electronic Letters*, vol. 32, Feb. 1996.

[5] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations." TDA progress report 42-122, Jet propulsion Lab., Pasadena, CA, August 1995.

[6] S. Crozier, J. Lodge, P. Guinand, and A. Hunt, "Performance of turbo codes with relative prime and golden interleaving strategies," in *Sixth International Mobile Satellite Conference*, Ottawa, Canada, June 1999.

[7] M. Eroz and A. R. Hammons, "On the design of prunable interleavers for turbo codes," in *Vehicular Technology Conference*, Houston, USA, May 1999.

[8] A. S. Barbulescu and S. S. Pietrobon, "Terminating the trellis of turbo-codes in the same state," *Electronics Letters*, vol. 31, pp. 22–23, January 1995.

[9] J. Hokfelt, O. Edfors, and T. Maseng, "Interleaver design for turbo codes based on the performance of iterative decoding," in *IEEE International Conference on Communications*, Vancouver, BC, Canada, June 1999.

[10] A. Henriksson, J. Hokfelt, and O. Edfors, "Evaluation of an interleaver design algorithm for turbo codes in UMTS." Tdoc 419/98, ETSI SMG2 UMTS L1 Expert Group, Meeting no 7, Sweden, Oct. 1998.

Paper VI

On the Theory and Performance of Trellis Termination Methods for Turbo Codes

Johan Hokfelt, Ove Edfors and Torleiv Maseng

Submitted to
IEEE Journal on Selected Areas in Communications
Submitted April 2000

On the Theory and Performance of Trellis Termination Methods for Turbo Codes

Johan Hokfelt, Ove Edfors and Torleiv Maseng

Abstract

The performance of a Turbo code is in general severely degraded if no trellis termination is employed. This paper investigates the implications of the choice of trellis termination method for Turbo codes, and explains the origin of the performance degradation often experienced without trellis termination. An efficient method to derive the distance spectrum of Turbo codes for different trellis termination methods is presented. Further, we present interleaver design rules that are tailored to each termination method. Using interleavers designed with these restrictions, we demostrate that the performance difference between various termination methods are very small, including no trellis termination at all. For example, we demonstrate a Turbo code with a 500-bit interleaver that exhibit no sign of an error floor for frame error rates as low as 10^{-8}*, even though no trellis termination is employed.*

1 Introduction

Turbo codes are in general implemented as two recursive convolutional encoders in parallel, where the input to the second encoder is an interleaved version of the original information sequence [1, 2]. At the beginning of each information block, the encoders are initialized to their zero-states. Similarly, it is desirable to force the encoders back to the zero-state at the end of the information block, an operation known as trellis termination. For feedforward convolutional encoders, this is readily achieved by appending zeros, known as tail bits, at the end of the encoder input sequence. However, the recursive property of the constituent encoders used in Turbo codes implies a state-dependency on these tail bits, and hence, individual tail sequences are required for each constituent encoder.

The problem of trellis termination for Turbo codes has been addressed by many authors. In the original Turbo code proposed by Berrou et al. [2], the trellis of the first encoder was terminated in the zero state while the second encoder was truncated in an unknown state. Since then, various techniques

have been presented which allow both encoders to be terminated in their zero states. Examples of such techniques are: location of input positions that separately influence the final states of both encoders [3], interleaver restrictions that terminate the second encoder in the same state as the first encoder [4, 5], tail-biting Turbo codes [6], and post interleaver flushing [7]. Naturally, it is also an option to truncate both encoders without any trellis termination at all, as investigated for example in [8, 9]. Additional reports on trellis termination for Turbo codes are found in, among others, [10, 11].

The performance of a trellis termination methods is, to various extents, dependent on the particular interleaver used in the Turbo encoder – a fact rarely acknowledged in the literature. This dependency is the result of *interleaver edge effects* [12]. These edge effects degrade the distance spectrum of Turbo codes in situations where at least one of the encoder trellises is truncated in an unknown state. The degree of distance spectrum degradation is highly dependent on the particular choice of interleaver. Thus, the performance of a trellis termination method is the result of the combination of the termination method *and* the edge effects present for the particular interleaver used.

This paper describes interleaver edge effects in detail, and demonstrates how different termination methods are unequally sensitive to the phenomenon. A method to calculate the distance spectrum of Turbo codes for different trellis termination methods is presented. Further, we describe how the interleaver edge effects can be avoided by means of sophisticated interleaver design, hereby significantly reducing the need for trellis termination. Using interleavers designed specifically for each termination method, we investigate the error correcting performances of different termination schemes. The investigated trellis termination methods are: no termination at all, termination of the first encoder only, termination of both encoders with post interleaver flushing [7], termination of both encoders using both self-terminating interleavers [4, 5] and dual termination [3].

A typical Turbo code encoder is depicted in Fig. 1. The constituent encoders are rate 1/1 recursive convolutional encoders with m memory elements and $M = 2^m$ states. The two encoders are linked by an interleaver of length N, so that every block of N information bits entering the second constituent encoder is an interleaved, or permuted, version of the original N-bit information block. Depending on the degree of puncturing performed on the two parity sequences, the overall code rate can be varied from 1/3 to 1/1.

Due to the interleaver, the overall state space of the encoder becomes prohibitively large for maximum likelihood decoding. However, a reasonably low complexity decoding is obtained by iteratively decoding the codes generated by the first and second constituent encoders, respectively. The decoding of the constituent codes requires a soft-in/soft-out decoding algorithm, providing *a posteriori* probabilities (APPs) of the symbols in the decoded sequence. One such algorithm is the BCJR algorithm [13], which provides optimal APPs on a symbol-by-symbol basis. Due to the comparably high complexity of the BCJR

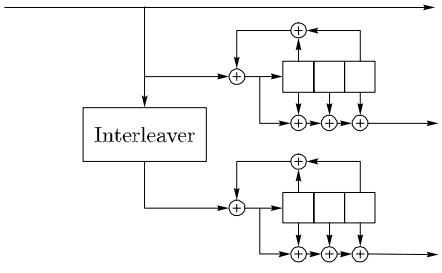


Figure 1: Turbo encoder structure.

algorithm, soft-output algorithms based on the Viterbi algorithm (SOVA) has been developed [14, 15]. In our simulations of Turbo codes, we use 15 decoding iterations employing the full BCJR algorithm, implemented in the logarithmic domain [16]. Further, all transmission is over the additive white Gaussian noise (AWGN) channel.

This paper is organized as follows. In Section 2, we restate the theory required to obtain the distance spectrum of the ensemble of Turbo codes, and present an efficient method to include the effects of different trellis termination methods in these calculations. In Section 3, we investigate the implications of using specific interleavers, whose performances deviate significantly from that of the ensemble of all interleavers. In Section 4, we discuss the issue of decoder initialization for the case when the trellis is truncated in an unknown state. Finally, we summarize our investigation in Section 5, which is followed by an appendix describing interleaver design restrictions that are tailored for specific choices of trellis termination methods.

2 Influence of Trellis Termination on the Distance Spectrum

The standard method used for performance assessment of Turbo codes, as for conventional block- and convolutional codes, is to calculate the Hamming distance spectrum of the code. With the distance spectrum, or, equivalently, the weight distribution, lower and upper bounds on the error correcting performance can be derived. However, the calculation of the distance spectrum of a Turbo code involves taking the particular interleaver used into account, a task that becomes prohibitively complex for reasonably large interleaver sizes. A less computationally demanding method was introduced by Benedetto et al. in [17], where they presented a method to derived the average distance spectrum for the ensemble of all interleavers of a certain length. In this section we summarize the method presented in [17], and present an efficient method to include

the influences of different trellis termination methods.

Benedetto et al. introduced [17] the *input-redundancy weight enumerating function* (IRWEF)

$$A\left(W,Z\right) \triangleq \sum_{w=0}^{N} \sum_{j=0}^{n-N} A_{w,j} W^{w} Z^{j}, \tag{1}$$

for an (n, N)-code, where N is the length of the interleaver and n is the codeword length. $A_{w,j}$ is the number of codewords with input weight w and parity weight j, and W and Z are dummy variables. At this point, we restrict the coefficients $A_{w,j}$ to include trellis paths that start from the zero state and end in the zero state after N trellis transitions. The trellis paths that do not end in the zero state will be treated separately in Section 2.1.

Since both constituent encoders in a Turbo code share the same input weight w, every codeword that belong to a Turbo code is a combination of two constituent code codewords that both result from weight w input sequences. It is therefore convenient to define the *conditional weight enumerating function* (CWEF)

$$A_{w}\left(Z\right) \triangleq \sum_{j=0}^{n-N} A_{w,j} Z^{j}, \qquad w = 0, 1, \ldots, k,$$

which enumerates the number of codewords of various parity weights j, conditioned on a certain input-weight w. The CWEF of the first and second constituent encoders are denoted $A_{w}^{C_1}\left(Z\right)$ and $A_{w}^{C_2}\left(Z\right)$ respectively, and the CWEF of the overall Turbo code $A_{w}^{TC}\left(Z\right)$. Introducing a probabilistic interleaver construction called a *uniform interleaver*, for which all distinct mappings are equally probable, Benedetto et al. obtained the CWEF of the ensemble of all Turbo codes using interleavers with length N as

$$A_{w}^{TC}\left(Z\right) = \frac{A_{w}^{C_1}\left(Z\right) A_{w}^{C_2}\left(Z\right)}{\binom{N}{w}},$$

where $1 \Big/ \binom{N}{w}$ is the probability that a specific weight-w sequence is mapped to another, specific, weight-w sequence. Finally, the multiplicities a_d of codewords with Hamming weight d is achieved as

$$a_{d} = \sum_{w=1}^{N} A_{w,d-w}^{TC}, \tag{2}$$

where $A_{w,d-w}^{TC}$ are the coefficients in the Turbo code CWEF, i.e. $A_{w}^{TC}\left(Z\right) \triangleq \sum_{j=0}^{n-N} A_{w,j}^{TC} Z^{j}$. Note that, due to the averaging of the ensemble of distance spectra, each multiplicity a_d and $A_{w,j}^{TC}$ are not necessarily integers.

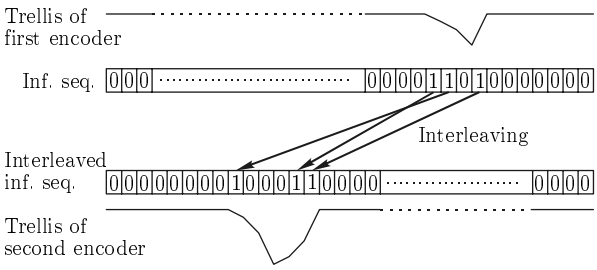


Figure 2: Example of an interleaver and an information sequence producing two low weight parity sequences, and thus a low weight code word.

The distance spectrum obtained with (2) can be used to derive an upper bound on the error correcting performance. For AWGN channels and maximum likelihood decoding, the error rate performance is upper bounded by

$$P_{FER} \leq \sum_{d} a_d Q\left(\sqrt{d\frac{2RE_b}{N_0}}\right), \tag{3}$$

where P_{FER} is the frame or block error rate, $Q(\cdot)$ is the upper tail probability of a normalized Gaussian random variable, R is the code rate, E_b is the energy per information bit and N_0 is single-sided noise spectral density. Note that the bound (3) is valid for the ensemble of all interleavers with a certain length; a single specific interleaver may perform both worse and better than the ensemble upper bound. For the bit error rate performance, the union bound is

$$P_{BER} \leq \sum_{w} \sum_{j} \frac{w}{N} A_{w,j}^{TC} Q\left(\sqrt{(w+j)\,\frac{2RE_b}{N_0}}\right). \tag{4}$$

When deriving the CWEF of the constituent codes of Turbo codes, it is common to take only the error events that end up in the zero state into account. This corresponds to taking only zero-terminating input sequences into consideration. Figure 2 depicts such a combination of interleaver and input sequence, for which a weight-3 input sequence is zero-terminating both before and after interleaving. Depending on the method of trellis termination, codewords might also exist that originate from trellis paths that do not end up in the zero state after N trellis transitions. Since these codewords can be thought of as being the result of the truncation imposed by the interleaver, they are referred to as *interleaver edge effects* [12]. In the next section we describe an efficient method for taking the interleaver edge effects into account, for various Turbo code trellis termination methods.

2.1 Interleaver Edge Effects

Interleaver edge effects refer to the implications on the distance spectrum resulting from the block partitioning of the input sequence, as the result of a limited-length interleaver. Due to this truncation, low weight parity words can be generated even though the encoder input sequences do not force the encoders back to the zero states. In terms of weight enumerating functions, this means that we require knowledge not only of the number of trellis paths that lead to the zero state after the last transition, but also of the number of paths that lead to the other states. This can be obtained by partitioning the IRWEF in (1) into a *state-dependent* counterpart, which enumerates the number of paths that lead to trellis state s, having input weight w and parity weight j. An efficient method to find the state-dependent IRWEF of a convolutional encoder valid after t trellis transitions is to extend the IRWEF of the same encoder obtained for $t-1$ transitions. Let $A_{t,s,w,j}$ denote the number of paths that lead to state s after t trellis transitions, having input weight w and parity weight j. Based on the encoder trellis, the state dependent IRWEF is recursively calculated as

$$A_{t,s,w,j} = \sum_{u=0}^{1} A_{t-1,S(s,u),w-u,j-P(S(s,u),u)},$$

where $S(s,u)$ is the state that leads to state s when the input symbol is u, and $P(S(s,u),u)$ is the parity weight of the corresponding trellis transition. The recursive procedure is initialized with $A_{0,s,w,j} = 0 \ \forall s,w,j$, except for $A_{0,0,0,0} = 1$, which corresponds to an encoder initialized in the zero state.

Let $E_{w,j}^{C_1}$ and $E_{w,j}^{C_2}$ denote the multiplicities of codewords that correspond to trellis paths that do not end up in the zero state after encoding length-N input blocks, for constituent code C_1 and C_2 respectively. These multiplicities form the corresponding CWEFs according to $E_w^{C_l}(Z) = \sum_{j=0}^{n-N} E_{w,j}^{C_l} Z^j, l = 1,2$. The overall CWEFs, including both zero-terminating codewords and edge effects, are then obtained as $\tilde{A}_w^{C_l}(Z) = A_w^{C_l}(Z) + E_w^{C_l}(Z)$, and the resulting CWEF for the Turbo code is

$$\tilde{A}_w^{TC}(Z) = \frac{\left(A_w^{C_1}(Z) + E_w^{C_1}(Z)\right)\left(A_w^{C_2}(Z) + E_w^{C_2}(Z)\right)}{\binom{N}{w}}. \tag{5}$$

When comparing different termination methods, it is illustrative to separate the part of the distance spectrum that originates from edge effects, i.e.

$$E_w^{TC}(Z) = \frac{A_w^{C_1}(Z) E_w^{C_2}(Z) + E_w^{C_1}(Z) A_w^{C_2}(Z) + E_w^{C_1}(Z) E_w^{C_2}(Z)}{\binom{N}{w}}, \tag{6}$$

so that $\tilde{A}_w^{TC}(Z) = A_w^{TC}(Z) + E_w^{TC}(Z)$. The difference in the $\tilde{A}_w^{TC}(Z)$:s for different trellis termination methods is in the way the $E_{w,j}^{C_l}$:s are calculated. This is treated in the subsequent paragraphs.

We separate the trellis termination methods into three principal classes:

1. No termination of either constituent encoder.

2. Termination of the first constituent encoder.

3. Termination of both constituent encoders.

Note that these classes refer to the trellis situations after encoding sequences that are N bits long, i.e. the length of the interleaver.

Class I. No trellis termination

With *no trellis termination* of either constituent encoder, the multiplicities of codewords that correspond to interleaver edge effects are calculates by summing the number of paths that end in non-zero states after N trellis transitions, i.e.

$$\begin{aligned} E_{w,j}^{C_1} &= \sum_{s=1}^{2^{m_1}-1} A_{N,s,w,j}^{C_1}, \text{ and} \\ E_{w,j}^{C_2} &= \sum_{s=1}^{2^{m_2}-1} A_{N,s,w,j}^{C_2}, \end{aligned}$$

where m_1 and m_2 are the number of memory elements in encoder 1 and 2, respectively. The overall distance spectrum including edge effect codewords, $\tilde{A}_w^{TC}(Z)$, as well as the edge effect distance spectrum only, $E_w^{TC}(Z)$, is calculated using (5) and (6).

An important and frequently used subclass of no trellis termination is *post interleaver flushing*, proposed in [7]. With this method, both encoders are flushed independently of each other, after encoding their N-bit input sequences. The combination of the weight spectra of the constituent encoders is then similar to the case of no trellis termination, since the trellises are not terminated by the end of their length-N input sequences. However, extra codeword weight is added as a consequence of the encoder flushings. This is accounted for by adding the weight of the flush bits and the corresponding parity bits to the parity weight in the IRWEFs. More precisely,

$$\begin{aligned} E_{w,j}^{C_1} &= \sum_{s=1}^{2^{m_1}-1} A_{N,s,w,j-F_1(s)}^{C_1} \\ E_{w,j}^{C_2} &= \sum_{s=1}^{2^{m_2}-1} A_{N,s,w,j-F_2(s)}^{C_2}, \end{aligned}$$

where $F_l(s)$, $l = 1, 2$, is the sum of the weight of the flush bits and parity bits generated when forcing the encoder to the zero state from state s.

Class II. Termination of the first encoder

By appending m_1 tail bits to the input sequence so that the first encoder is terminated in the zero state, the edge effect codewords are entirely removed from the first encoder. Note that the tail bits are included in the sequence that enters the interleaver, and that their Hamming weight is included in the input weight w. For the second encoder, the situation is identical to the case of no trellis termination. Hence,

$$\begin{aligned} E_{w,j}^{C_1} &= 0 \\ E_{w,j}^{C_2} &= \sum_{s=1}^{2^{m_2}-1} A_{N,s,w,j}^{C_2}. \end{aligned}$$

Class III. Termination of both encoders

It is also possible to terminate both constituent encoders in the zero states. Two different ways of achieving this is reported in the literature:

1. By imposing interleaver restrictions, the second encoder can be forced to end up in the same state as the first encoder [4, 5]. It is then sufficient to append tail bits according to termination class II, in order to terminate both encoders in the zero states. The necessary interleaver restrictions are

$$\pi(i) \bmod L = i, \qquad i = 1, 2, \ldots, N, \tag{7}$$

   where $\pi(i)$ is the position of input bit i after interleaving, and L is the period of the impulse response of the constituent encoders (assumed to be identical). Following the terminology in [5], this method is referred to as using *self-terminating interleavers*.

2. By identifying specific, interleaver dependent, input positions it is possible to force the constituent encoders to their zero states independently of each other [3]. This is achieved without any restrictions on the choice of interleaver, but with a slight increase in the number of input bits dedicated for trellis termination (n termination bits are required, where $\max(m_1, m_2) \leq n \leq m_1 + m_2$). Following the terminology in [3], this method is referred to as *dual termination*.

Even though the second method result in a slightly higher rate loss, it has the advantage of not imposing restrictions on the interleaver design.

With both encoders terminated in their zero states, all edge effect codewords are entirely removed. Consequently, $E_{w,j}^{C_1} = E_{w,j}^{C_2} = 0$, so that $E_w^{TC}(Z) = 0$ and $\tilde{A}_w^{TC}(Z) = A_w^{TC}(Z)$.

Figure 3(a) shows the lower parts of the edge effect distance spectra for the described termination methods, for a Turbo code with interleaver length 500

bits and constituent encoders with feedback and generator polynomials 15_8 and 17_8, respectively. As expected, "no termination" results in the worst distance spectrum, significantly improved by terminating the first encoder and post interleaver flushing. Naturally, since termination of both encoders entirely removes the interleaver edge effects, there exist no edge effect codewords. Figure 3(b) shows the corresponding total distance spectra, including both ordinary and edge effect codewords. Figure 4 shows the union bound on the frame error rates achieved with (3), corresponding to the distance spectra in Figure 3(b). As expected, the union bound diverges at a signal-to-noise ratio close to the cut-off rate, which is 2.03 dB for rate 1/3 coding. Also shown in Figure 4 are the simulated performances of a large number of randomly chosen interleavers, using the described trellis termination methods. The simulated performances rank in the same order as predicted by the derived distance spectra; however, the simulation results are actually *above* the derived union bounds. This is not an inconsistency, bearing in mind that the union bound is valid for maximum likelihood decoding, while the simulation results are obtained with suboptimal iterative decoding.

3 Interleaver Edge Effects with Non-Uniform Interleavers

Even though calculating the average distance spectrum for the different termination methods gives a good insight to the issues of trellis termination, it is important to keep in mind what the method does – i.e. the averaging over the entire ensemble of interleavers having a certain length. As an example of when the averaging method coincide poorly with individual interleaver performances, we study in this section ordinary block interleavers. With such interleavers, in which the input sequence are written row-wise and read column-wise, the very last input position remains at the last position also after interleaving. Hence, with no trellis termination, the minimum distance of the code is 3 regardless of the interleaver length and the choice of component encoders. If post interleaver flushing is employed, the minimum distance is increased to 13 (for feedback and generator polynomials 15_8 and 17_8, respectively), again regardless of the interleaver length. Even though 13 is considerably larger than 3, it is still a poor minimum distance for interleavers of reasonable lengths, say above 100 bits. As a comparison, the minimum distances when terminating the first encoder and terminating both encoders is 28 (with multiplicity 473), when using a 500-bit block interleaver with 20 rows and 25 columns. Figure 5 shows simulated frame error rate performances for these codes, along with their respective free distance asymptotes [18]. The poor performance for no trellis termination follows, as described, from the severe edge effect that an ordinary block interleaver results in.

The particular influence from the edge effect that arise when using ordi-

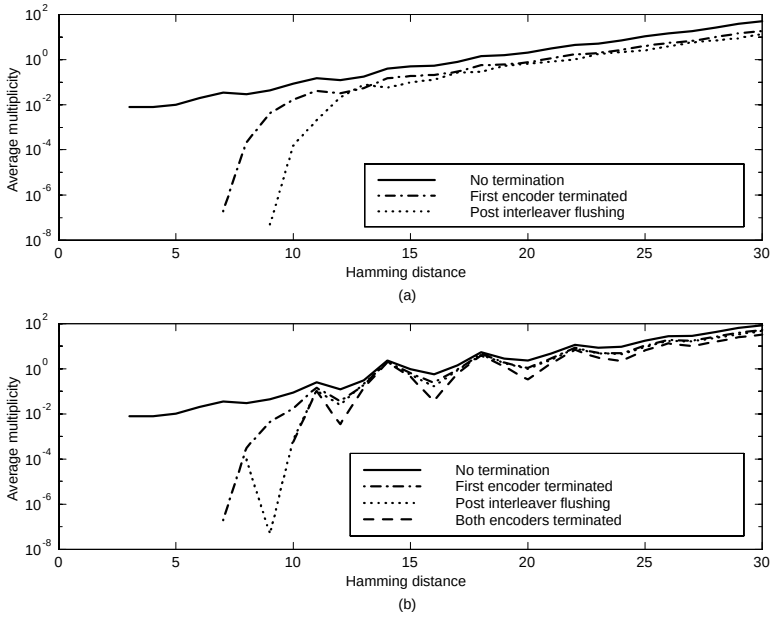


Figure 3: Average weight distributions for Turbo codes using the ensemble of all 500-bits interleavers, for four different trellis termination methods. The constituent encoders use feedback and parity polynomials 15_8 anf 17_8, respectively. (a) The part of the weight distribution that result from interleaver edge effects only. (b) The entire weight distribution, including both edge effect codewords and ordinary codewords for which both encoders end in the zero state.

nary block interleavers can be singled out by comparing with the performances achieved with reversed block interleavers [19]. When using reversed block interleavers, the interleaved sequence is read in reversed order, starting from the last column and the last row. This procedure effectively removes the interleaver edge effects, thereby significantly improving the performance of no termination and post interleaver flushing. The performance of a 500-bit reversed block interleaver, using the different termination methods, are shown in the upper part of Figure 6. As seen, there is essentially no difference in the performances of the different termination methods.

It is well known that it is possible to design interleavers that perform better than both ordinary block interleavers and reversed block interleavers. Most interleaver design methods described in the literature are based on design criteria that strive to avoid interleaver mappings that correspond to low weight codewords produced by zero-terminating input sequences, as exemplified in Figure 2. However, interleavers designed without considerations to the edge effects

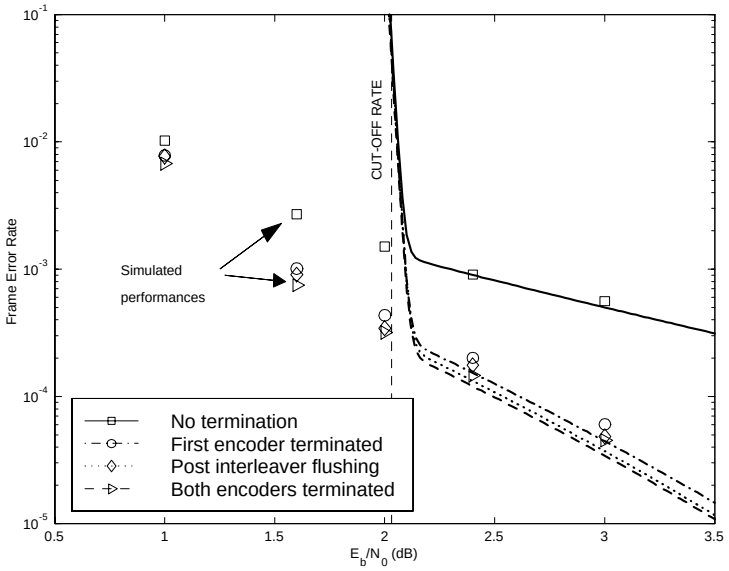


Figure 4: Union bound off the frame error rate performances achieved with the different trellis termination methods, for 500-bit interleavers. The constituent encoders use feedback and generator polynomials 15_8 anf 17_8, respectively. Also shown are simulated performances, achieved by simulating a large number of randomly chosen interleavers.

may result in low weight edge effect codewords. As a consequence, comparisons between different trellis termination methods may be corrupted due to the stochastic presence of the edge effects.

In order to evaluate the performances of the trellis termination methods when using interleavers with good performance, we have designed interleavers with specific criteria that target the interleaver edge effects. The edge effect criteria are used as a supplement to an interleaver design algorithm that use both ordinary distance spectrum criteria (such as in [20]), as well as a criterion based on the correlation properties of the extrinsic information [21]. The interleaver design restrictions that avoid interleaver edge effects are described in Appendix A.

The performance of 500-bit interleavers designed specifically for each termination method are shown in Figure 6, by the two lower sets of curves. The best performances are achieved with dual termination, termination of the first encoder, and post interleaver flushing. The error floors of these codes are not reached even for bit error rates as low as 10^{-9}. The slight degradation in performances achieved with no termination and self-terminating interleavers are

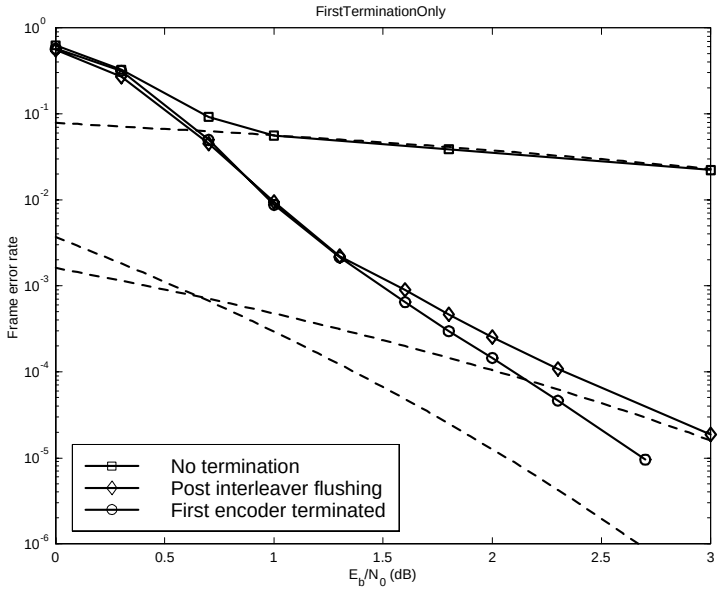


Figure 5: Simulated frame error performances for Turbo codes using 500-bit block interleavers (20 rows by 25 columns) and feedback and parity polynomials 15_8 and 17_8. Three termination methods are shown: no termination, post interleaver flushing, and termination of the first encoder.

both due to implications regarding the interleaver design. For the case of no termination, the number of edge effects codewords to avoid in the interleaver design becomes very large, resulting in a high computational complexity. On the other hand, when designing self-terminating interleavers, the interleaver restrictions (7) severely limit the design freedom. Therefore, it becomes difficult to design self-terminating interleavers that perform as well as interleavers designed without these restrictions.

An important parameter regarding the computational complexity of interleaver design is the length of the period of the encoder impulse responses, i.e. L. The larger the length of the period L, the smaller the computational complexity required to design an interleaver with a specified minimum distance. Further, the multiplicities of codewords with weights just above the minimum distance will in general be larger for a code with lower L. Figure 7 shows the performances of two Turbo codes that use constituent encoders with 3 and 4 memory elements respectively. The corresponding periods of the encoder impulse responses are 7 and 15. This indicates that we should expect a better asymptotic performance for the Turbo code with 4 memory elements, which is supported by the performances shown in Figure 7. Note that there in no indi-

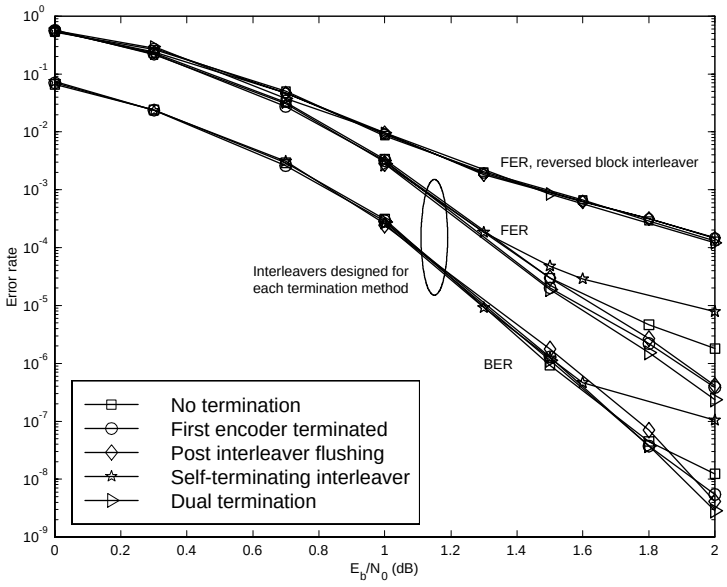


Figure 6: Error rate performances of Turbo codes using 500-bit interleavers that do not result in interleaver edge effects. The constituent encoders use feedback and parity polynomials 15_8 and 17_8, respectively.

cation of an error floor for this Turbo code, even though no trellis termination is employed. Further, note that the Turbo code with fewer memory elements, and thus lower decoding complexity, perform better for a rather large and important part of the SNR range. This behavior has been observed for example in [22]; however, the explanation for this somewhat surprising behavior remains to be presented.

4 Discussion on Decoding Initialization

One of the issues of trellis termination for Turbo codes is the initialization of the backward recursion in the BCJR algorithm, see for example [23, 24, 25]. In essence, the *a posteriori* probabilities produced by the BCJR algorithm for a certain trellis transition is influenced by three quantities used in the decoding, usually denoted α, β and γ. The α- and β-values can be interpreted as state probabilities, while the γ-values represent branch transition probabilities in the trellis. The γ-values at a given time t are based solely on the channel statistics received for symbol t. In contrast, the state probabilities α at time t summarizes the received channel statistics from time 0 up to t. Similarly, the β-values are based on the channel statistics from time $t+1$ till the end of the block, i.e. to

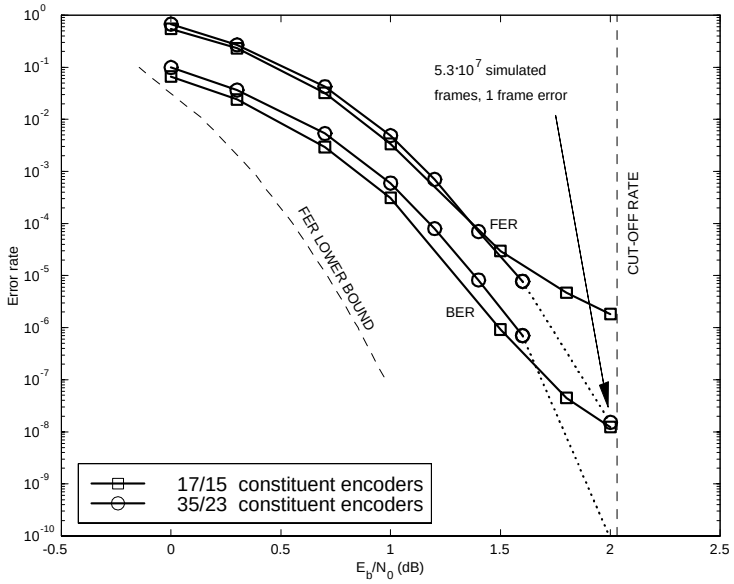


Figure 7: Performance of Turbo codes without trellis termination, for constitutent encoders with 3 and 4 memory elements. Both encoders use 500-bit interleavers designed for no trellis termination.

time N. The α- and β-values are computed recursively, a procedure referred to as the forward and the backward recursion, respectively.

Since the constituent encoders are initialized in their zero states before each information block is encoded, it is straightforward to initialize the α-values for time 0: state zero is assigned probability 1, while all the other states are assigned probability 0. If the constituent encoders are terminated in their zero states, the β-values are initialized in the same manner. However, when using no trellis termination, or termination of the first encoder only, there are at least one trellis for which the final state is unknown.

At least two methods to deal with the state-uncertainty has been proposed in the literature. In the original Turbo code paper [1], Berrou et al. proposed to initialize the β-values with equal probabilities, i.e. $1/M$, where M is the number of states in the trellis. Another possibility is to use the final α-values obtained from the forward recursion, as evaluated for example in [23]. However, this method is in violation with the derived expression for the β-values, which states that they should depend only on channel statistics from the present time up to the end of the block. By using the final α-values for initialization, the β-values become dependent on the channel statistics received during the entire block.

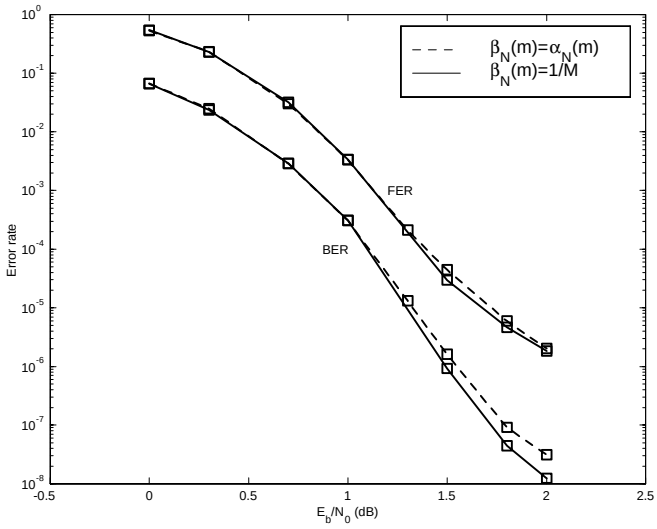


Figure 8: Comparison of the frame- and bit error rate performances for two different initialization methods for the backward recursion in the BCJR algorithm, when no trellis termination is employed. Interleaver length: 500 bits, encoder polynomials: $(17/15)_8$.

Figure 8 shows simulated error rate performances of the previously used Turbo code using a 500-bit interleaver with no trellis termination and constituent encoders with polynomials 15_8 (feedback) and 17_8 (parity). The initialization for the backward recursion, i.e. the β-values, are performed with the aforementioned methods. For low signal-to-noise ratios, there is no detectable difference in the error correcting performances. However, at medium and high SNRs, the decoder that uses equiprobable initial states perform better than the decoder that use the final α-values for initialization, especially in terms of bit error rate. These simulations are in agreement with the above reasoning, claiming that the initial β-values should not depend on the α-values from the forward recursion.

5 Conclusions

In this paper, we have discussed central aspects of trellis termination methods for Turbo codes. An efficient method to calculate Turbo code distance spectra for the ensemble of interleavers, including the effects of the choice of termination method, has been presented. This method illustrates the result of *interleaver edge effects*, thus allowing for basic understanding of the properties

that govern the performance of a specific termination method. For the ensemble of interleavers, it is evident that it is essential for the code performance that some kind of trellis termination is used.

The performance differences between commonly used termination methods, such as *termination of the first encoder only, post interleaver flushing,* and *dual termination*, is in general small. However, examples are shown for which the differences are substantial, explained by the fact that the different methods are not equally sensitive to interleaver edge effects. Hence, we stress the importance of the combination of the choice of trellis termination method and the choice of interleaver. In particular, we present interleaver design guidelines that are tailored to the chosen termination method. It is demonstrated that for interleavers designed with such design criteria, the performances of the different termination methods are nearly indistinguishable, also when using no trellis termination at all. In fact, we demonstrate Turbo codes using no trellis termination and 500–bit interleavers that show no sign of an error floor, even at frame error rates as low as 10^{-8}.

Appendix A

In this appendix we present interleaver design restrictions that aim to avoid interleaver edge effects. The restrictions on the interleaver design depends on the chosen method of trellis termination. To start with, since termination of both encoders completely removes all edge effects, this method requires no interleaver design restrictions. For the other two classes of trellis termination, the interleaver design includes restrictions based on the particular choice of termination method.

For the purpose of describing the interleaver design restrictions, we distinguish between three principal types of interleaver edge effects: (Type I) truncation where the first encoder is in a non-zero state, (Type II) truncation where the second encoder is in a non-zero state, and (Type III) truncation where both encoders are in non-zero states. The three types of edge effects are illustrated in Figure 9.

Let the component encoders be represented by the generators $G_1(D) = \frac{G_{1p}(D)}{G_{1f}(D)}$ and $G_2(D) = \frac{G_{2p}(D)}{G_{2f}(D)}$, respectively, and their input sequences by $X(D)$ and $X'(D)$ ($X'(D)$ is thus the interleaved version of $X(D)$). All zero-terminating input sequences are divisible by the feedback polynomial, i.e. $X(D) = Y(D)\,G_f(D)$, where $\deg Y(D) \leq \deg X(D)$. Thus, the parity sequence is $X(D)\,G_1(D) = Y(D)\,G_{1p}(D)$. Further, let $i_1, i_2, i_3, \ldots$ and $i'_1, i'_2, i'_3, \ldots$ denote the positions of the first, second, third, and so on, input ones before and after interleaving, so that $X(D) = \sum_{k=0}^{N-1} D^{i_k}$ and $X'(D) = \sum_{k=0}^{N-1} D^{i'_k}$. Finally, the Hamming weight of a sequence, or the corresponding polynomial representation, is denoted $w_H(\cdot)$.

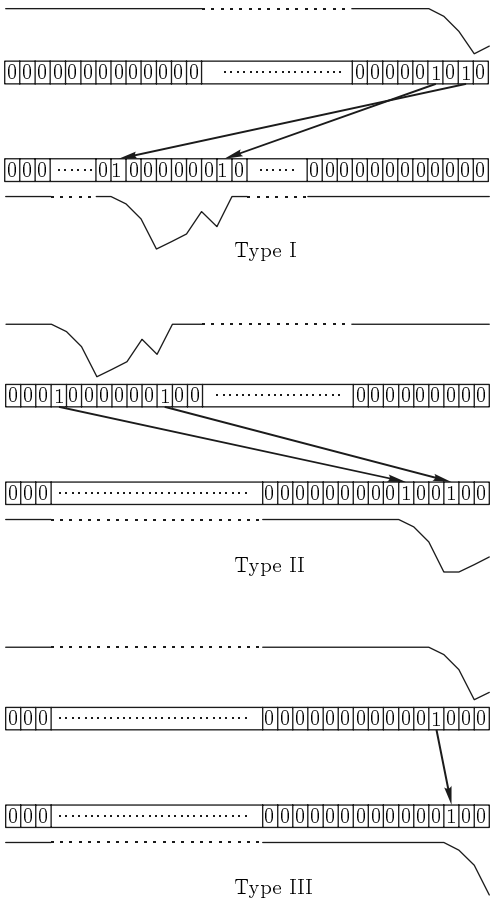


Figure 9: Examples of three types of interleaver edge effects: truncation of the first trellis in a non-zero state (Type I), truncation of the second trellis in a non-zero state (Type II), and truncation of both trellises in non-zero states (Type III).

The total codeword weight of a Turbo code codeword is

$$d = w + w_H\left(\langle X\left(D\right)G_1\left(D\right)\rangle_N\right) + w_H\left(\langle X'\left(D\right)G_2\left(D\right)\rangle_N\right),$$

where w is the input weight, and $\langle\cdot\rangle_N$ denotes truncation of all terms in a polynomial for which the D-exponent is larger than or equal to N. The complexity required to check (and avoid) all the necessary interleaver mappings grows rapidly with increasing input weight w. Fortunately, the probability of edge effects resulting in low weight codewords decreases with increased input weight. This is especially true when designing interleavers with some sort of spreading criterion, such as the S-random design [12] or the correlation criterion [21]. With such criteria, we have found it sufficient to check for mappings of input sequences with weight $w = 2, 3$ and 4. Further, the spreading criterion relieve the number of input sequences that need to be checked for $w = 3$ and 4. Consequently, the interleaver design restrictions we present in the following relies on a spreading criterion being used in the interleaver design.

Figure 10 shows the principal appearances of Type I edge effect interleaver mappings that a spreading criterion fails to avoid. For all such input sequences, the interleaver design should ensure that

$$w_H\left(\langle X\left(D\right)G_1\left(D\right)\rangle_N\right) + w_H\left(X'\left(D\right)G_2\left(D\right)\right) < d_{\text{design}} - w,$$

where d_{design} is the targeted minimum distance of the overall Turbo code. The input sequences $X\left(D\right)$ and $X'\left(D\right)$ are constrained to be on the forms given in the upper part of Table 1, which summarizes the characteristics of Type I edge effect input sequences. For weight-2 and weight-4 input sequences, the sequence characterization is based solely on the periods of the encoder impulse responses, i.e. L_1 and L_2. For weight-3 inputs, the zero-terminating sequences are instead characterized by a set of fundamental zero-terminating weight-3 sequences. All the existing weight-3 zero-terminating input sequences can be expressed using one of these fundamental sequences. These are in turn represented by the sets of constants $\phi_{l1}$and $\phi_{l2}, l = 1, 2, \ldots, l_{\max}$where $l_{\max}$ is the number of such fundamental sequences that exist for a certain feedback polynomial. More precisely, every zero-terminating weight-3 input can be written as $D^n\left(1 + D^{\phi_{l1}+k_1 L} + D^{\phi_{l2}+k_2 L}\right)$, where n, k_1 and k_2 are integers. Values of ϕ_{l1}, ϕ_{l2}, for the most common encoders with 2–4 memory elements, are given in Table 2. For example, an encoder with feedback polynomial $1 + D^3 + D^4$ (23 in octal representation), every zero-terminating weight-3 input sequence can be written on one of the forms $D^n\left(1 + D^{1+7k_1} + D^{12+7k_2}\right)$, $D^n\left(1 + D^{2+7k_1} + D^{9+7k_2}\right)$ or $D^n\left(1 + D^{5+7k_1} + D^{10+7k_2}\right)$.

Type II and Type III edge effects are treated the same way as Type I edge effects. The characterization of the input sequences that correspond to Type II and Type III edge effects is given in the middle and lower part of Table 1.

The interleaver design constraints presented here can be implemented with sufficiently low computational complexity to allow for design of large interleavers. The most demanding constraints arise when *no trellis termination* is

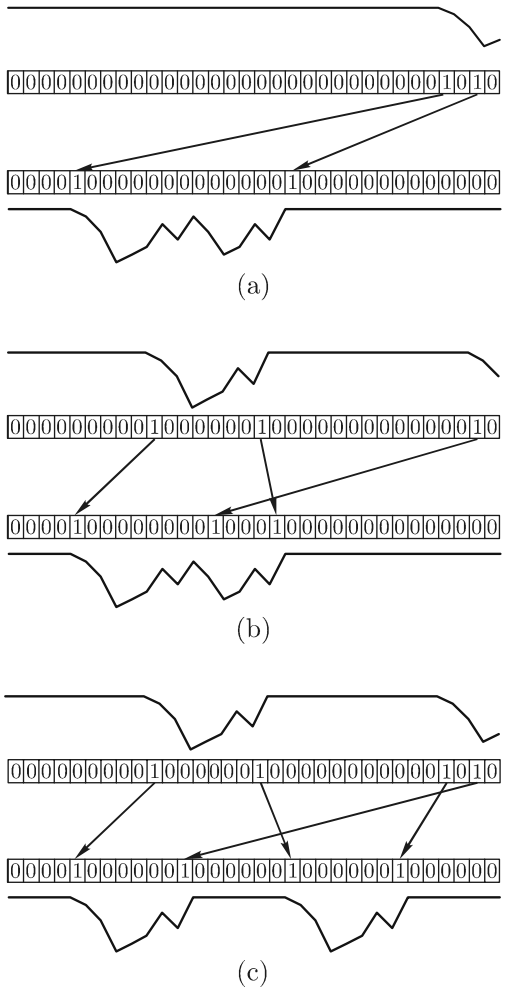


Figure 10: Examples of Type I edge effect interleaver mappings not avoided by a spreading interleaver design criterion, for (a) weight-2, (b) weight-3, and (c) weight-4 input sequences.

	w	Characteristics of $X(D)$	Characteristics of $X'(D)$
Type I	2	$i_2 - i_1 \bmod L_1 \neq 0$	$i'_2 - i'_1 \bmod L_2 = 0$
	3	$i_2 - i_1 \bmod L_1 = 0$	$D^{i'_1} + D^{i'_2} + D^{i'_3} = D^n(1 + D^{\phi_{l1}+k_1 L_2} + D^{\phi_{l2}+k_2 L_2})$
	4	$i_2 - i_1 \bmod L_1 = 0$ $i_4 - i_3 \bmod L_1 \neq 0$	$i'_2 - i'_1 \bmod L_2 = 0$ $i'_4 - i'_3 \bmod L_2 = 0$
Type II	2	$i_2 - i_1 \bmod L_1 = 0$	$i'_2 - i'_1 \bmod L_2 \neq 0$
	3	$D^{i_1} + D^{i_2} + D^{i_3} = D^n(1 + D^{\phi_{l1}+k_1 L_1} + D^{\phi_{l2}+k_2 L_1})$	$i'_2 - i'_1 \bmod L_2 = 0$
	4	$i_2 - i_1 \bmod L_1 = 0$ $i_4 - i_3 \bmod L_1 = 0$	$i'_2 - i'_1 \bmod L_2 = 0$ $i'_4 - i'_3 \bmod L_2 \neq 0$
Type III	2	$i_2 - i_1 \bmod L_1 \neq 0$	$i'_2 - i'_1 \bmod L_2 \neq 0$
	3	$i_2 - i_1 \bmod L_1 = 0$	$i'_2 - i'_1 \bmod L_2 = 0$
	4	$i_2 - i_1 \bmod L_1 = 0$ $i_4 - i_3 \bmod L_1 \neq 0$	$i'_2 - i'_1 \bmod L_2 = 0$ $i'_4 - i'_3 \bmod L_2 \neq 0$

Table 1: Characterization of low weight input sequences that can cause low weight edge effect codewords, not avoided by using a spreading criterion in the interleaver design.

used, since all the types of edge effects must be avoided. When terminating the first encoder, both Type I and Type III edge effects are removed, thus reducing the interleaver design complexity considerably.

.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo Codes," in *Proc. 1993 IEEE International Conference on Communication (ICC)*, pp. 1064–1070, Geneva, Switzerland, May 1993.

[2] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-Codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, October 1996.

[3] P. Guinand and J. Lodge, "Trellis termination for turbo encoders," in *17th Biennial Symp. On Communications*, Kingston, Canada, May 1994.

[4] A. S. Barbulescu and S. S. Pietrobon, "Terminating the trellis of turbo-codes in the same state," *Electronics Letters*, vol. 31, pp. 22–23, January 1995.

m	G_f	L	l	ϕ_{l1}	ϕ_{l2}
2	5	2	-	-	-
	7	3	1	1	2
3	11	3	-	-	-
	13	7	1	1	5
	15	7	1	1	3
	17	4	-	-	-
4	21	4	-	-	-
	23	15	1	1	12
			2	2	9
			3	5	10

m	G_f	L	l	ϕ_{l1}	ϕ_{l2}
4	25	6	1	2	4
	27	7	-	-	-
	31	15	1	1	4
			2	2	8
			3	5	10
	33	6	-	-	-
	35	7	-	-	-
	37	5	-	-	-

Table 2: Fundamental zero-terminating weight-3 sequences for recursive encoders with feedback polynomial G_f (in octal representation).

[5] M. Hattori, J. Murayama, and R. J. McEliece, "Pseudo-random and self-terminating interleavers for turbo codes," in *Winter 1998 Information Theory Workshop*, San Diego, USA, February 1998.

[6] S. Crozier, P. Guinand, J. Lodge, and A. Hunt, "Construction and performance of new tail-biting turbo codes," in *6-th International Workshop on Digital Signal Processing Techniques for Space Applications (DSP '98)*, Noordwijk, The Netherlands, September 1998.

[7] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *IEEE International Conference on Communications*, New York, USA, 1995.

[8] M. C. Reed and S. S. Pietrobon, "Turbo-code termination schemes and a novel alternative for short frames," in *Seventh IEEE International Symposium on Personal, Indoor and Mobile Communications*, New York, USA, 1996.

[9] J. Hokfelt, O. Edfors, and T. Maseng, "A survey on trellis termination alternatives for turbo codes," in *IEEE Vehicular Technology Conference*, Houston, Texas, USA, May 1999.

[10] W. J. Blackert, E. K. Hall, and S. G. Wilson, "Turbo code termination and interleaver conditions," *Electronics Letters*, vol. 31, pp. 2082–2084, November 1995.

[11] O. Joerssen and H. Meyr, "Terminating the trellis of turbo codes," *Electronics Letters*, vol. 30, pp. 1285–1286, August 1994.

[12] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations." TDA progress report 42-122, Jet propulsion Lab., Pasadena, CA, August 1995.

[13] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, pp. 284–286, March 1974.

[14] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," in *Globecom 1989*, pp. 471.1–471.7, Dallas, Texas, USA, November 1989.

[15] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, March 1996.

[16] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," in *IEEE International Conference on Communications*, Seattle, USA, 1995.

[17] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, pp. 409–428, March 1996.

[18] L. Perez, J. Seghers, and D. J. C. Jr., "A distance spectrum interpretation of turbo codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1698–1709, November 1996.

[19] H. Herzberg, "Multilevel turbo coding with short interleavers," *IEEE Journal on Selected Areas in Communications*, pp. 303–309, February 1998.

[20] J. Hokfelt and T. Maseng, "Methodical interleaver design for turbo codes," in *International Symposium on Turbo Codes & Related Topics*, Brest, France, September 1997.

[21] J. Hokfelt, O. Edfors, and T. Maseng, "A turbo code interleaver design criterion based on the performance of iterative decoding," *IEEE Communications Letters*, vol. 4, May 2000.

[22] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello, "A note on asymmetric turbo-codes," *Communications Letters*, vol. 3, pp. 2082–2084, March 1999.

[23] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proceedings Globecom '94*, pp. 1298–1303, Dec. 1994.

[24] M. C. Reed and S. S. Pietrobon, "Turbo-code termination schemes and a novel alternative for short frames," in *Proc. International Symposium on Personal, Indoor and Mobile Radio Communications 1996*, Taipei, Taiwan, October 1996.

[25] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-output decoding algorithms for continuous decoding of parallel concatenated convolutional codes," in *Proceedings of International Conference on Communications, ICC 1996*, pp. 112–117, 1999.