Modern Coding Schemes for Unequal Error Protection

by Neele von Deetzen
n.vondeetzen@jacobs-university.de

Ph.D. Dissertation

JACOBS
UNIVERSITY

Dissertation Committee:
Prof. Dr.-Ing. Werner Henkel, Jacobs University Bremen
Prof. Dr. Götz Pfander, Jacobs University Bremen
Prof. Dr.-Ing. Martin Bossert, Ulm University

November 19, 2008
School of Engineering and Science
Jacobs University Bremen

To ...

Acknowledgements

...

Abstract

In this thesis, we present modern and efficient channel coding techniques for the protection of user data with heterogeneous error sensitivities. Especially multimedia data being transmitted through communication networks often consist of unequally important parts, such as header information, essential payload, and additional data for increased quality. Protecting all data equally makes the transmission inefficient. A system providing unequal error protection (UEP) may be much more efficient and improve the perceptual quality at the receiver.

UEP transmitters and receivers should be designed such that transmission errors only lead to graceful degradation. For good channel conditions, the quality at the receiver is usually good. If the channel conditions degrade, UEP receivers should still be capable of exploiting at least the most important data in order to allow for graceful degradation instead of complete failure.

First, we introduce time-variant, rate-compatible pruned convolutional codes, which are a counterpart of the well-known punctured convolutional codes. Pruning may be an alternative to puncturing, especially if no feedback channel from the receiver to the transmitter is available. Variable-rate code families can be constructed from a given mother code by selectively pruning state transitions in the trellis of the code, thereby reducing the code rate. Theoretically, any code rate smaller than that of the mother code can be generated by applying suitable pruning patterns. We show that the free distance of a convolutional mother code can be specifically increased by pruning and that pruned convolutional codes are automatically rate compatible. Furthermore, we list tables of pruning patterns leading to good decoding results when being applied in convolutional and Turbo codes.

As a by-result, we present an analysis of hybridly concatenated codes and their decoder scheduling. Time-invariant pruning can be represented as the serial concatenation of a pruning code and the mother code. Using pruned codes for Turbo codes leads to a hybrid serial/parallel concatenation. The decoding success of such a hybrid concatenation strongly depends on the scheduling of the constituent decoders. We show a detailed analysis of the decoding process and propose an optimisation strategy for successful decoding with a minimum number of necessary iterations.

Another efficient coding scheme is multilevel coding which is a combination of channel coding and modulation, where both are jointly optimised. Multilevel codes are not

restricted to certain types of codes or modulation schemes and are, therefore, very flexible. The theory of multilevel codes provides a very natural and intuitive extension to unequal error protection. We present design rules for such systems based on the optimum design criteria for multilevel codes, i.e., the mutual information. By applying pruned Turbo codes as channel codes and higher order signal constellations in the modulation unit, we show examples of an image transmission where the UEP design yields good results whereas a standard design is not able to reconstruct the image at the receiver at all. The results and flexibility are further improved by applying special hierarchical modulation schemes instead of standard signal constellations.

The third part of the thesis contains design strategies for bandwidth-efficient low-density parity-check (LDPC) codes providing UEP. An intuitive way of designing UEP-LDPC codes is to design the variable node degree distribution in an irregular way. When dealing with higher order signal constellations, the code bits experience non-uniform disturbances which have to be taken into account during density evolution. We present a hierarchical optimisation algorithm using a detailed density evolution and give a list of optimised degree distributions.

The last part of this thesis is also connected to UEP-LDPC codes and deals with the UEP properties of different construction algorithms for the parity-check matrix of an LDPC code. We experience differences in the UEP behaviour of different construction algorithms despite them producing graphs with exactly the same degree distributions. We discuss several well-known algorithms and analyse properties of the parity-check matrix which are relevant for these different behaviours. In order to confirm our argument, we modify a construction algorithm without UEP capability such that the relevant properties are changed and a UEP-capable code is obtained.

Contents

Chapter 1

Introduction

In the last few decades, communication technologies have developed fast due to a drastically growing demand on the transmission of huge amounts of data. Especially, many digital communication techniques and standards have recently been developed to overcome the needs. One may think of Digital Subscriber Line (DSL), Digital Audio Broadcast (DAB), Digital Video Broadcast (DVB), the Global System for Mobile Communications (formerly Groupe Spécial Mobile, GSM), Code Division Multiple Access (CDMA), or the Universal Mobile Telecommunications System (UMTS). Compared to analog transmission schemes, digital communication facilitates compression, copying, reproduction, protection, and representation of data. However, the transmission over the channel itself is analog, and digital/analog and analog/digital converters are employed.

One of the most important steps in the development of communication theory was the pioneering work of C. Shannon [Sha48] where he proved theoretical limits of information theory, the well-known source and channel coding theorems. An important keyword is the *channel capacity* which defines the amount of data which can, theoretically, be transmitted reliably over a given channel. Details thereon will be given later on. Unfortunately, Shannon did not show how to reach these bounds in practice. Since then, researchers have been working hard on the development of capacity-achieving systems, and have approached capacity more and more closely. One of the research focuses within communication systems is channel coding which is employed for increasing the reliability of a lossy system by adding redundancy before transmission. At the receiver, the redundant data symbols are used to detect and correct errors.

There are many applications in communication environments that deliver data of different error sensitivities. Especially in multimedia there exist file formats where parts of the data are more important than others and, thus, errors due to additive noise or multipath propagation may have more or less severe effects. These different classes of importance should therefore be handled differently during transmission. Protecting all data equally is inefficient and wastes transmission capacities. On the contrary, important data may be protected better than less important data in order to maximise the perceptual quality. The aim is to achieve graceful degradation of the quality if errors occur.

There are other reasons for applying unequal error protection than the properties of the source data. For time-variant channel conditions one may be interested in a stable transmission quality by adapting the reliability of the system. Furthermore, data might be protected differently depending on the user terminal: A big screen TV certainly needs a higher resolution than the screen of a handheld or a mobile phone.

The unequal protection can be realised in many ways and at different places in a communication link, e.g. it could be included into adaptive modulation or into adaptive bit and power allocation when using multicarrier modulation. This work, however, deals with unequal error protection within channel coding.

1.1 Communication Systems

Digital communication systems can be summarised by the simplified block diagram in Fig. 1.1. Note that this is a digital model which assumes the existence of digital data only. Relations between the analog and the digital entities are included in the digital models. Further details on digital communications can be found in [Pro01]. A source produces digital data which are encoded, or compressed, in favour of a compact description with as little redundancy as possible. Ideally, the source encoded data contain no redundancy at all and are represented by as few data as possible. Practically, some encoders do not realise this. After source encoding, data may be encrypted for the sake of privacy and security. Systems with encryption provide unique keys to authenticated users which are used for communication. The source, the source encoder, and the encryption can be considered as an equivalent digital source model. In the context of this thesis, we assume that the equivalent source provides binary data where the occurrences of zeros and ones are equally probable.

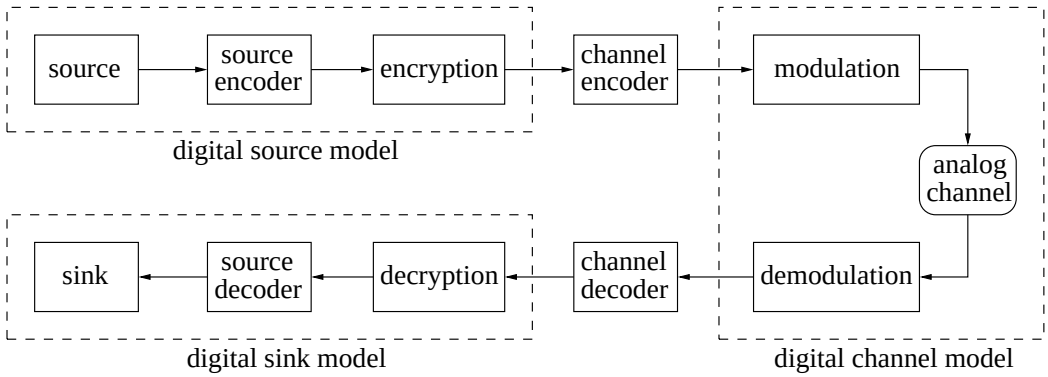


Figure 1.1: Block diagram of a digital communication system

The next step is channel encoding which, together with channel decoding, is the main focus of this thesis. The channel encoder obtains information words or sequences from the source and adds redundancy in order to construct codewords or code sequences. The encoding rule determines how much redundancy is added, i.e., the *code rate* and how the codewords are constructed from the information words. The set of all possible codewords is called the *code* and its properties determine the correction capability.

The codewords are mapped to signal points of a modulation alphabet in order to be ready for transmission. Large signal constellations increase the data rates and bandwidth efficiency but also lead to higher error sensitivities. In order to account for the properties of higher order signal constellations, coding schemes should be well-designed depending on the channel and the modulation scheme such that (more or less) reliable transmission is guaranteed.

During transmission, the symbols are affected by disturbances, which are summarised in the channel model. The data may experience different kinds of disturbances like additive and multiplicative noise, attenuation, delay, or multipath propagation. Depending on the kind and the origin, disturbances may occur as single events, in bursts, or as stationary noise.

At the receiver, the disturbed symbols are demodulated in order to retrieve the estimated codewords. Due to the disturbances, errors may occur in the received data which means that the estimated codewords may be different from the original ones. The aim of the channel decoder is to detect and correct errors such that it yields reliable estimates of the information words. Different strategies may be applied at the decoder which

have different correction capabilities depending on the kind of errors that occurred on the channel. The obtained estimates are decrypted and sent to the source decoder in order to obtain useful information for the sink.

As mentioned before, the main focus of this thesis is channel coding. However, properties of the source as well as of the modulation have to be taken into account, as well. It is important to be aware of the structure of the source data in order to find suitable rules of how to treat them. Furthermore, the modulation affects the error probabilities of distinct bits to different extents and has, thus, also to be taken into consideration. Based on these conditions, we develop coding schemes that are capable of providing UEP.

1.2 Channel Models and Properties

Before presenting the basic principles of channel coding, we discuss some channel models. Generally, a channel is defined by the conditional probability distribution $p_{Y|X}(y|x)$ which gives the probability of a channel output symbol $y \in \mathcal{A}_{out}$ given the channel input symbol $x \in \mathcal{A}_{in}$. Note that random variables are given by capital letters $(X,\ Y)$, whereas their realisations are denoted by lower-case symbols $(x,\ y)$. The channel input and output alphabets $\mathcal{A}_{in}$ and $\mathcal{A}_{out}$ can be discrete or continuous and do not have to be equal or of equal size. In the following, we define a few important channel models.

Definition 1.1 (Binary Symmetric Channel (BSC)) *Let the binary input and output alphabet be $\mathcal{A}_{in} = \{-a,\ +a\}$ and $\mathcal{A}_{out} = \{-b,\ +b\}$, respectively. Then, the conditional probability mass function of a BSC is given by*

$$p(Y = +b|X = -a) = p(Y = -b|X = +a) = p\ , \tag{1.1}$$

$$p(Y = +b|X = +a) = p(Y = -b|X = -a) = 1 - p\ . \tag{1.2}$$

Definition 1.2 (Binary Erasure Channel (BEC)) *Let the input and output alphabet be $\mathcal{A}_{in} = \{-a,\ +a\}$ and $\mathcal{A}_{out} = \{-b,\ \varepsilon,\ +b\}$, respectively. Then, the conditional*

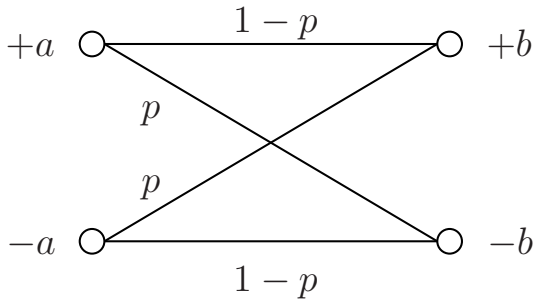


Figure 1.2: Model of the Binary Symmetric Channel

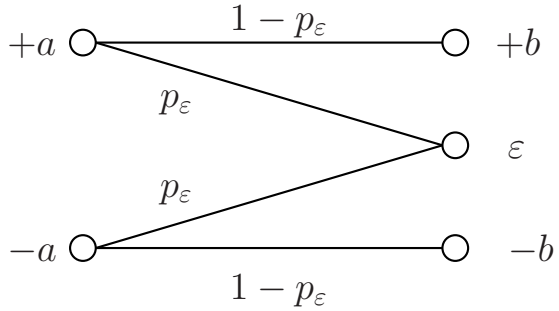


Figure 1.3: Model of the Binary Erasure Channel

probability mass function of a BEC is given by

$$p(Y = +b|X = -a) = p(Y = -b|X = +a) = 0\,, \tag{1.3}$$

$$p(Y = \varepsilon\,|X = +a) = p(Y = \varepsilon\,|X = -a) = p_\varepsilon\,, \tag{1.4}$$

$$p(Y = +b|X = +a) = p(Y = -b|X = -a) = 1 - p_\varepsilon\,. \tag{1.5}$$

Definition 1.3 (Additive White Gaussian Noise (AWGN) Channel) *Let the discrete input alphabet be $\mathcal{A}_{in}$. Furthermore assume that the channel input values x are superposed by the (continuous) additive, white, Gaussian noise values n.*

$$y = x + n \tag{1.6}$$

Thus, the output alphabet is the (continuous) set of real numbers $\mathcal{A}_{out} = \mathbb{R}$ and the

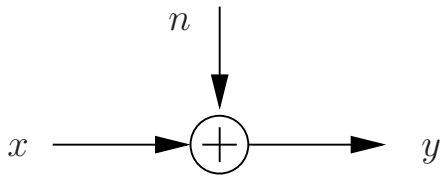


Figure 1.4: Model of the Additive White Gaussian Noise Channel

conditional probability distribution of an AWGN channel is given by

$$p_{Y|X}(y|x) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \cdot e^{-\frac{(y-x)^2}{2\sigma_n^2}} \ , \tag{1.7}$$

where σ_n^2 is the variance of the Gaussian random noise.

All results in this thesis are obtained assuming an AWGN channel. Although various aspects and theoretical results of channel coding have only been proved for the BEC, they have also been applied to AWGN channels with satisfying results.

Generally, when dealing with channels, we are interested in the amount of information about a transmitted symbol which is available in the corresponding received symbol. Therefore, we define the information content of a discrete symbol X_ν to be

$$I(X_\nu) = \log_2 \frac{1}{P(X_\nu)} = -\log_2 P(X_\nu) \tag{1.8}$$

Especially, we are interested in the average information content, which is called the *entropy*.

Definition 1.4 (Entropy of a random variable) *The entropy of a random variable X is defined by its average information content*

$$H(X) = -\sum_\nu P(X_\nu) \cdot \log_2 P(X_\nu) \ . \tag{1.9}$$

Let us define the joint entropy of two random variables X and Y as

$$H(X,Y) = -\sum_\nu \sum_\mu P(X_\nu, Y_\mu) \cdot \log_2 P(X_\nu, Y_\mu) \ . \tag{1.10}$$

Furthermore, let us define the equivocation and the irrelevance as

$$H(X|Y) = -\sum_\nu \sum_\mu P(X_\nu, Y_\mu) \cdot \log_2 P(X_\nu|Y_\mu) \quad \textit{and} \tag{1.11}$$

$$H(Y|X) = -\sum_\nu \sum_\mu P(X_\nu, Y_\mu) \cdot \log_2 P(Y_\mu|X_\nu) \ . \tag{1.12}$$

Finally, the mutual information of two random variables is given by

$$H(X;Y) = H(X) + H(Y) - H(X,Y) \; . \tag{1.13}$$

For later purposes, we also define the binary entropy.

Definition 1.5 (Binary entropy function) *We define the entropy of a binary source with input probabilities $P(0) = p$ and $P(1) = 1 - p$ as the binary entropy function*

$$h(p) = -p \log_2(p) - (1-p) \log_2(1-p) \; . \tag{1.14}$$

Imagine X and Y being the transmitted and the received symbol alphabet. If X is not fully contained in Y, the equivocation $H(X|Y)$ represents the 'lost' information. In contrast, the irrelevance $H(Y|X)$ represents information that is contained in Y but not in X, i.e., useless information. Furthermore, the mutual information $H(X;Y)$ represents the amount of information which is contained in both X and Y, i.e., the amount of information about the transmitted symbols that is still available at the receiver.

An important measure of a channel is its *channel capacity* C, which is the maximum of the mutual information over all possible distributions $P(X_\nu)$ of the channel input alphabet $\mathcal{A}_{in}$.

Definition 1.6 (Channel Capacity C) *The channel capacity of a discrete channel with input probabilities $P(X_\nu)$ and transition probabilities $P(Y_\mu|X_\nu)$ is given by*

$$C = \sup_{P(X)} \sum_\nu \sum_\mu P(Y_\mu|X_\nu) P(X_\nu) \cdot \log_2 \frac{P(Y_\mu|X_\nu)}{\sum_l P(Y_\mu|X_l) P(X_l)} \; . \tag{1.15}$$

Correspondingly, in the case of a continuous channel with input distribution $p_X(x)$ and conditional probability distribution $p_{Y|X}(y|x)$, the channel capacity is defined by

$$C = \sup_{p_X(x)} \int_Y \int_X p_{Y|X}(y|x) \log_2 \frac{p_{Y|X}(y|x)}{p_Y(y)} \; dX \; dY \; . \tag{1.16}$$

With the above definitions, the channel capacities of the BSC, the BEC, and the AWGN channel can be given by:

$$C_{BSC} = 1 - h(p) \; , \tag{1.17}$$

$$C_{BEC} = 1 - p_\varepsilon \,, \tag{1.18}$$

and

$$C_{AWGN} = \frac{1}{2}\log_2(1 + \frac{S}{N}) \,, \tag{1.19}$$

where $S/N = \sigma_x^2/\sigma_n^2$ is the signal-to-noise ratio (SNR) of the channel, with σ_x^2 and σ_n^2 being the transmit signal power and the noise variance. For derivations and proofs, please see [CT06].

In his channel coding theorem, Shannon proved that channel coding can achieve an arbitrarily small number of errors after decoding, given that the code rate is smaller than or equal to the channel capacity and that the code length goes to infinity. Furthermore, an arbitrarily small number of errors cannot be achieved if the code rate exceeds the channel capacity, regardless of the code length. For further details about information theory, the reader is referred to [CT06]. In the following section, we will briefly present principles of channel coding.

1.3 Channel Coding Principles

Channel coding in general adds redundancy to information data at the transmitter in order to be able to detect or even correct errors at the receiver. There are different strategies of doing so. In the following, we introduce binary block codes, convolutional codes, and concatenated codes.

1.3.1 Linear Block Codes

Let an information word be given by the length-K vector $\mathbf{u} = (u_0 \ \dots \ u_{K-1})$. Generally, the set of possible information words, or messages, is q^K, where $q \geq 2$ is the alphabet size. In this thesis, we only deal with binary codes, i.e., $q = 2$ and $u_i \in \{0, 1\}$.

A binary block code $\mathcal{C}$ of length N is defined by a set of length-N codewords $\mathbf{x} = (x_0 \ \dots x_{N-1})$, where $N \geq K$ and $x_i \in \{0, 1\}$. For encoding, the messages are uniquely assigned to the codewords such that the mapping is bijective and the cardinality of the code is $|\mathcal{C}| = q^K$. The difference in length between the messages and the codewords is the length of the redundancy $M = N - K$.

Definition 1.7 (Code rate R) *The code rate R of a binary block code $\mathcal{C}$ is defined by*

$$R = \frac{K}{N} = \frac{\log_2 |\mathcal{C}|}{N} = 1 - \frac{M}{N} \; . \tag{1.20}$$

It is bounded by $0 \leq R \leq 1$ and gives information about the amount of redundancy relative to the codeword length.

The mapping of the distinct messages to the codewords is determined by the channel encoder. For block codes, the encoding rule can usually be represented by the multiplication of the messages with a generator matrix $\mathbf{G}$ of size $[K \times N]$.

$$\mathbf{c} = \mathbf{u} \cdot \mathbf{G} \tag{1.21}$$

Depending on the structure of the generator matrix, the encoder may be systematic or non-systematic. K columns of a systematic generator matrix form an identity matrix such that the message bits are verbatim preserved within the codeword, although possibly permuted. A linear block code can also be described by its parity-check matrix $\mathbf{H}$ of size $[N - K \times N]$ which is defined by

$$\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0} \; . \tag{1.22}$$

The parity-check matrix is usually used for decoding but may be used for encoding, as well. It is very important, since it may give information about code properties and the decoding behaviour. The simplest form of decoding a linear block code is syndrome decoding. From (1.21), it follows that

$$\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0} \; . \tag{1.23}$$

Let us assume that a codeword $\mathbf{c}$ is corrupted by the channel and is available at the receiver as $\hat{\mathbf{c}} = \mathbf{c} + \mathbf{e}$, where $\mathbf{e}$ denotes the error vector. Therewith,

$$\hat{\mathbf{c}} \cdot \mathbf{H}^T \;\; = \;\; \underbrace{\mathbf{c} \cdot \mathbf{H}^T}_{=\mathbf{0}} + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T \; , \tag{1.24}$$

which is called the syndrome. An error will be detected if $\hat{\mathbf{c}} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T \neq \mathbf{0}$. Thus, any error vector which is nonzero and which is not a codeword itself will be detected. Different strategies may assign a syndrome to a valid codeword. We are

especially interested in iterative decoding procedures which is a less complex alternative to syndrome decoding in many cases. Iterative belief propagation decoding will be described later in this thesis.

One of the most important properties of a linear code is the minimum Hamming distance. Before its definition, we introduce the Hamming weight w_H and the Hamming distance d_H.

Definition 1.8 (Hamming weight w_H of a codeword) *The Hamming weight w_H of a codeword is the number of nonzero bits in it.*

$$w_H(\mathbf{c}) = ||\mathbf{c}||_0 = |supp(\mathbf{c})| \tag{1.25}$$

Definition 1.9 (Hamming distance d_H of two codewords c and c′) *The Hamming distance between two codewords $\mathbf{c}$ and $\mathbf{c}'$ is the number of bits by which they differ,*

$$d_H(\mathbf{c}, \mathbf{c}') = \sum_{i=0}^{N-1} (c_i + c_i') \bmod 2 = ||(\mathbf{c} + \mathbf{c}') \bmod 2||_0 = |supp((\mathbf{c} + \mathbf{c}') \bmod 2)| \ . \tag{1.26}$$

Definition 1.10 (Minimum Hamming distance d_{min}) *The minimum Hamming distance d_{min} of a binary linear block code is equal to the smallest number of bits by which any two codewords differ.*

$$d_{min} = \min_{\substack{\mathbf{c},\mathbf{c}' \\ \mathbf{c} \neq \mathbf{c}'}} d_H(\mathbf{c}, \mathbf{c}') \tag{1.27}$$

The minimum Hamming distance d_{min} determines the asymptotic performance of a code and gives a bound on how many erroneous bits in a codeword can be detected and corrected. Let t_d and t_c be the maximum number of detectable and correctable errors in a codeword, respectively. Then

$$t_d \geq d_{min} - 1 \quad \text{and} \quad t_c \geq \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \ . \tag{1.28}$$

When applying decoding methods that correct and detect errors at the same time,

$$t_c + t_d + 1 \leq d_{min} \tag{1.29}$$

must hold. Clearly, the number of correctable errors increases with growing d_{min}. Thus, one important aim in code design is the maximisation of the minimum distance d_{min} which dominates the asymptotic behaviour. A very useful performance estimation is given by the *union bound*, which provides an upper bound on the error probability of a certain decoded codeword.

Definition 1.11 (Union bound on the error probability of a codeword) *For an AWGN channel and antipodal transmission, the probability that a codeword is decoded to a wrong codeword can be lower-bounded by*

$$P_e(\mathbf{c}) \leq \frac{1}{2} \sum_{d=d_{min}}^{n} a_d \operatorname{erfc}\left(\sqrt{d \frac{E_s}{N_0}}\right) , \tag{1.30}$$

where a_d represents the number of codewords $\mathbf{c}'$ with Hamming distance $d_H(\mathbf{c}, \mathbf{c}') = d$. E_s/N_0 denotes the signal-to-noise ratio with respect to symbol energy E_s, where $N_0/2$ is the two-sided power spectral density of the random noise. The complementary error function is defined by

$$\operatorname{erfc}(x) = \frac{2}{\pi} \int_x^{\infty} e^{-t^2} dt . \tag{1.31}$$

The union bound is the sum over all possbile error events. Since the erfc-function is strictly decreasing, it is clear that for large signal-to-noise ratios, the first terms of the sum dominate the bound. Especially the minimum distance d_{min} which defines the very first term is an important parameter when estimating the asymptotic performance. An upper bound can be given by

$$P_e(\mathbf{c}) \leq \frac{1}{2} a_{d_{min}} \operatorname{erfc}\left(\sqrt{d_{min} \frac{E_s}{N_0}}\right) \tag{1.32}$$

where $a_{d_{min}}$ is the multiplicity of codewords with the minimum Hamming weight.

1.3.2 Convolutional Codes

Besides block codes, convolutional codes are another important type of codes. The encoder consists of a shift register and has, therefore, memory. The number of memory elements is denoted by N_{mem} and the constraint length is defined by $L_c = N_{mem} + 1$. The content of the memory elements is denoted as the state of the encoder. The

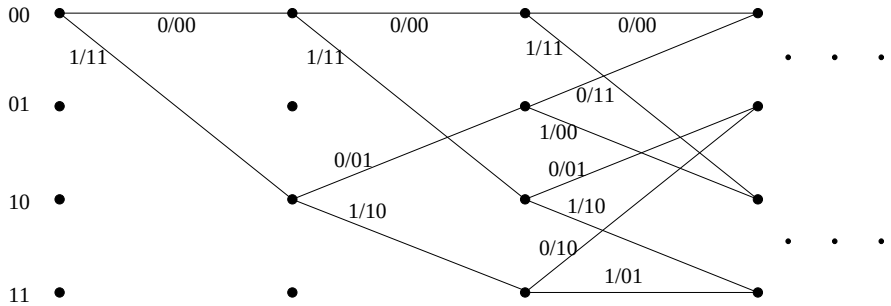


Figure 1.5: Trellis diagram of a rate-1/2 NSC code with generator matrix $\mathbf{G}(D) = (1 + D \;\; 1 + D + D^2)$

most commonly used convolutional codes are non-recursive non-systematic convolutional (NSC) codes and recursive systematic convolutional (RSC) codes. As the name suggests, the basic operation is a convolution, and the input and output are given as (infinite) sequences. The generator and the input and output sequences are usually given by polynomials in D, where D represents a delay operator, corresponding to z^{-1} of the Z-transform. The encoding rule of a convolutional code is therewith given by

$$\mathbf{u}(D) = \ldots \mathbf{u}_{i-1}D^{-1} + \mathbf{u}_i D^0 + \mathbf{u}_{i+1}D^1 \ldots , \tag{1.33}$$

$$\mathbf{c}(D) = \ldots \mathbf{c}_{i-1}D^{-1} + \mathbf{c}_i D^0 + \mathbf{c}_{i+1}D^1 \ldots , \tag{1.34}$$

$$\mathbf{c}(D) = \mathbf{u}(D) \cdot \mathbf{G}(D) \; . \tag{1.35}$$

Every recursive systematic encoder can be transformed into a non-recursive non-systematic encoder representing the same code. The generator matrix $\mathbf{G}(D)$ of an NSC is a $[K \times N]$ matrix with polynomial entries. For RSCs, the matrix usually contains fractions of polynomials. For a more detailed derivation, the reader is referred to [JZ99]. The input/output relations and state transitions may be illustrated by a Trellis diagram which shows possible state transitions on a temporal axis. Figure 1.5 shows a trellis diagram of an NSC code with generator matrix $\mathbf{G}(D) = (1 + D \;\; 1 + D + D^2)$. The bold dots denote the encoder states along the horizontal temporal axis. The labels at the state transitions represent the corresponding input and output bits. The Trellis diagram is fully developed after N_{mem} steps.

Decoding algorithms of convolutional codes are usually based on the trellis diagram. Well-known decoding algorithms are the BCJR algorithm by Bahl, Cocke, Jelinek, and Raviv [BCJR74] and its approximations or the Viterbi algorithm [Vit67]. The BCJR algorithm and its applications are symbol-based maximum a-posteriori (MAP) decoders

which are based on the following computation.

$$\begin{aligned}\hat{\mathbf{c}} &= \arg\max_{\mathbf{a}\in\mathcal{C}} P(\mathbf{a}|\mathbf{y}) = \arg\max_{\mathbf{a}\in\mathcal{C}} P(\mathbf{y}|\mathbf{a}) \cdot \frac{P(\mathbf{a})}{P(\mathbf{y})} \\ &= \arg\max_{\mathbf{a}\in\mathcal{C}} P(\mathbf{y}|\mathbf{a}) \cdot P(\mathbf{a}) \ , \end{aligned} \tag{1.36}$$

whereas the Viterbi algorithm is a maximum-likelihood (ML) sequence estimator which generally assumes equally likely channel input symbols. Therefore, $P(\mathbf{a}) = P(\mathbf{a}') \,\forall\, \mathbf{a}, \mathbf{a}' \in \mathcal{C}$, and the optimisation target is

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{a}\in\mathcal{C}} P(\mathbf{y}|\mathbf{a}) \ . \tag{1.37}$$

By including $P(\mathbf{a})$, however, also the Viterbi algorithm can be made a MAP decoder. There are several other decoding methos, e.g. list decoding or sequential decoding [JZ99]. We will not derive these algorithms and refer to the given literature.

Like block codes, there exists an important distance measure also for convolutional codes. The *free distance* d_{free} of a convolutional code is the minimum number of bits in which any two paths in the Trellis differ.

1.3.3 Concatenation

Concatenated codes consist of several codes which are combined in serial or parallel. In a serial concatenation, the codewords of the outer code are encoded by the inner code, whose codewords are transmitted over the channel. In a parallel concatenation, all encoders encode the same information word, and the codewords are multiplexed for transmission.

1.3.3.1 Parallel Concatenation

The most popular concatenated encoders consist of two encoders, although this is not a necessary condition. The well-known *Turbo codes* [BGT93a] typically consist of two parallel convolutional encoders separated by an *interleaver*. The interleaver π permutes the information sequence of the first encoder $\mathbf{u}_1$ for the second encoder in order to avoid statistical dependencies between the disturbances on two received sequences. Figures 1.6 and 1.7 show the encoder and the decoder of a Turbo code. Usually, RSC codes are employed such that the systematic bits are only transmitted once.

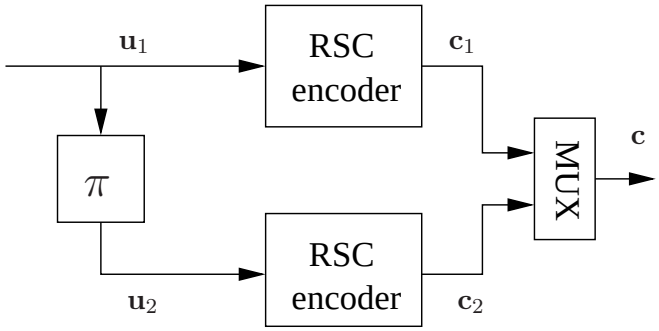


Figure 1.6: Block diagram of a Turbo encoder

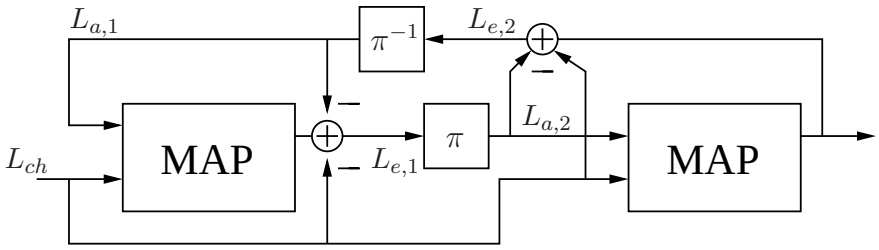


Figure 1.7: Block diagram of an iterative Turbo decoder

For soft decoding, it is useful to define the log-likelihood ratio (LLR) $L(x)$ which is a measure of the reliability of the received information. The sign of the LLR gives information about the estimated symbol, whereas its absolute value is a measure of the reliability with which the symbol is estimated.

Definition 1.12 (Log-likelihood ratio (LLR) $L(x)$) *The log-likelihood ratio of a received value y is given by*

$$L(x) = \log \frac{p(x=+1|y)}{p(x=-1|y)} = \underbrace{\log \frac{p(y|x=+1)}{p(y|x=-1)}}_{L(y|x)=L_{ch}} + \underbrace{\log \frac{p(x=+1)}{p(x=-1)}}_{L_a(x)} . \tag{1.38}$$

$$\tag{1.39}$$

The first part of the sum, L_{ch}, is called *intrinsic information*, or intrinsic L-values. It contains information from the channel. The second part, $L_a(x)$, is the *a-priori information* which solely depends on the channel input distribution. Based on L_{ch} and $L_a(x)$, the decoder computes estimates on the transmitted sequences, i.e., the a-posteriori information $L_{app}(x)$.

The decoding process of a Turbo code is performed in an alternating fashion, where the two component decoders iterate and exchange information. The information which is exchanged between the decoders is called extrinsic information. It is computed by subtracting a-priori information and intrinsic channel output values from the computed a-posteriori information. The extrinsic information of a decoder is interleaved or deinterleaved and passed to the other decoder as a-priori information.

$$L_{e,1/2} = L_{app,1/2} - L_{a,1/2} - L_{ch} \tag{1.40}$$

$$L_{a,2} = \pi(L_{e,1}) \text{ and} \tag{1.41}$$

$$L_{a,1} = \pi^{-1}(L_{e,2}) \tag{1.42}$$

In a parallel concatenation, the information passed between the two decoders corresponds to the non-interleaved and interleaved information sequences $\mathbf{u}_1$ and $\mathbf{u}_2$. The final decision of such a parallel decoder is based on the final a-posteriori estimates of both component decoders.

$$\hat{\mathbf{c}} = \frac{1}{2}(1 - \text{sign}(L_{app,1} + L_{app,2})) \tag{1.43}$$

1.3.3.2 Serial Concatenation

As mentioned before, a serial concatenation consists of two or more codes, where a code encodes the interleaved codeword of the previous code. Figures 1.8 and 1.9 show the block diagrams of the encoder and the decoder of such a concatenation.

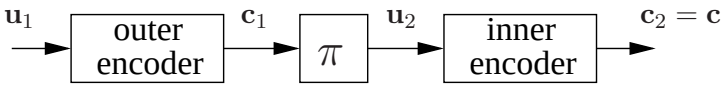


Figure 1.8: Block diagram of a serial encoder

The decoding process of the serial concatenation is similar to that of the parallel concatenation. The difference is that only the inner decoder obtains intrinsic channel information. Furthermore, the inner decoder estimates its uncoded sequence and passes its extrinsic information to the outer decoder as a-priori information about its code sequence. The outer decoder in turn estimates its code sequence and passes its extrinsic part to the inner decoder as a-priori information about its uncoded sequence. In order to avoid statistical dependencies, the a-priori information is subtracted from the a-posteriori information in both cases.

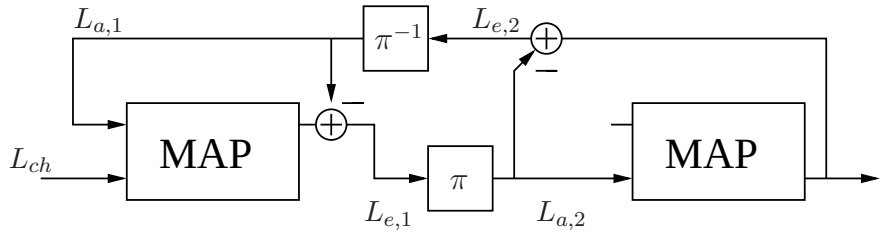


Figure 1.9: Block diagram of an iterative decoder of a serial concatenation

Let $L^u_{a,i}$ and $L^c_{a,i}$ denote the a-priori information of a decoder corresponding to the uncoded sequence and to the code sequence, respectively, and let this notation be valid for the extrinsic and the a-posteriori information, as well. Therewith, the update operations of a serial decoding process are

$$L^u_{e,2} = L^u_{app,2} - L^u_{a,2} - L_{ch} \tag{1.44}$$

$$L^c_{a,1} = \pi(L^u_{e,2}) \tag{1.45}$$

for the inner decoder and

$$L^c_{e,1} = L^c_{app,1} - L^c_{a,1} \tag{1.46}$$

$$L^u_{a,2} = \pi(L^c_{e,1}) \tag{1.47}$$

for the outer decoder.

The final decision of a serial decoder is based on the final a-posteriori estimate of the outer decoder.

$$\hat{\mathbf{c}} = \frac{1}{2}(1 - \text{sign}(L^u_{app,1})) \tag{1.48}$$

1.3.3.3 EXIT Charts

A very useful tool for the analysis of iterative decoding schemes is the extrinsic information transfer chart, known as *EXIT chart*. The EXIT chart plots the mutual information based on the extrinsic values versus the a-priori mutual information of a decoder. It was first developed by Stephan ten Brink in 2001 [tB01a]. The starting point for the derivation of the EXIT chart is the observation that the a-priori input L_a of one of the decoders can be modelled by an independent Gaussian random variable n_a with variance σ_a and zero mean together with the transmitted systematic bits x_s.

$$L_a = \frac{\sigma_a}{2} x_s + n_a \tag{1.49}$$

The conditional probability density function of the L-values L_a is

$$p_a(L_a|X = x_s) = \frac{1}{\sqrt{2\pi}\sigma_a} e^{-\frac{(L_a-(\sigma_a^2/2)x_s)^2}{2\sigma_a^2}} \tag{1.50}$$

This leads to the mutual information $I_a = I(X; L)$.

$$I_a = \frac{1}{2} \cdot \sum_{x_s=-1,1} \int_{-\infty}^{+\infty} p_a(\xi|X = x_s) \cdot \log_2 \frac{2 \cdot p_a(\xi|X = x_s)}{p_a(\xi|X = -1) + p_a(\xi|X = +1)} \, d\xi \tag{1.51}$$

With Eq. (1.50), Eq. (1.51) becomes

$$I_a(\sigma_a) = 1 - \int_{-\infty}^{+\infty} p_a(\xi|X = x_s) \cdot \log_2(1 + e^{-\xi}) \, d\xi \tag{1.52}$$

$$= 1 - E\left\{\log_2(1 + e^{-L_a})\right\} \approx 1 - \frac{1}{N} \sum_{i=1}^{N} \log_2(1 + e^{-x_{s,i} \cdot L_{a,i}}) \, . \tag{1.53}$$

The mutual information for the extrinsic output L_e can be formulated in the same way, i.e.,

$$I_e(\sigma_a) = 1 - E\left\{\log_2(1 + e^{-L_e})\right\} \approx 1 - \frac{1}{N} \sum_{i=1}^{N} \log_2(1 + e^{-x_{s,i} \cdot L_{e,i}}) \, . \tag{1.54}$$

It should be noted that the mutual information is always bounded by $0 \leq I \leq 1$. The EXIT chart plots the transfer function between the a-priori information and the extrinsic information of a component decoder.

$$I(L_{e,1/2}; X) = T(I(L_{a,1/2}; X)) \tag{1.55}$$

When using EXIT charts for iteratively decoded concatenated codes, the transfer curves for both decoders are plotted into one single transfer chart. Since the extrinsic information of one decoder becomes the a-priori information of the other, the transfer curve of the second decoder is flipped, thereby representing the inverse transfer function.

$$I(L_{a,2/1}; X) = T^{-1}(I(L_{e,2/1}; X)) \tag{1.56}$$

This is possible because the original transfer function is monotonically increasing and therefore, its inverse exists. One can additionally add the trajectories between the

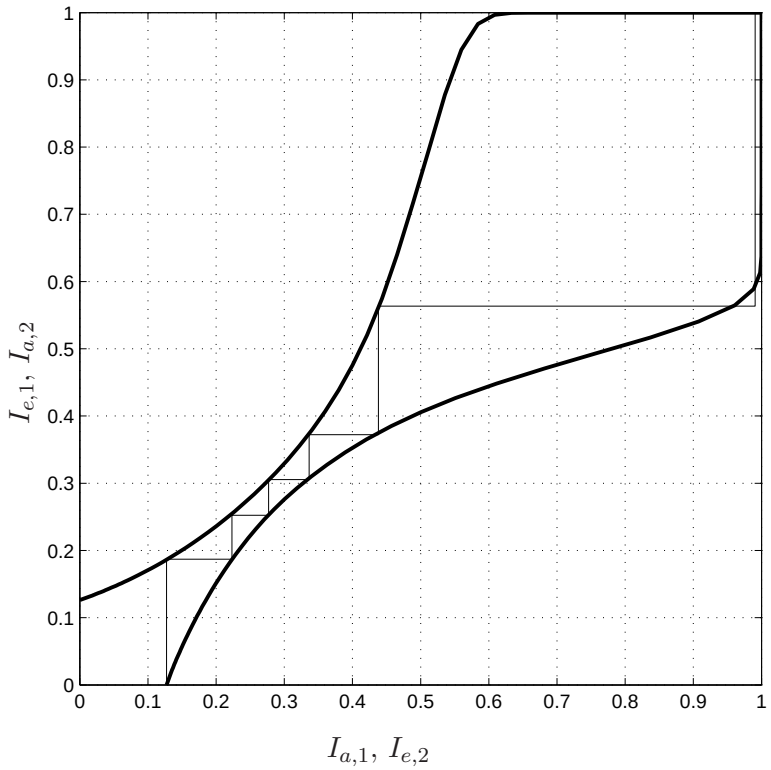


Figure 1.10: Example of an EXIT chart

two transfer functions which represent the information transferred between the two decoders. Figure 1.10 shows a typical EXIT chart of an iterative Turbo decoding process with two identical encoders.

Firstly, the EXIT chart shows if the decoding process converges successfully. If the tunnel between the transfer curves is open, the trajectories reach the point (1,1) which means full knowledge about the transmitted information at the receiver. In case the transfer curves intersect, the trajectories 'get stuck' and successful decoding is not possible. The signal-to-noise ratio where the tunnel opens is often called the threshold of the code. Secondly, one can read the number of necessary decoding iterations from the EXIT chart.

The quality of the decoding results depends on the signal-to-noise ratio. For lower signal-to-noise ratios, the two curves intersect and do not permit error-free transmission. With increasing signal-to-noise ratios, the two curves widen and let the intersection point move towards the point $(1, 1)$. For even larger signal-to-noise ratios, the already

open curves further widen which leads to faster convergence.
Hence, the segments of bit-error rate curves are related to the properties of an EXIT chart. These can be divided into three parts. The first one, the pinch-off region, is where the bit-error rate hardly decreases with increasing signal-to-noise ratio. The second part is called the waterfall region and describes the region, where the bit error rate dramatically decreases. The third region represents the "error floor" of the bit error rate.
These three regions can also be determined by the EXIT chart. The first region represents those cases, where the transfer curves intersect below the point $(1, 1)$. The waterfall region describes the case, when the transfer curves open and allow the decoding to converge and the error floor represents the case, when the transfer curves further widen. However, not much can be deduced for this error floor region.
Evaluating the EXIT chart in addition to the union bound provides a rough estimate for the bit-error rate curve, since the EXIT chart provides information about the waterfall region, whereas the union bound approximates the asymptotic behaviour for high signal-to-noise ratios and low bit-error rates.
Consider the independent random variable n_a as the noise of an independent channel. In the case of this independent channel being a binary erasure channel, it has been proved for serial concatenations in [AKtB04] that the area under the first (upper) EXIT curve represents the capacity, whereas area under the second (lower) represents the code rate. In order to reach the capacity, the area between the two curves has to be minimised without intersection of the curves.

1.4 Thesis Outline

This thesis is divided into four parts. Chapter 2 deals with the well-known convolutional codes and with Turbo codes. We construct UEP-capable codes by varying the code rate in a flexible way. The presented alternative to puncturing is time-variant pruning. After presenting the originally proposed technique, we improve it by increasing the flexibility and the ability to control essential properties. We investigate pruned codes in terms of flexibility, error-rate performance, and distance measures. Furthermore, we discuss the optimisation of the so-called pruning patterns in order to obtain free distances as large as possible. In Appendix A, we list several convolutional and Turbo code families based on convolutional mother codes of different constraint lengths.

In Appendix B, we discuss a by-result of pruned convolutional codes. Pruned convolutional codes can be described as the serial concatenation of a pruning code and a mother code. If these are employed in Turbo codes, the overall code is a mixed parallel and serial concatenation, called hybrid concatenation in the following. Especially the decoding of hybrid Turbo codes offers challenging problems. In this work, we discuss the decoder scheduling, i.e., the order in which the component decoders are activated. We provide a detailed description of the decoder and the messages which are exchanged. Furthermore, we present multiple EXIT charts which are useful for optimising the scheduling in terms of convergence speed.

In Chapter 3, UEP multilevel codes are presented. Multilevel codes jointly address coding and modulation. The information theoretic design rules are independent of the choice of channel codes, i.e., one may apply block codes, convolutional codes, or concatenated codes. Multilevel codes are very convenient for designing and controlling UEP properties. We introduce the analysis of the mutual information within multilevel codes and discuss the fundamental concept towards a UEP-capable scheme. We present several examples to illustrate the approach and to show the results of the proposed design. We provide an option to improve the perceptual quality of multimedia data by omitting the least important bits in favour of the most important data, maintaining good quality while preserving the data rate.

Chapters 4 and 5 deal with the design and the properties of low-density parity-check (LDPC) codes for unequal error protection. Chapter 4 presents fundamentals of LDPC codes. Afterwards, a design algorithm is developed which optimises the degree distribution of a UEP-LDPC code especially for higher order constellations. In the context

of multimedia data, transmission systems have to face increasing amounts of data to be transmitted. Therefore, it is important that communications schemes handle resources like power or bandwidth efficiently. Applying higher order signal constellations is a natural way to deal with this. Since the bit positions in a symbol are affected differently by channel noise, it is important to take the equivalent channel model into consideration when optimising the degree distribution. We present examples with 8-PSK and 64-QAM modulation which show satisfying results in terms of bit-error rates.

Chapter 5 also deals with UEP-LDPC codes. Assuming given (optimised) degree distributions, it analyses different algorithms for constructing parity-check matrices. We briefly introduce several existing algorithms and investigate their behaviour in terms of UEP capabilities. We point out the differences and discuss relevant properties of the parity-check matrix. In order to verify the correctness of our statements, we modify an existing algorithm which traditionally does not yield UEP-capable codes. Thereby, we obtain an algorithm which produces highly UEP-capable codes, outperforming the existing UEP-capable construction algorithms.

Chapter 6 summarises the results of this thesis and gives an outlook to open topics.

Chapter 2

Time-Variant Rate-Compatible Pruned Convolutional Codes

This chapter presents an approach for unequal error protection convolutional and Turbo codes, called *pruning*, which is a flexible alternative to the well-known puncturing. Both approaches yield variable-rate codes by changing the number of information bits or codebits. Puncturing was introduced by J. Hagenauer in 1988 [Hag88] and has been investigated much. The so-called rate-compatible punctured convolutional (RCPC) codes are especially useful for unequal error protection together with automatic repeat request (ARQ). We will present a similar approach which is based on varying the number of encoder input bits.

After explaining the technical details, we present some simulation results to confirm the theoretical basis. Furthermore, we will discuss distance properties of the approach and propose optimisation strategies. We give a comparison of pruning and puncturing and present lists of good pruning patterns in the appendix.

2.1 General Description

In this section, we describe the original pruning method for achieving unequal error protection with convolutional codes which can later be applied in Turbo codes. A straightforward approach of varying the performance of a convolutional code is puncturing, i.e., excluding a certain amount of code bits from transmission and, thus, increasing

the code rate $R = K/N$, where K and N are the numbers of information bits and code bits. Another method called pruning was presented in [WC02] and [HvD06a] for convolutional codes and Turbo codes, respectively. Pruning is based on modifying the number of input bits to the encoder K, i.e., the numerator of the code rate instead of the denominator. The aim of pruning is to find a code family consisting of codes with code rates $R_i = K_i/N$. In order to keep the decoding complexity low, it is desired that the individual codes should have certain properties such that decoding can be performed by similar decoders. This requirement leads to the construction of several subcodes from a given mother code. The subcodes of rates $R_{s,i} = K_{s,i}/N$, $1 \leq k_{s,i} < K_m$, are constructed by multiplying the polynomial generator matrix of the mother code of rate $R_m = k_m/n$, denoted by $\boldsymbol{G}_m(D)$, by another polynomial generator matrix $\boldsymbol{G}_p(D)$ called pruning matrix.

$$\boldsymbol{G}_s(D) = \boldsymbol{G}_p(D) \cdot \boldsymbol{G}_m(D) \tag{2.1}$$

Choosing different pruning matrices and especially varying the dimensions, leads to different subcodes $\boldsymbol{G}_{s,i}(D)$ of the mother code. The dimensions of the pruning matrix are defined as $[K_p \times K_m]$ in order to guarantee proper matrix multiplication, leading to a subcode generator matrix of dimensions $[K_p \times N]$. Thus, varying the number of rows of the pruning matrix varies the code rate of the subcode. Obviously, for a mother code of rate $R_m = K_m/N$, one can construct subcodes with $K_m - 1$ different code rates $1/N, \ldots, (K_m - 1)/N$. Given that the number of memory elements of the sub-encoder m_s is equal to (or smaller than) the number of memory elements in the mother encoder, the influence of building a subcode from a mother code can directly be depicted in the trellis diagram of the convolutional codes. Figure 2.1 shows a trellis section of a mother code (left) and a subcode (right) given by the following generator matrices.

$$\begin{aligned} \boldsymbol{G}_s(D) &= \boldsymbol{G}_p(D) \cdot \boldsymbol{G}_m(D) \\ &= (1 \;\; D) \cdot \begin{pmatrix} 1 & 1+D & 1+D \\ 1+D & D & 1+D \end{pmatrix} \\ &= \begin{pmatrix} 1+D+D^2 & 1+D+D^2 & 1+D^2 \end{pmatrix} \end{aligned} \tag{2.2}$$

The labels to the left of the states denote the input and output of the transitions leaving the states.

When comparing the two trellis sections, one can see that the set of paths in the right trellis is a subset of the set of paths in the mother trellis. This means that one does not have to store a separate decoder for all codes of a family but the decoder

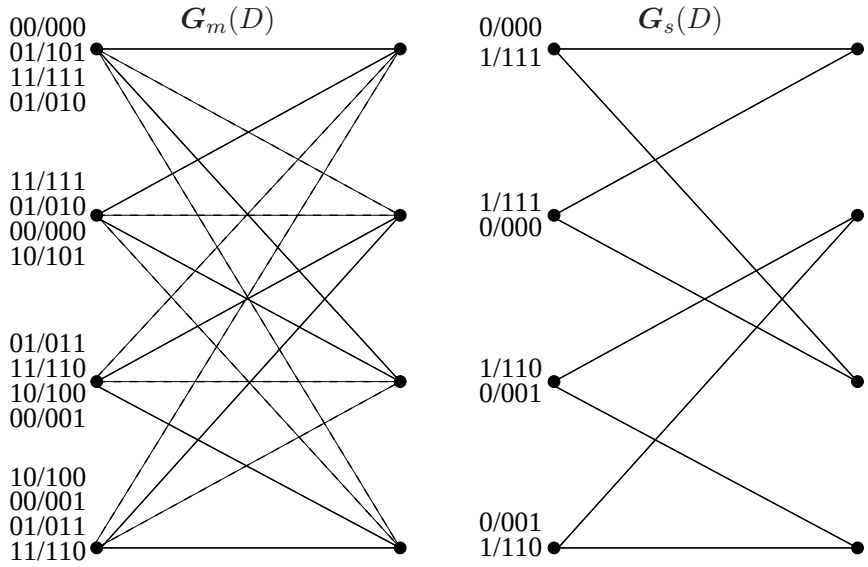


Figure 2.1: Trellis sections of mother code (left) and subcode (right)

structure basically stays the same. When switching between different decoders, only the transition probabilities between the states have to be adapted. With this procedure, we are able to construct a code family consisting of codes with K_m different code rates.

Since the multiplication of polynomial generator matrices corresponds to a serial concatenation of codes, a Turbo code which employs pruned convolutional codes may be represented as a hybrid Turbo code with a serial concatenation in each parallel branch. As a side result of this thesis, Appendix an analysis of hybridly concatenated codes and their decoder scheduling. Time-invariant pruning can be represented as the concatenation of a pruning code and the mother code. Using pruned codes for Turbo codes leads to a hybrid concatenation. The decoding success depends on the scheduling of the constituent decoders. We show an analysis of the decoding process and propose an optimisation strategy for successful decoding with a minimum number of necessary iterations.

In the following, we improve the above concept of pruning for convolutional and Turbo codes. A drawback of pruning is the low number of available code rates in a code family, especially for small mother codes, i.e., mother codes with small K_m. Particularly, the number of available code rates is dependent on the dimensionality of the mother code. For example, a mother code of rate $R_m = 2/3$ only allows for one subcode rate, namely $R_s = 1/3$. For applications like multimedia, where unequal error protection is desired,

usually more levels of protection are required. This calls for an approach, where the number of available code rates is independent of the code itself.

This leads to the concept of time-variant pruning. The corresponding measure in the field of puncturing is employing puncturing matrices defining a pattern of successive puncturing rules. The same can be done for pruning, such that for each time instant, a subcode is chosen according to some pattern and the overall code rate is flexible and controllable. Let us define the pruning period L_p to be the length of such a pattern which is repeatedly applied. Let the code applied at time instant l be denoted by $\mathbf{G}_l(D) \in \{\boldsymbol{G}_m(D), \boldsymbol{G}_{s,i}(D)\}$, $1 \leq i \leq K_m - 1$, $0 \leq l \leq L_p - 1$ and K_l be the number of encoder input bits corresponding to $\mathbf{G}_l(D)$. The overall code rate of the scheme is then defined by

$$R = \frac{1}{L_p \cdot N} \sum_{l=0}^{L_p-1} K_l \ , \tag{2.3}$$

which is bounded by $1/N \leq R \leq K_m/n$. Naturally, the resolution of possible code rates within the interval grows with increasing L_p.

Imagine a rate $R = 2/3$ mother code. A very easy case of such an above proposed pruning matrix can be defined by choosing one component of the pruning matrix to be 0 and the other one to 1:

$$\boldsymbol{G}_p(D) = \left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix} , \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\} \ . \tag{2.4}$$

This would select only one of the inputs as active and the other one as inactive in order to construct a subcode. Switching between the mother code and the subcode then means switching between code rates $R = 2/3$ and $R = 1/3$. This procedure can also be described as simply 'switching off' one of the inputs of the mother encoder according to the pruning pattern. Switching off one input is equivalent to feeding the corresponding input with zeros. Consequently, one does not even have to switch between the two encoders but only has to feed the mother encoder with zeros where the pruning pattern specifies to use the subcode. The effective code rate can be given as

$$R = \frac{L_p \cdot K - N_0}{L_p \cdot N} \ , \tag{2.5}$$

where N_0 denotes the number of digits fixed to 0. Thereby, we have found a very simple way to adapt the overall code rate of the scheme by just fixing some encoder input digits.

At the receiver, the pruning pattern is known such that the reliability of the fixed zeros can be set to infinity (or equivalently, the probability of a 0 can be set to 1) and may help decoding the other bits reliably.

However, one possible problem of the above proposed scheme might arise, when the number of fixed zeros is not negligible compared to the overall number of uncoded data. Especially for Turbo codes, the information bits are assumed to be equally distributed since the Turbo decoder suffers from statistical dependencies otherwise. If many zeros are fixed in the uncoded sequence, the probabilities of occurrence of a 0 and a 1 are not equal any more.

A possible solution for this problem is to not only insert zeros at the specified positions but also ones. This can be done in a predefined manner such that the receiver knows where zeros and ones are fixed. A very straightforward manner would be to choose the fixed digits to be 0 and 1 in an alternating fashion. This would directly preserve the equal distribution of zeros and ones.

In the following, we will give a small example in order to illustrate the proposed schemes. We will both show the insertion of fixed zeros only and of fixed zeros and ones. Assume an application that requires a code rate of $R = 0.4$ and the available codes are a mother code of rate $R_m = 2/3$ and a subcode of rate $R_s = 1/3$. A possible pruning pattern would specify the following succession of employed codes leading to the desired code rate: $\boldsymbol{G}_m(D)\ \boldsymbol{G}_s(D)\ \boldsymbol{G}_s(D)\ \boldsymbol{G}_s(D)\ \boldsymbol{G}_s(D)$. This sequence would encode 6 input bits and output 15 code bits, thus $R = 0.4$. When employing the mother code only and fixing the specified digits to zeros, the corresponding input sequence would be

$$\mathbf{u} = \begin{pmatrix} u_1 & 0 & 0 & 0 & 0 \\ u_2 & u_3 & u_4 & u_5 & u_6 \end{pmatrix} . \tag{2.6}$$

When inserting not only zeros but zeros and ones in an alternating fashion, a possible encoder input sequence is given by

$$\mathbf{u} = \begin{pmatrix} u_1 & 0 & 1 & 0 & 1 \\ u_2 & u_3 & u_4 & u_5 & u_6 \end{pmatrix} . \tag{2.7}$$

These small examples show how a code rate other than that of the mother code or one of the subcodes can easily be achieved by the proposed method.

When performing a computer search for a suitable pruning scheme, it is usually not sufficient to study pruning patterns alone. Additionally, it has to be ensured that

at interval boundaries, the states at joint trellis segments are the same as already required in rate-compatible punctured convolutional codes [Hag88]. With the improved approach shown above, this problem does automatically not arise any more since the decoder is operating on one and the same trellis, namely the mother trellis, only varying certain a-priori probabilities. Thus, trellis structures do not change at transitions between different protection intervals at all.

2.2 Simulation Results

In this section, we show some simulation results which confirm the theory in the previous section. In Fig. 2.2, performance curves are shown for a non-recursive, non-

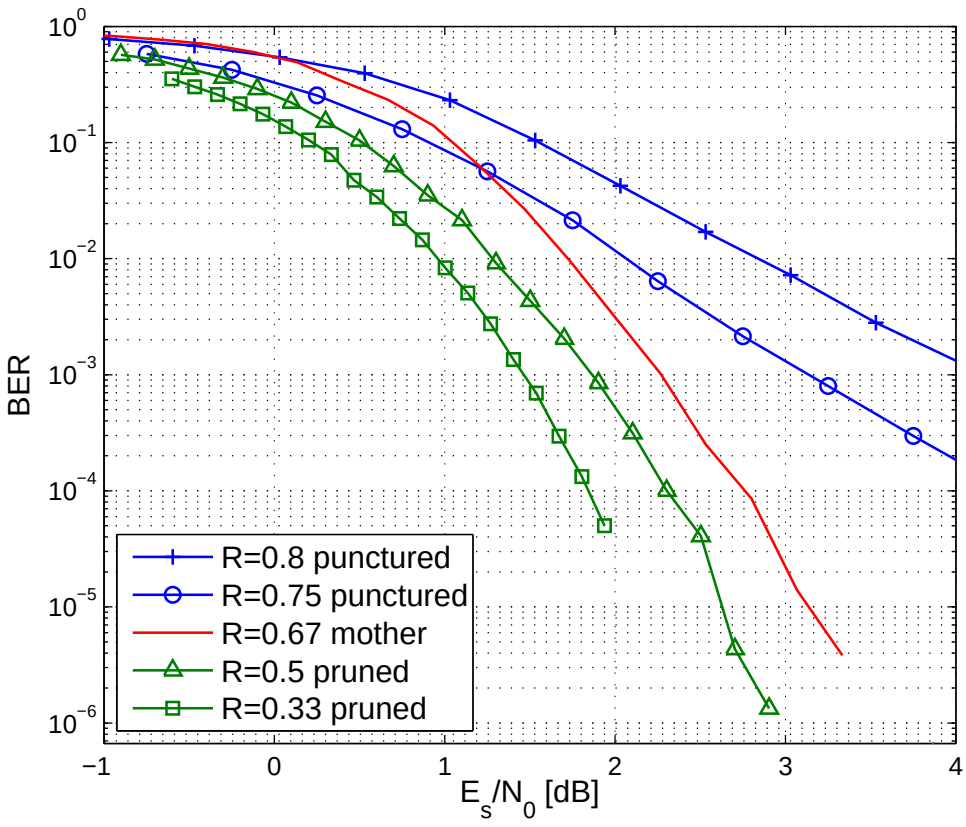


Figure 2.2: Set of bit-error-rate curves of pruned and punctured NSC codes

systematic convolutional (NSC) mother code of rate $R_m = 2/3$ and punctured as well

as pruned subcodes of rates $R_{s,i} = \{0.33\ ,\ 0.5\ ,\ 0.75\ ,\ 0.8\}$. Figure 2.3 shows performance curves of Turbo codes employing recursive, systematic convolutional (RSC) codes. The mother code rate is $R_m = 0.5$ and other code rates were obtained by puncturing ($R_{s,i} = \{0.6\ ,\ 0.7\}$) and pruning ($R_{s,i} = \{0.25\ ,\ 0.38\}$). For the Turbo code, we used equal convolutional codes as component codes in the encoder. More flexibility and probably faster decoding convergence is possible with different component codes. However, different codes require more memory for the implementation.

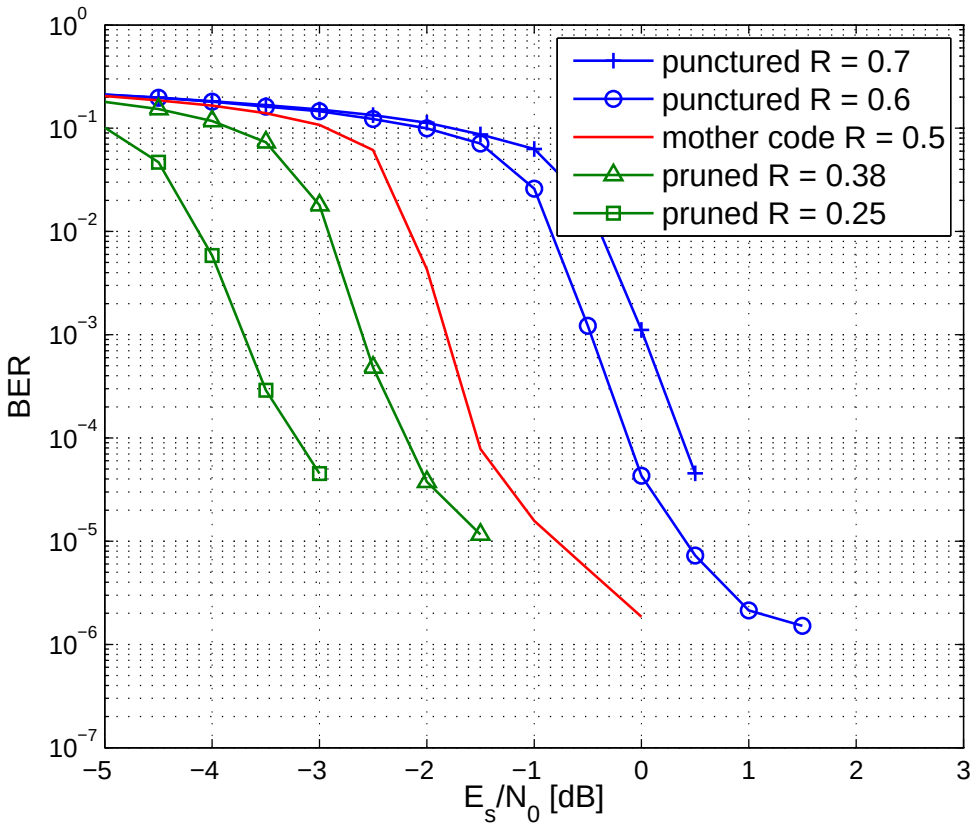


Figure 2.3: Set of bit-error-rate curves of pruned and punctured Turbo codes built from RSC codes

2.3 Linearity and Distance Properties

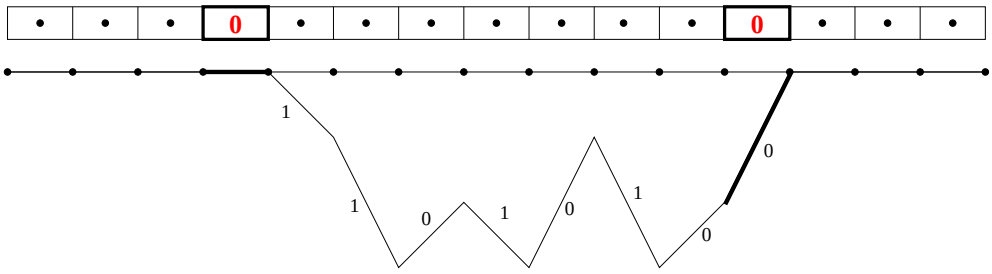
Regarding the free distance of a pruned subcode, one can generally say that it is in either case greater than or equal to the free distance of the mother code. As stated above, both codes can be illustrated by the same trellis. Fixing certain probabilities of a zero to be infinity means pruning those paths corresponding to a one at the input. Either, if a segment of the minimum-weight path is pruned, the free distance of the code is increased, or if it is not pruned, the free distance stays the same.

Note that the fixed positions of a pruned codeword do not necessarily have to be zeros. Fixing the values to one generally has the same effect in that it builds a subcode. However, analysing the distance properties is easier with fixed zeros due to the following reason. Fixing certain encoder input positions in an information sequence to zero will build a subset of all possible code sequences by only allowing a subset of all possible state transitions in the trellis. The subscode will be linear (proof necessary?) and it will definitely contain the all-zero codeword. This means that the evaluation of the distance spectrum can be done by analysis of the weight spectrum. However, if a bit is fixed to a one, the subcode does not contain the all-zero sequence and is, thus, not linear. Therefore, the distance spectrum of this subset cannot be easily determined by the weight spectrum. However, the effect of fixed ones is still comparable to fixed zeros concerning the distance spectrum. Only the analysis is not as easy. For this reason, we stick to fixed zeros in the following.

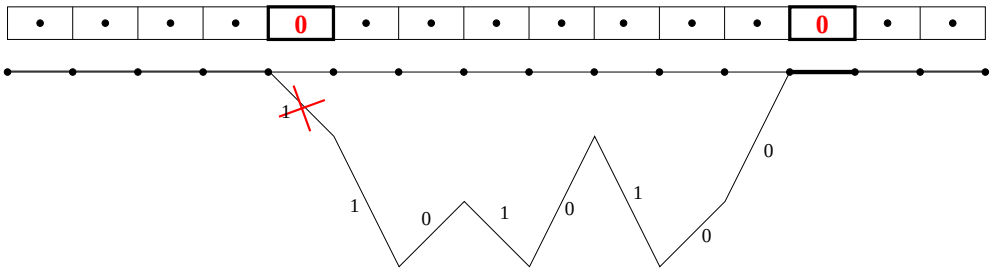
2.4 Optimisation of the Pruning Pattern

As mentioned before, the free distance of a pruned code is always greater than or equal to the free distance of its mother code. Clearly, the aim should be to delete the minimum-weight path from the trellis such that the free distance is determined by the next-higher weight path. Regarding the pruning pattern, this means that the distance between two constraints, i.e., fixed values should at least be equal to the length of the minimum-weight path, such that at least one constraint is always involved in the minimum-weight path. In the case of only one constraint, i.e., only one fixed bit, the probability of deleting the path is only $P(u_i = 0)$, which is 0.5 in most cases. Imposing more constraints on the minimum-weight path makes its deletion more probable.

The optimum would be if any cyclic permutation of the pruning pattern eliminates the minimum-weight path from the trellis. We will show some examples in the following.



(a) Pruning pattern does not erase minimum-weight path



(b) Pruning pattern erases minimum-weight path

Figure 2.4: Illustration of a pruning pattern which does not erase the minimum-weight path in any case

Fig. 2.4 shows an arbitrary non-zero trellis path of length 8 representing the minimum-weight path of a code. The labels next to the state transitions represent the corresponding input bits. The pruning pattern is indicated by the boxes above the path, where a zero means a fixed zero in the input sequence and a dot means any bit from the original input sequence. In this case, the pruning pattern fixes one bit out of eight bits, such that there is only one constrained transition in the whole path at any time. A single constraint can not guarantee the elimination of a path, its probability is given above as $P(u_i = 0)$.

When increasing the number of constrained transitions in the path, the probability of an elimination grows, of course. Knowing the input sequence corresponding to the minimum-weight path, one can optimise the pruning pattern such that it guarantees the erasure of the path (with as few constraints as possible).

Figure 2.5 shows a pruning pattern with two constraints out of seven bits which guar-

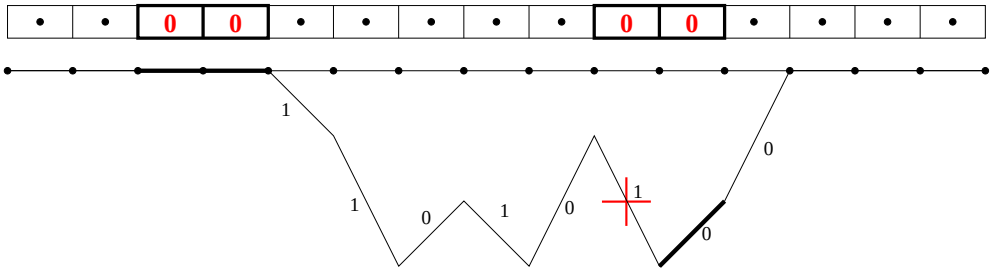
antees the erasure of the minimum-weight path. Shown are some cyclic permutations of the pruning pattern. It is easy to see that this pruning pattern will erase the path in any case, regardless of the cyclic permutation offset. Therewith, the minimum-weight path is erased and the next-higher weight path which is not erased by the pruning pattern defines the free distance of the code.

2.5 Puncturing versus Pruning

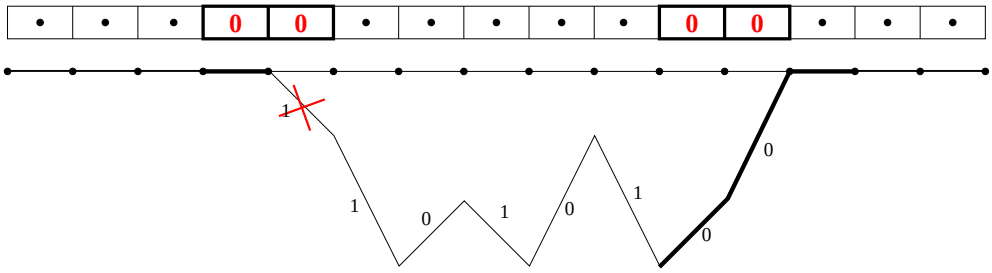
This section contains a comparison of punctured and pruned convolutional and Turbo codes. Puncturing is a well-known technique which is widely used. Its advantage is clearly its flexibility. When it is used with automatic repeat request (ARQ), it allows for minimum data traffic. If a punctured codeword has been received but was not able to be decoded successfully, the receiver may request further information. Instead of sending a completely new codeword, the transmitter only sends additional, previously punctured bits until the codeword can be decoded successfully. Another point is that, with puncturing, high rate codes with low decoding complexity can easily be constructed.

Nevertheless, pruning also has advantages, especially if there is no feedback channel available from the receiver to the transmitter and the advantages of ARQ cannot be used with puncturing. Regarding the free distance, it is easy to see that it is diminished by puncturing. On the contrary, the free distance may be increased, or even specifically controlled by pruning. This is especially interesting for higher signal-to-noise ratios, and for the error floor of Turbo codes. See Fig. 2.2 for example. One can see that the slope of the punctured codes is not as steep as of the mother code and the pruned codes. This is probably due to a reduced free distance.
Furthermore, the exact knowledge of certain bits may help decoding other bits during iterative decoding processes of Turbo codes, since they serve as extrinsic information to more unreliable bits. This effect is important for the waterfall region of the performance curve of a Turbo code.

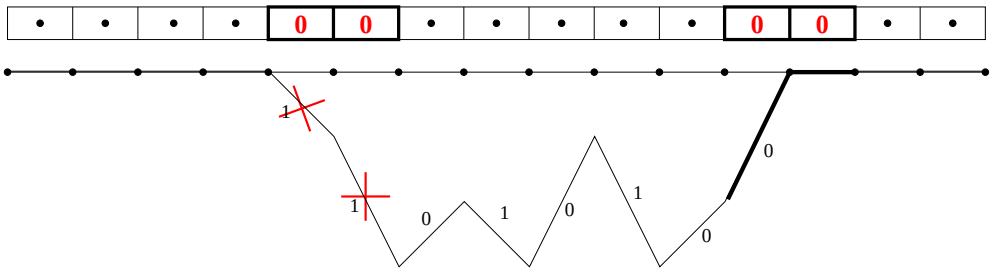
Furthermore, pruned convolutional codes are automatically rate compatible, as already stated in section 2.1. This means that pruning patterns can be applied independently from each other in any desired order. Furthermore, source data do not have to be ordered with decreasing importance, as required by RCPC codes.



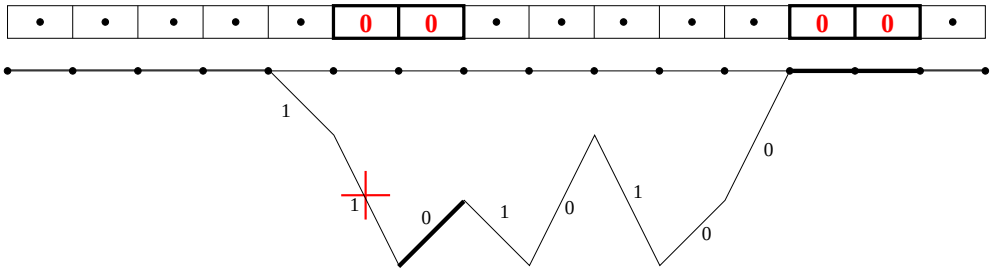
(a) Pruning pattern erases minimum-weight path by its 6th segment



(b) Pruning pattern erases minimum-weight path by its first segment



(c) Pruning pattern erases minimum-weight path by its first and second segment



(d) Pruning pattern erases minimum-weight path by its second segment

Figure 2.5: Illustration of a pruning pattern which erases the minimum-weight path in any case

2.6 List of Good Codes

We ran an exhaustive computer search in order to find mother codes together with different pruning patterns which behave well in iterative decoding. We used EXIT charts [tB01a] for evaluation of the convergence behaviour. One assessment criterion was amongst others the convergence threshold, which is the lowest SNR where error-free decoding is theoretically possible, i.e., where the tunnel between the EXIT curves opens and the mutual information between the decoded and the transmitted sequence is one (or very near to one). Furthermore, we report the area between the EXIT curves, since it is a measure of how close the waterfall region is to the Shannon limit and how steep it is [AKtB04]. Although this has only been proved for the binary symmetric channel, it has been observed for the AWGN channel as well. We also determined the approximate distance δ of the convergence point from the Shannon limit in dB. The minimum distance of the mother convolutional codes and their pruned subcodes was determined by evaluating low-weight input sequences.

Tables with good codes can be found in Tables A.1-A.3 in Appendix A. The tables show three convolutional mother codes with constraint lengths $L_c = 3$, $L_c = 4$, and $L_c = 5$ which showed reasonably early convergence. The convolutional mother code rate is $R_{CC} = 1/2$ in all cases, such that the Turbo code rate is $R_{TC} = 1/3$. The pruning pattern search was performed for pruning periods up to $L_p = 6$.

The code tables show that the higher the degree of pruning (and the lower the code rate), the larger is the minimum distance. This is natural, since with a large number of constraints, it is more likely that the minimum distance path is erased.

Chapter 3

Multilevel Codes and Hierarchical Signal Constellations

When dealing with the transport of multimedia data, their structural properties are important for an efficient and successful transmission. Images and, in general, multimedia data often have heterogeneous error sensitivities, especially when progressive source encoding is applied. These file formats contain parts which are more important than others and, thus, detection errors due to additive noise, multipath propagation, or other disturbances on the channel may have more (or less) severe effects. The knowledge of such structural details can be utilised by protecting these different classes of importance differently during transmission in order to make it more efficient and increase the perceptible quality.

Such a heterogeneous treatment may be obtained in many ways and at different instances in a communication scheme, e.g., applying hierarchical or adaptive modulation, or adaptive bit and power allocation when using multicarrier modulation. This work, however, deals with unequal error protection (UEP) within coded modulation, especially multilevel codes (MLC).

Coded modulation is a well-known technique which jointly optimises the coding scheme and the modulation scheme [Ung82a, IH77, Ung87a, Ung87b]. The signal constellation is successively subdivided into smaller subsets, where each partitioning level is assigned a label. These labels are protected by separate channel codes with appropriate protection capabilities. The codes have to be designed carefully depending on the

modulation scheme and its partitioning or labelling strategy. According to [WFH99], the optimal way of designing the codes (in terms of mutual information) is to match the code rates to the channel capacities of the partitioning steps, assuming perfect codes. This means that, for a given signal-to-noise ratio and given modulation scheme and partitioning, the optimal code rates are well defined. However, there are also other design approaches with similar results, like bit-interleaved coded modulation [CTB98] or low-density parity-check codes optimised for a certain modulation scheme [SSN04]. The corresponding channel codes in a multilevel coding scheme can be block codes, convolutional codes, or concatenated codes.

Previous work on unequal error protection multilevel codes has been published in [Wei93a], [GZY03], [KP01], and [LR05]. They focus on the design of special modulation schemes for achieving unequal error protection, especially with non-uniformly spaced signal constellations where symbols are grouped and the Euclidean distances within and between those groups differ. In [Wei93a], a time-division multiplexing scheme is proposed, switching between different conventional multilevel coding schemes. In [GZY03], non-uniform constellations together with a wavelet transform are applied in order to perform UEP image transmission. A non-uniform partitioning scheme leading to unequal error protection is presented in [KP01], and in [LR05], multiple block coded modulation is used together with nonregular partitioning. However, none of these publications focuses on the properties of the channel codes.

In this chapter, we present a way to modify the original multilevel coding approach [WFH99] in order to obtain and control unequal error protection by defining general design rules for the code dependent on a given signal constellation. We do not restrict the method to particular codes, but develop general rules which are applicable for any kind of codes. The chapter is structured as follows. The system model of a multilevel coding scheme is given in Section 3.1. Modifications for obtaining unequal error protection and their performances are given in Section 3.2 for standard signal constellations. Section 3.3 contains a discussion about the flexibility and possible improvements of the proposed scheme. In Section 3.4, hierarchical signal constellations are proposed to circumvent certain problems. Section 3.5 finally verifies the construction rules by an image transmission application.

3.1 System Model

A multilevel code consists of an M_m-ary modulation scheme and a coding unit. The signal constellation is successively partitioned into subsets until the subsets only contain a single signal point. The partitions are labelled at each partitioning level by symbols of an appropriate alphabet. A common approach for this partitioning strategy is Ungerböck's set partitioning, which maximises the minimum Euclidean distance between any two symbols of a subset [Ung82a], [UC76].

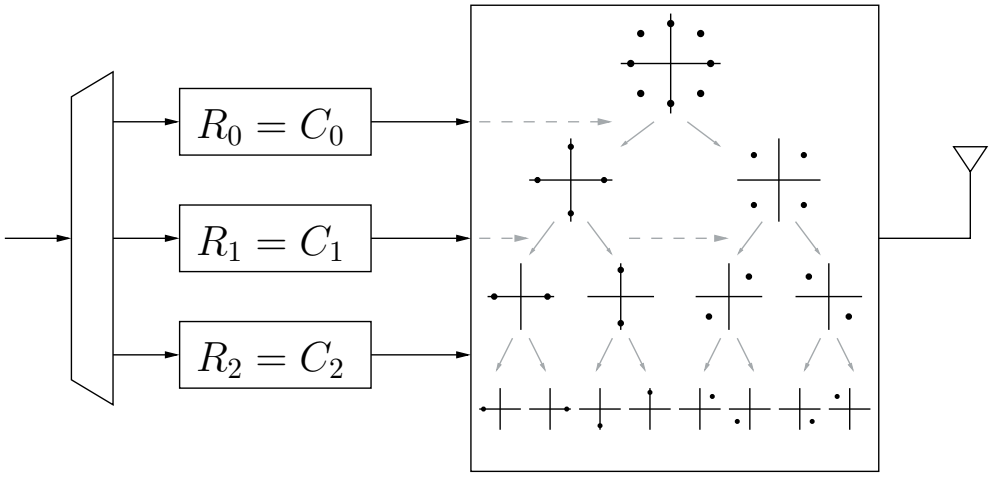


Figure 3.1: Transmitter structure of an 8-PSK MLC scheme

The labels of each partitioning level are components of codewords of individual codes at each level. Different design strategies have been proposed for these codes. For a long time, the balanced distances rule was believed to be best suited, where $\max\left\{\min_{i=0,\ldots,l-1}\left\{d_i^2\delta_i\right\}\right\}$, and thus $d_i^2\delta_i = \text{const}$ holds, where d_i^2 is the minimum squared Euclidean distance of the corresponding sub-constellation and δ_i is the minimum Hamming distance of the code at level i. In [WFH99], the authors proved the capacity design rule to be optimum in terms of mutual information. According to this rule,

$$R_i = C_i \tag{3.1}$$

for perfect codes, which means that the code rate at each level should be equal to the capacity of the partitioning. This is connected to Shannon's theorem which states that

1. a vanishing error probability is possible for $R < C$ for block lengths going to

infinity, and

2. the error probability will never vanish for $R > C$, regardless of the block length.

As a note on this, transmitting with a code rate $R > C$ will make error-free transmission impossible, whereas a code rate $R < C$ will just reduce efficiency but maintain the possibility of error-free transmission. In the context of finite block length codes, performance is improved the farther the rate is from capacity. Figure 3.1 shows the structure of such a multilevel coding transmitter with 8-PSK modulation and a 3-level set partitioning.

Figure 3.2 shows the capacity curves for a set partitioning of an 8-PSK scheme. It contains curves for 8-PSK, QPSK, and BPSK, since the set partitioning of the 8-PSK scheme leads to these kinds of subsets. The capacities of the individual partitioning levels follow from the chain rule of mutual information, [WFH99], and are given by

$$\begin{aligned} C_i &= I(Y; X^i | X^0 \cdots X^{i-1}) \\ &= \mathrm{E}_{x^0 \ldots x^{i-1}} \left\{ C(\mathcal{A}(x^0 \cdots x^{i-1})) \right\} - \mathrm{E}_{x^0 \ldots x^i} \left\{ C(\mathcal{A}(x^0 \cdots x^i)) \right\} , \end{aligned} \tag{3.2}$$

where $C(\mathcal{A}(x^0 \cdots x^{i-1}))$ represents the capacity of a signal subset $\mathcal{A}(x^0 \cdots x^{i-1})$. The vector $(x^0 \cdots x^{l-1})$ represents the bits addressing a symbol and X^i is the random variable corresponding to the ith bit. Correspondingly, Y^i represents the random variable of the ith bit based on the received symbol. As an example, the capacity of the first partitioning level of an 8-PSK scheme would be $C_0 = C^{\text{8-PSK}} - C^{\text{QPSK}}$.

A low-complex decoding method is called multistage decoding (MSD), where the levels are decoded one after another, taking into account decisions of previously decoded levels. Since the lower levels' performance is affected by the upper levels due to error propagation, the upper levels are chosen for important data and the lower ones for less important data.

The dashed lines in Fig. 3.4 show the error rates of the three levels in an 8-PSK multilevel code with MSD versus the SNR. Note that the SNR is given by E_s/N_0 in dB in order to compare the levels in a way that shows UEP properties. The codes are Turbo codes of length $N = 1000$ and generators $\mathbf{G}(D) = (1 \quad 1 + D + D^3 \quad 1 + D + D^2 + D^3)/(1 + D^2 + D^3)$ for both component codes. They are designed according to the capacity design rule in [WFH99] for an operating point of $E_s/N_0 = 6$ dB. We employ the same mother code for all levels, and different code rates are obtained by puncturing

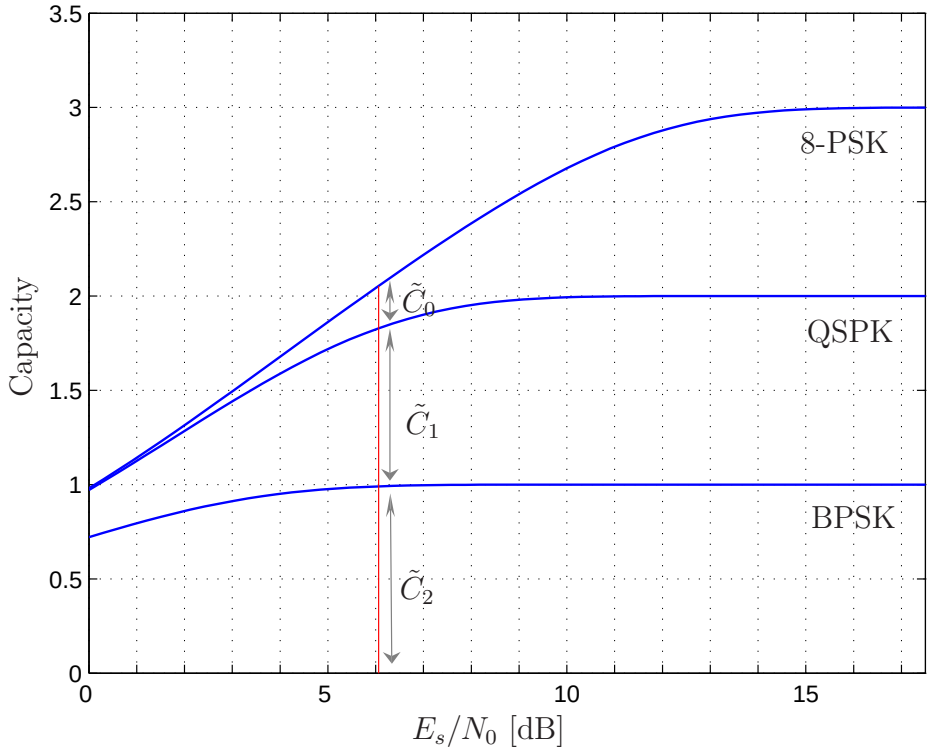


Figure 3.2: Capacity curves of the partitioning levels in an 8-PSK scheme with Ungerböck's set partitioning

and pruning [HvD06b], [HvDH+07b].

3.2 Modification for UEP

As shown in Fig. 3.4, the levels do not differ significantly in performance. This section describes how to modify the scheme in order to protect the bits in a symbol differently.

This work focuses on designing the coding unit rather than the modulation unit of the MLC scheme. According to the capacity design rule given in (3.1), the choice of the code rates is crucial to the performance of the system. Our approach is to vary the code rates in order to, on the one hand, improve the performance for the important data and, on the other hand, accept some performance degradation for less important

data. In spite of the lower Euclidean distance, we assign the most important data to the first partitioning level since the other levels are affected by error propagation in case of a wrong decision in the first level in MSD.

The general idea is to allow for a lower code rate for the important data and increase the code rate of the less important data, such that the overall rate remains constant. This allows for a fair comparison to the original non-UEP scheme. The variation of the code rates is achieved by choosing different operating points (w.r.t. signal-to-noise ratio) for the levels.

Consider again Fig. 3.2. For $E_s/N_0 = 6$ dB, we have approximately the following capacities at the levels:

$$\begin{aligned} C_0 &= 0.23\,, \\ C_1 &= 0.84\,, \\ C_2 &= 0.98\,. \end{aligned} \tag{3.3}$$

Thus, the optimal system would have the code rates $R_0 = 0.23$, $R_0 = 0.84$, and $R_0 = 0.98$ and all levels would have their waterfall region near the overall operating point. Assume now, the first level shall be better protected. In order to accomplish this, one may choose an individual operating point for this level at a lower SNR. Considering the capacity curves in Fig. 3.2, this means reducing this level's capacity. Sticking to the capacity design rule, this reduces the level's optimal code rate.

In return, the code rates of the other levels have to be balanced in order to keep the overall code rate constant. How the compensation is divided onto the remaining levels has to be traded off depending on the application. Figure 3.3 shows the capacity curves again, now with different individual operating points.

We choose $\tilde{C}_0 = 0.09$ which means that the reduction has to be balanced by the other levels. A possible choice is

$$\begin{aligned} \tilde{C}_0 &= R_0 = 0.09\,, \\ \tilde{C}_1 &= R_1 = 0.94\,, \\ \tilde{C}_2 &= R_2 = 1\,. \end{aligned} \tag{3.4}$$

The new operating points are now located at those SNRs where the new capacities $\tilde{C}_i$ are equal to the true capacities C_i. Thus, the individual operating points for the new capacities are approximately at 4.4 dB, 7.6 dB, and 7 dB.

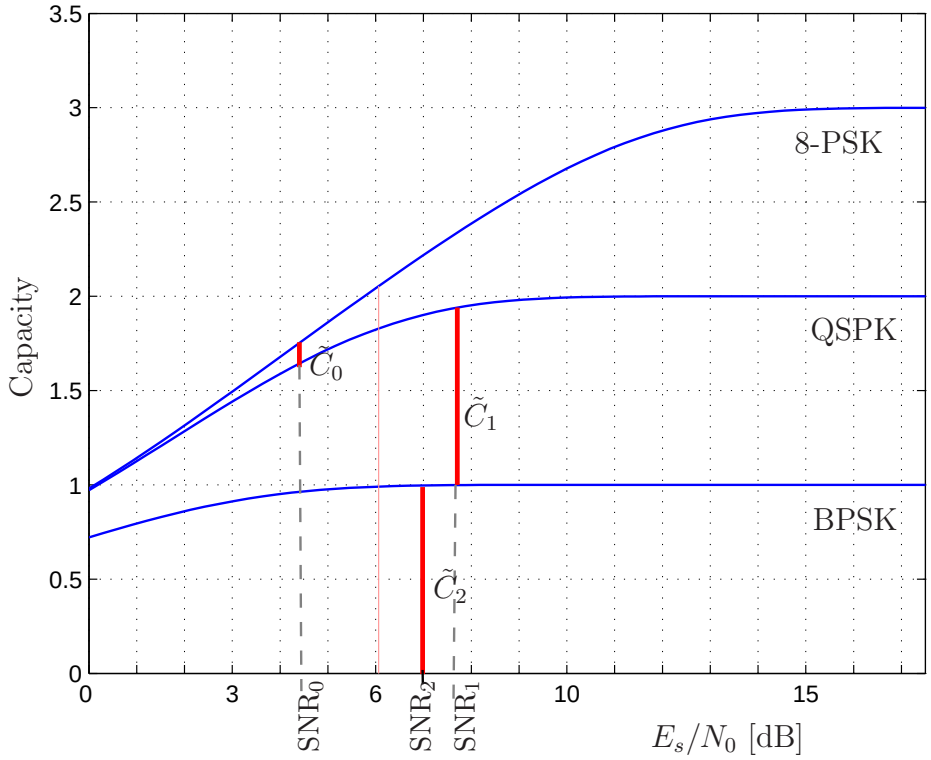


Figure 3.3: Individual operation points

The bit-error rates of the new system are shown in Fig. 3.4. The code parameters are the same as in Section 3.1 and the puncturing rates are designed for an operating point of $E_s/N_0 = 6$ dB. One can see that the new waterfall regions are near the designed operating points, as desired. Changing the code rate of a level, of course, has a direct impact on the waterfall region. By designing the code rates to be equal to the true capacities at a certain operating point (SNR), all levels (should) have good performance at (and above) the operating point, and a high error rate for lower SNR. By modifying the individual code rates, each level will show this behaviour at that particular SNR which yields a (true) capacity equal to the code rate on that level. Note that we assume optimal codes in this consideration. The actual location of the waterfall region depends on how close the code performances are to the Shannon limit.

Furthermore, the general previous statement only holds for maximum-likelihood decoding. As shown in Fig. 3.3, the individual operating point of the third level may be lower

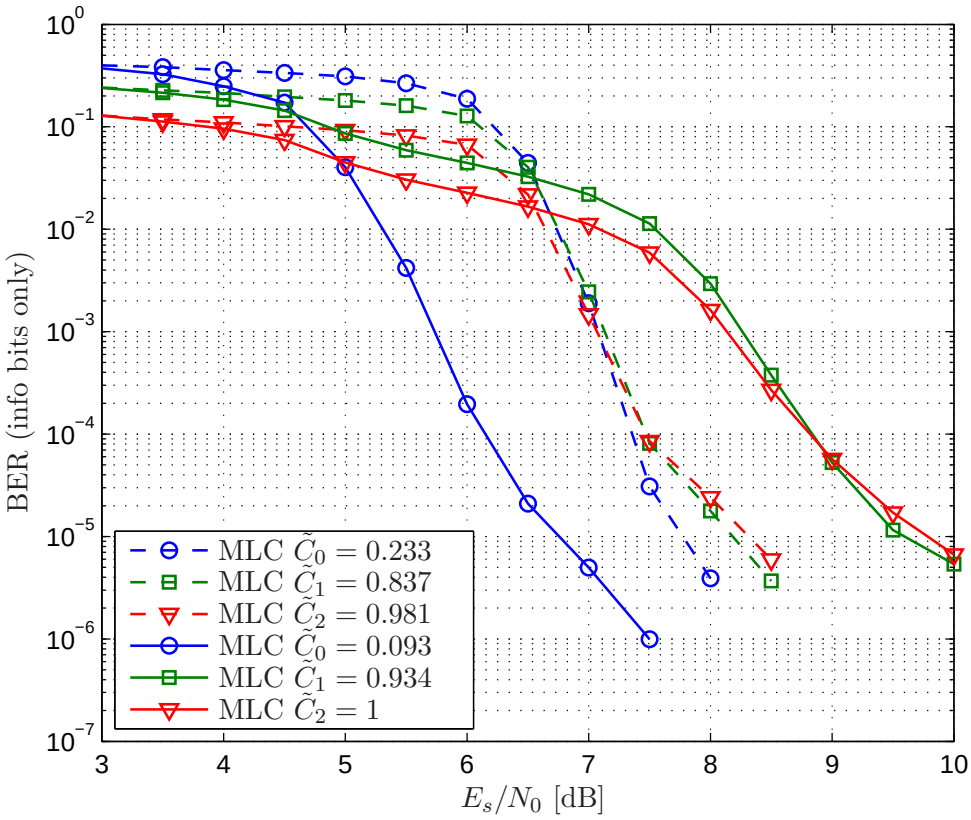


Figure 3.4: Performance of an original MLC scheme and a UEP-MLC scheme with individual operating points

than that of the second level. In the case of maximum-likelihood decoding, this would lead to waterfall regions close to these operating points. If MSD is used, the decoding process of a level is usually affected by error propagation and cannot outperform the higher levels. Thus, the resulting real operating point of level 2 would be the same as of level 1 in Fig. 3.3.

3.3 Flexibility

According to the above description, the waterfall regions of a desired UEP-MLC scheme can theoretically directly be chosen by defining appropriate code rates, as long as reasonably good channel codes are used which are near to the Shannon limit. The only

constraint is that the overall rate should be kept constant in order to be able to compare different codes or maintain the spectral efficiency. Generally, this method leads to a UEP system which is very easy to design and control.

However, taking the capacity curves with set partitioning into consideration, the choice of code rates can be very limited when the overall rate should be maintained. There is only a small SNR range where it is possible to trade off the code rates. Consider Fig. 3.2 again and imagine a desired overall operating point of 11 dB. The capacities of both the second and the third level are 1. Thus, the code rate of the first level can not be reduced without reducing the overall code rate. Generally, the more levels have capacities smaller than 1, the larger the degree of freedom in the design of a UEP scheme.

In contrast, at an operating point of 4 dB, the first level already has a very low rate and it does not make much sense to reduce it even further, in favour of data throughput.

To circumvent the problem at high SNR, where $R_i = 1$, $0 < i \leq l-1$, one can still reduce the code rate of the first level. In order to maintain the throughput, an appropriate amount of information data from the last, least important level can be omitted before encoding which, in fact, leads to a rate $R_{l-1} > 1$ and allows R_0 to be reduced. In the context of scalable multimedia data, where UEP is desired, truncation of the least important data in favour of more important data is a common method. Another solution to the problem are hierarchical signal constellations which are presented in the next section.

3.4 Hierarchical Signal Constellations

In this section, we present UEP-MLC codes in combination with hierarchical signal constellations which may be designed to provide a desired amount of UEP. In [DVB],[HL06], hierarchical modulation is presented, where the symbols are not uniformly distributed in the signal space but are basically grouped such that the single bit positions in a symbol have different (or even desired) error probabilities. Figure 3.5 shows an exemplary hierarchical 16-QAM signal constellation with inter-symbol distances d_0 and d_1. By optimising these distances, or their relation d_0/d_1, the amount of UEP can be varied. However, small losses in the channel capacity have to be accepted for certain SNR

regions.

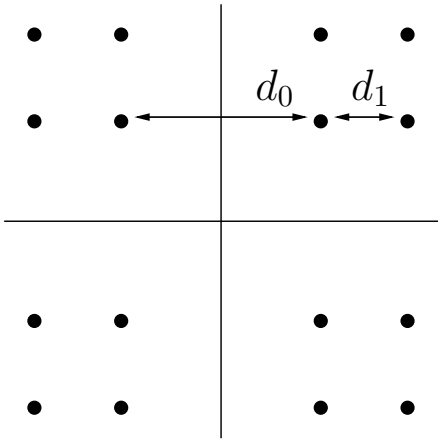


Figure 3.5: Hierarchical 16-QAM signal constellation

Note that this 16-QAM scheme can be seen as an outer and an inner QPSK alphabet. A goal of the constellation design should be that the capacities of the partitioning levels should be more uniform than for standard signal constellations. Especially, the capacity of the first (and second) level should be increased such that shifting operating points is feasible. The labelling or partitioning strategy for this example is chosen such that the first two bits are used to address the 'clouds' of the outer QPSK scheme and the last two bits address a symbol in the respective cloud. The further the clouds lie apart, the more protected are the first two bits. This also means that the capacity of the first two bits is increased with the relation d_0/d_1, while the capacities of the other two bits is decreased. Figure 3.6 shows the capacities of the partitioning levels of a uniform 16-QAM scheme with set partitioning and a hierarchical 16-QAM scheme with $d_0/d_1 = 2$. It is clear that the overall channel capacity of the hierarchical modulation is slightly lower than that of the standard modulation for some SNRs. However, the curves are much more suitable for UEP-MLC than standard modulation, because the distances between the capacities are more uniform and shifting individual operating points is feasible for a much bigger SNR region. Assuming an operating point of $E_s/N_0 = 11$ dB, the capacities are

$$\begin{aligned} C_0 &= 0.98\ , \\ C_1 &= 0.98\ , \\ C_2 &= 0.56\ , \\ C_3 &= 0.56\ . \end{aligned} \tag{3.5}$$

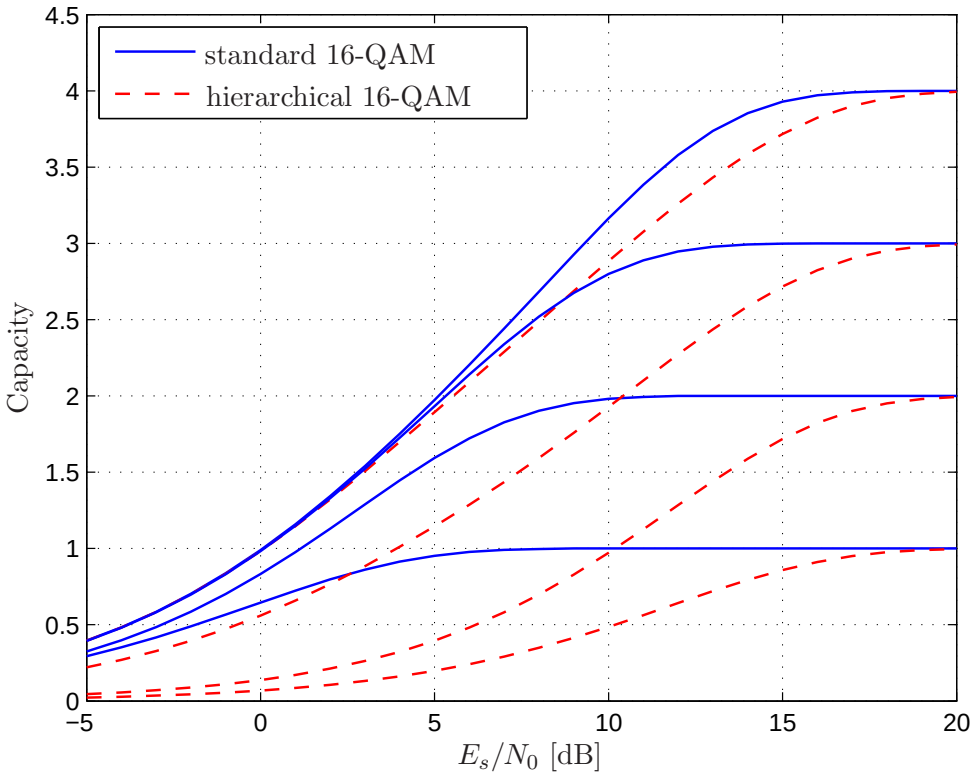


Figure 3.6: Capacity curves of the partitioning levels in a standard 16-QAM scheme with Ungerböck's set partitioning and a hierarchical 16-QAM scheme with $d_0/d_1 = 2$

When shifting the individual operating points, the SNR range where this is sensible and efficient is much larger and the difference in protection between high and low protection classes may be chosen much larger.

Generally, a higher flexibility in the choice of operating points can already be achieved with Gray labelling instead of set partitioning to some extent. However, hierarchical modulation offers an even higher degree of freedom and flexibility. It does not only make UEP design feasible but in a way controllable by optimising the factor d_0/d_1, or even designing this factor differently for real and imaginary signal components.

It should be noted that hierarchical modulation together with conventional MLC (without individual operating points) is not useful because the capacity design rule will 'destroy' the UEP from modulation. Only when creating individual operating points, one will gain from hierarchical modulation.

3.5 Image Transmission Application

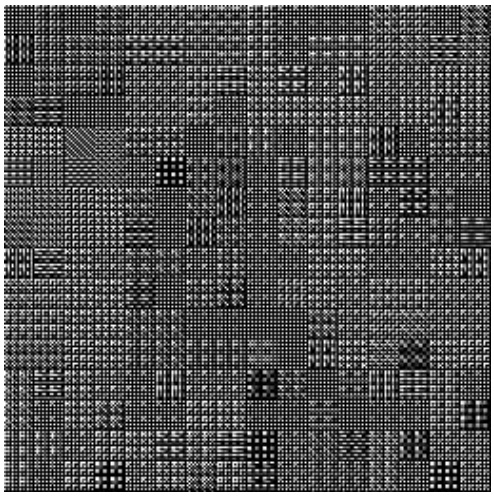
In this section, we show two UEP-MLC examples of the above solution. First, an image is divided into smaller blocks of size 8×8 bytes which are (each) progressively source encoded using the *Embedded Zero-Tree Wavelet* (EZW) algorithm. The algorithm forces the data to be arranged according to their importance, with the most important data first. Afterwards, the sequence of source encoded blocks is assigned to the levels of a UEP-MLC scheme, the most important data are encoded on the first level and so on, and such that the codewords are of equal size on all levels. This means that the fraction of bits assigned to the first level is $R_0/\log_2(M)$, and accordingly for the following levels, where $\log_2(\cdot)$ is the dual logarithm. After modulation, the symbols are transmitted over an AWGN channel, decoded using MSD and fed to the source decoder. In case the receiver detects an error in a block (which can easily be done by an additional error detecting code), the source decoder only considers data from the beginning of the block up to the error location and omits the rest.

We choose an overall operating point of $E_s/N_0 = 11$ dB. According to the shown bit-error rates of a general MLC scheme, the error rates will still be high at exactly the operating point but start to drop immediately. This means bad quality at the receiver side. We verify this by transmitting the Lena image over a conventional MLC system with 8-PSK at $E_s/N_0 = 11$ dB. The codes are designed for an overall operating point exactly at this SNR, i.e., $R_0 = 0.8$, $R_1 = 1$, $R_2 = 1$.

Figures 3.7(a) and 3.7(b) show the original image and the received and decoded image. As expected, the quality is bad and the image is not identifiable. In order to account for the structural properties of the image file, we modify the system to provide UEP by defining the code rates to be $R_0 = 0.375$, $R_1 = 1$, $R_2 = 1.425$, which corresponds to individual operating points of 7.5 dB and 10.5 dB for levels 0 and 1. For level 3, the code rate does not have an equivalent operating point, since the capacity cannot exceed 1. Note that the code rate $R_2 = 1.4125$ is, in fact, accomplished by applying no code at all and omitting the last 29.82% of the third level's information sequence. As codes, again Turbo codes were used together with puncturing and pruning. Figure 3.7(c) shows the resulting image which contains only a few errors in the whole picture.

A similar simulation was done with a 16-QAM scheme. Figure 3.7(d) shows the decoded image after transmission with hierarchical 16-QAM modulation with $d_0/d_1 = 2$, where the operating points were chosen to be 2 dB, 3 dB, 17.5 dB, and 15 dB. In this case,

(a) Original Lena image being transmitted



(b) Decoded Lena image transmitted over a standard 8-PSK MLC scheme

(c) Decoded Lena image transmitted over an 8-PSK UEP-MLC scheme

(d) Decoded Lena image transmitted over a hierarchical 16-QAM UEP-MLC scheme

Figure 3.7: Original image (a), transmitted over a conventional 8-PSK MLC scheme (b), over an 8-PSK UEP-MLC scheme (c), and over a hierarchical 16-QAM UEP-MLC scheme (d) at $E_s/N_0 = 11$ dB.

even the few errors which are still present in Fig. 3.7(c) were able to be corrected.

Chapter 4

UEP Low-Density Parity-Check Codes for Coded Modulation

In this chapter, we present LDPC codes especially designed for higher order constellations. In order to allow for bandwidth-efficient transmission, it is desirable to use higher order modulation techniques, such as M-QAM, M-PSK or more advanced constellations. Modulation with higher order constellations (HOC) may imply that different bits in the symbol have different error probabilities, i.e., the modulation already provides some UEP. As presented in Chapter 3, coded modulation is a well-known strategy to optimise the coding scheme given the modulation to improve the performance of transmission systems in terms of overall bit-error rate, [Ung82b, IH77, Ung87a, Ung87b]. However, applying short codes on each level may have drawbacks (e.g. higher BER) compared to designing one long code whose output bits are appropriately assigned to the levels. Therefore, we employ the latter, taking advantage of the longer code.

This chapter focuses on LDPC codes, originally presented by Gallager in [Gal62]. They exhibit a performance very close to the capacity for the binary-input additive white Gaussian noise (BI-AWGN) channel, [RSU01]. The close-to-optimal performance of LDPC codes for the BI-AWGN channel suggests the use of LDPC codes also for other channels. The aim is to design LDPC codes for transmissions using higher order constellations in applications where the source bits have different sensitivities to errors and UEP is desired.

LDPC codes have been designed for larger constellation sizes in previous works. In

[HSMP03], separate codes have been designed for each level in a multilevel coding scheme. On the contrary, bit-interleaved coded modulation (BICM), [CTB98, SSN04] employ only one code for all levels of the modulation. A design method for discrete multitone modulation is proposed in [SA07]. This method may be applied directly also to HOC and is very similar to the design approach suggested in [SSN04]. In [SSN04] and [SA07], the codes are designed to have local properties that match the higher order constellations and bit positions of the modulation scheme are assigned to the codeword bits.

UEP is commonly provided by coded modulation or adapted code rates, for example by puncturing. These methods have in common that different codes are used for each level of protection. However, for most applications the more important bits are fewer than the less important bits, leading to even shorter information words if these bits are encoded with a separate code with lower rate. To avoid the use of short codewords (or a very long delay), LDPC codes that provide UEP within one codeword may be designed, instead. Such codes can for example be constructed by an algebraic method based on the Plotkin construction, [KM06]. However, since it is widely observed that the connection degree of the variable nodes affects the bit-error rate for a limited number of decoding iterations it is more typical to design the variable and/or check node degree distribution of the code in an irregular way using density evolution, [KSS03a, PDF04a, SHD06, PDF07, RPNF07]. In this case the codeword bits are divided into several protection classes with different protection depending on the connection degrees of their bits. In [SHD06], the check node degree distribution is adapted, keeping the variable node degree distribution fixed, whereas the authors in [PDF07] optimise the irregular variable node degree distribution while keeping the check node degree distribution fixed. The basic idea in [PDF04a, SHD06, PDF07, RPNF07] is to achieve UEP by dividing the degree distributions into sub-distributions. Such sub-distributions have also been employed for systems without UEP capabilities to account for higher order constellations [SSN04, SA07, PNRF05]. In [SSN04], the different amount of protection for each modulation level is taken into account in the initialisation of the density evolution algorithm that is employed to optimise the sub-degree distributions of the code. Both [SA07] and [PNRF05] design LDPC codes for a set of parallel subchannels.

In this chapter, we propose a UEP-LDPC code design for higher order constellations that is based on optimising the variable node degree distribution $\lambda(x)$. Apart from designing the code to account for the UEP provided by the modulation itself and

thereby reducing the overall BER, the aim is to provide a flexible design method that can use the UEP from the modulation to create a code with other UEP properties which are usually specified by the source coding unit. We design UEP-LDPC codes using sub-degree distributions both for the protection classes and for the classes of bits with different protection resulting from the modulation. This allows for a very flexible code design where any conventional modulation scheme, like M-QAM or M-PSK as well as more complex schemes like hierarchical constellations [FR93], may be used. The separation of the variable node degree distribution into sub-degree distributions significantly increases the number of design parameters. Therefore it is important to note that the code design is solved by iterative linear programming (LP), which enables an efficient optimisation of the sub-degree distributions. Our code design is based on the design method for UEP-LDPC codes suggested in [PDF07] that employs iterative LP.

The chapter is organised as follows. In Section 4.1, we briefly present fundamentals on LDPC codes, including the Tanner graph representation, graph-based decoding strategies, and information processing during decoding. Section 4.2 presents the overall system and modulator models. Section 4.3 contains the main contribution, introducing the code optimisation of UEP-capable codes used with higher order constellations and providing an algorithm for the code design. In Section 4.4, some simulation results for 8-PSK and 64-QAM are presented and compared to conventional codes optimised for BPSK channels.

4.1 Fundamentals of LDPC Codes

LDPC codes are linear block codes with a sparse parity-check matrix $\mathbf{H}$ which is of dimension $(N-K)\times N$. The code rate is denoted by $R = K/N$ where K and N are the lengths of the information word and the codeword. An LDPC code can be represented by a bipartite Tanner graph [Tan81] which consists variable nodes v_i, check nodes c_j, and edges e_t. Consider the parity-check matrix below with $N = 10$, $K = 5$, and $R = 1/2$.

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{4.1}$$

The corresponding Tanner graph is shown in Fig. 4.1, where the circles on the left denote the variable nodes and correspond to the code bits, or to the columns of the parity-check matrix. The squares on the right correspond to the parity-check constraints, or to the rows of the parity-check matrix. A variable node v_i is connected to c_j through an edge e_t if and only if the entry $h_{j,i} \neq 0$, which means that the ith code bit is involved in the jth parity-check constraint.

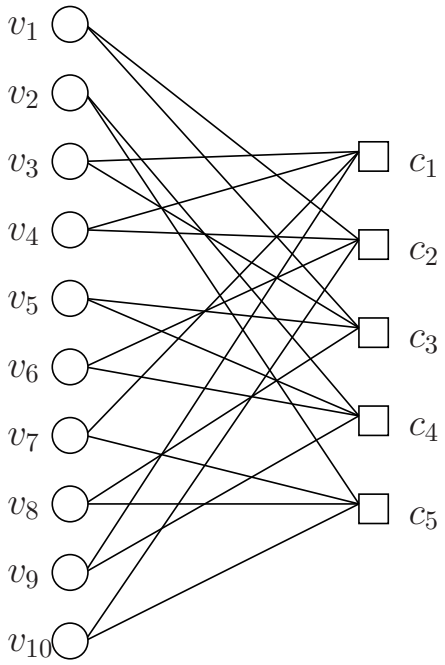


Figure 4.1: (2,4) regular Tanner graph

The number of edges connected to a node is called the degree of the node. A graph is said to be (d_v, d_c) regular if all variable nodes have the same degree d_v and all check nodes have the same degree d_c. Figure 4.1 shows a (2,4) regular realisation. Note that it is not the code which is regular but the parity-check matrix or the corresponding graph.

However, irregular LDPC codes are known to approach capacity more closely than regular LDPC codes. The irregular variable node and check node degree distributions may be defined by the polynomials [RSU01]

$$\lambda(x) = \sum_{i=2}^{d_{v_{max}}} \lambda_i x^{i-1} \tag{4.2}$$

and

$$\rho(x) = \sum_{i=2}^{d_{c_{max}}} \rho_i x^{i-1} \tag{4.3}$$

respectively, where $d_{v_{max}}$ is the maximum variable node degree and $d_{c_{max}}$ is the maximum check node degree of the code. The coefficients λ_i and ρ_i describe the proportion of edges connected to nodes with a certain degree. In order to optimise the degree distribution of an irregular LDPC code, the decoding behaviour has to be investigated. Using a message passing algorithm, the messages along the edges of the graph are updated iteratively. The messages at the input of a variable node and a check node at each iteration represent mutual information and can be computed by means of density evolution using a Gaussian approximation [CRU01] and thereby, the decoding behaviour of a code ensemble with a specific degree distribution may be predicted. Density evolution is an asymptotic tool, however, for simplicity, we use it to design codes of finite length.

The mutual information messages at the input from check nodes to variable nodes x_{cv} and from variable nodes to check nodes x_{vc} at iteration l, computed by means of density evolution using the Gaussian approximation [CRU01], are given by

$$x_{cv}^{(l-1)} = 1 - \sum_{j=2}^{d_{c_{max}}} \rho_j J((j-1)J^{-1}(1 - x_{vc}^{(l-1)})) \,, \tag{4.4}$$

$$x_{vc}^{(l)} = \sum_{i=2}^{d_{v_{max}}} \lambda_i J(\frac{2}{\sigma^2} + (i-1)J^{-1}(x_{cv}^{(l-1)})) \,, \tag{4.5}$$

with $J(\cdot)$ computing the mutual information $x = J(m)$ by

$$\begin{aligned} J(m) &= 1 - \mathbb{E}\{\log_2(1 + e^{-z})\} \\ &= 1 - \frac{1}{\sqrt{4\pi m}} \int_{\mathbb{R}} \log_2(1 + e^{-z}) \cdot e^{-\frac{(z-m)^2}{4m}} dz \end{aligned} \tag{4.6}$$

for a consistent Gaussian random variable $z \sim \mathcal{N}(m, 2m)$ with mean m and variance $2m$. Combining (4.13) and (4.5), one can define the density evolution (DE) as a function of the mutual information of the previous iteration, the noise variance, and the degree distributions:

$$x^{(l)} = F(\lambda, \rho, \sigma^2, x^{(l-1)}) \; . \tag{4.7}$$

Using this density evolution, one may predict the decoding behaviour and optimise the degree distribution under the constraint that the mutual information is increasing with every decoding iteration:

$$F(\lambda, \rho, \sigma^2, x) \overset{!}{>} x \; . \tag{4.8}$$

The decoding of LDPC codes is done in an iterative fashion, updating reliability information at variable nodes and at check nodes alternatingly.

4.2 System Model

4.2.1 Model Description

The independent and identically distributed (i.i.d.) information bits u_i are assigned to $N_p - 1$ protection classes, which are usually defined by the source coding unit and do not have to be of equal size. We apply only one UEP-LDPC code, providing N_p protection classes at its output (see Fig. 4.2), where the parity bits correspond to the least protected class $\mathcal{P}^{N_p}$. The bits of the protection classes are re-multiplexed and assigned to certain bit positions of the modulator, corresponding to modulation classes $\mathcal{M}^1, \ldots, \mathcal{M}^{N_m}$. Bits belonging to one modulation class have the same error probability after the demapper. The bit assignment will be described in Section 4.3.1. In the following, we assume an AWGN channel with noise variance σ^2.

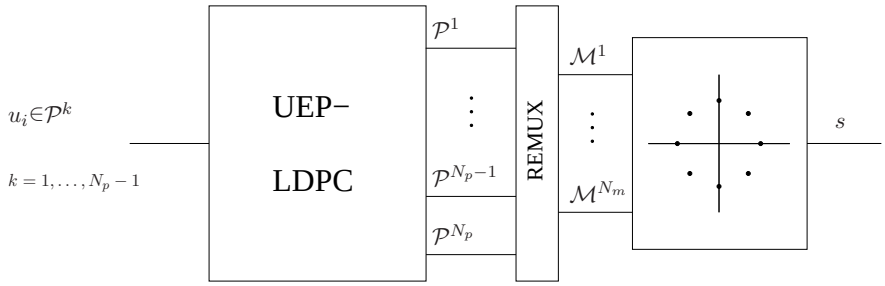


Figure 4.2: Schematic description of the proposed scheme. The source bits are encoded by a UEP-LDPC code and the coded bits are assigned to modulation classes before modulation.

4.2.2 Modulation

Let us assume a modulation scheme with $M_m = 2^{l_m}$ symbols, label ed by binary vectors $\mathbf{d} = (b_{l_m-1}, \ldots b_1, b_0)$. In order to design codes for HOCs, we investigate the error probabilities of the individual bit positions in the symbol. Depending on the signal constellation and the labelling strategy, the error probabilities of the individual bit positions may be determined.

One may use traditional constellations like 8-PSK or 64-QAM as well as non-uniform constellations, so-called hierarchical modulation [LH06] or multiresolution modulation [FR93]. When UEP is desired, the latter have the advantage that they can provide more UEP to the bit positions in the symbol than traditional constellations. Furthermore, the design methods in the above references allow for a controllable amount of UEP through variations in the distances between the signal points.

Generally, one can approximate the symbol-error probability P_s as well as the bit-error probabilities $P_{b,b_0} \ldots P_{b,b_{l_m-1}}$ of a constellation using the union bound. For Gray labelling, the average bit-error probability is often assumed to be equal for all bit positions, i.e., $\tilde{P}_{b,b_i} \approx \frac{1}{\log_2(M)} \cdot P_s$. However, it is important to be aware of the different bit-error probabilities when designing codes for HOCs. In this work we only consider Gray labelling, since the bit-errors have low statistical dependencies and can thus be assumed to be independent [SWBK03], which is very important for the message passing decoder. However, for labellings other than Gray, the differences in BER may be more significant. If such labellings are of interest, we suggest the use of the improved decoding algorithm proposed in [NSL06] that combines decoding and demodulation and takes into account the statistical dependencies among bit errors originating in the

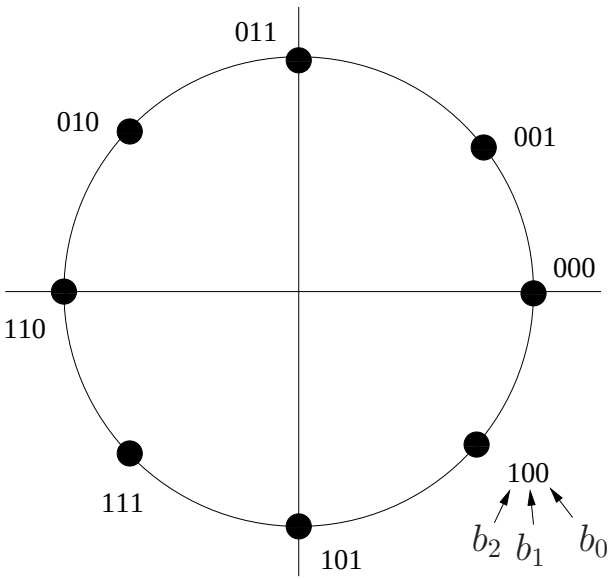


Figure 4.3: 8-PSK modulation with Gray labelling.

same symbol.

In [WFH99], it was stated that a symmetric channel with input alphabet of size 2^{l_m} can be represented by l_m equivalent binary-input channels. Depending on the labelling, the equivalent channels may not be symmetric. Since the symmetry of the channel is an important property for the density evolution of LDPC code ensembles, we use a method called *i.i.d. channel adapters* [HSMP01]. In order to force symmetric component channels, i.i.d. binary vectors $\mathbf{t} = [t_1 \dots t_n]$ are added modulo-2 to the codeword. At the receiver, the log-likelihood ratio L_j is multiplied by -1 if $t_j = 1$. Since the equivalent component codes are now symmetric, one can assume its behaviour being equal to BPSK signalling. From the known bit-error probabilities of the binary component channels, it is easy to determine their equivalent noise variances σ_i^2 which are required by the density evolution later on. We define the noise vector $\boldsymbol{\sigma}^2 = [\sigma_1^2 \ \dots \ \sigma_{N_m}^2]$ to be a vector that contains the equivalent noise variances for each separate bit-error rate, i.e., for each modulation class, ordered with the lowest variance first. We assume that there are N_m distinct equivalent noise variances, where $N_m \leq l_m$.

Example 4.1 (8-PSK) *8-PSK modulation with Gray labelling is shown in Fig. 4.3. Considering only neighboring signal points, the approximate symbol-error rate expres-*

sion for this modulation is given as [Pro01],

$$P_{s,8-PSK} = \operatorname{erfc}\left(\sqrt{3 \cdot E_b/N_0}\sin\frac{\pi}{8}\right) , \tag{4.9}$$

where $\operatorname{erfc}(\cdot)$ *is the complementary error function. The average bit-error rate is* $\tilde{P}_b \approx P_s/\log_2(M)$*. The expressions for the bit-error probabilities of the individual bits in the symbol are*

$$P_{b,b_0} \approx \frac{1}{2} \cdot \operatorname{erfc}\left(\sqrt{3 \cdot E_b/N_0}\sin\frac{\pi}{8}\right) , \tag{4.10}$$

$$P_{b,b_1} = P_{b,b_2} \approx \frac{1}{4} \cdot \operatorname{erfc}\left(\sqrt{3 \cdot E_b/N_0}\sin\frac{\pi}{8}\right) . \tag{4.11}$$

Note that the above expressions are obtained by applying approximations. The approximations are assumed to be appropriate for our purposes but can be replaced by more exact formulae. □

From the different bit-error probabilities, one can determine equivalent noise variances of the single bit positions assuming different BPSK channels,

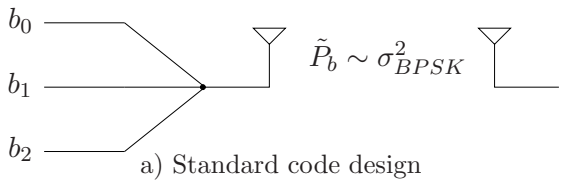
$$\sigma_j^2 = \frac{1}{2\left(\operatorname{erfc}^{-1}(2P_{b,b_j})\right)^2} . \tag{4.12}$$

Usually, LDPC codes are designed for one noise variance and it is assumed that all bits are transmitted over the corresponding channel. In this paper we design codes using the equivalent noise variances from (4.12). To compare the standard code design with our new design in a fair way, we calculate an equivalent BPSK noise variance, denoted by σ_{BPSK}^2, which is the noise variance of a BPSK channel that would give a bit-error probability equal to the average bit-error probability of the HOC. Fig. 4.4 shows a schematic explanation of the channel assumptions made by the receiver.

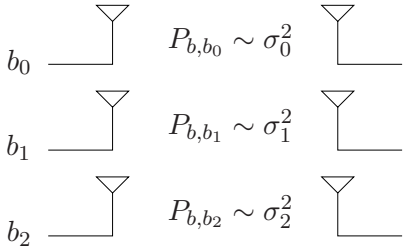
In the following, we assume that N_m equivalent BPSK channels are employed together with i.i.d. channel adapters instead of the HOC channel and we claim that this approximation meets our requirements.

4.2.3 Notations

We consider a UEP-LDPC code with N_p protection classes. The proportions of each class are given by the normalised lengths of each class corresponding to the information



a) Standard code design



b) Design for HOCs

Figure 4.4: Channel assumptions for a) standard code design b) design for HOCs.

bits, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{N_p-1}]$, i.e., α_i equals the number of bits belonging to protection class $\mathcal{P}^i$ divided by the total number of information bits k. The proportion distribution of the bits in the codeword belonging to the protection classes is, thus, given by $\mathbf{p} = [\alpha_1 R, \ldots, \alpha_{N_p-1} R, (1-R)]$. N_m is the number of different bit-error rates for the bits in a symbol and we will describe the bits with a distinct bit-error rate as belonging to one modulation class $\mathcal{M}^j$. $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{N_m}]$ defines the proportion of bits that belongs to each modulation class.

The vector $\boldsymbol{\lambda}$ contains the overall variable node degree distribution, both for different protection classes and different modulation classes. Let $\lambda^{\mathcal{P}^k}_{\mathcal{M}^j,i}$ be the proportion of edges connected to variable nodes of degree i that belong to modulation class $\mathcal{M}^j$ and protection class $\mathcal{P}^k$. Define

$$\boldsymbol{\lambda}^{\mathcal{P}^k}_{\mathcal{M}^j} = [\lambda^{\mathcal{P}^k}_{\mathcal{M}^j,2}, \ldots, \lambda^{\mathcal{P}^k}_{\mathcal{M}^j,d_{v_{max}}}]^T$$

and

$$\boldsymbol{\lambda} = \left[\boldsymbol{\lambda}^{\mathcal{P}^1}_{\mathcal{M}^1}{}^T, \ldots, \boldsymbol{\lambda}^{\mathcal{P}^{N_p}}_{\mathcal{M}^1}{}^T, \ldots, \boldsymbol{\lambda}^{\mathcal{P}^1}_{\mathcal{M}^{N_m}}{}^T, \ldots, \boldsymbol{\lambda}^{\mathcal{P}^{N_p}}_{\mathcal{M}^{N_m}}{}^T \right]^T,$$

where $(\cdot)^T$ denotes the transpose. $\boldsymbol{\lambda}^{\mathcal{P}^k}_{\mathcal{M}^j}$ is a column vector of length $d_{v_{max}} - 1$ and $\boldsymbol{\lambda}$ is a vector of size $((d_{v_{max}} - 1) \cdot N_p \cdot N_m \times 1)$. The vector

$$\boldsymbol{\rho} = [\rho_2, \ldots, \rho_{d_{c_{max}}}]^T$$

describes the check node degree distribution and we define **1** to be an all-ones vector of appropriate length.

4.3 UEP-LDPC Codes for Higher Order Constellations

4.3.1 Optimisation of the Degree Distribution

The mutual information messages from check nodes to variable nodes x_{cv} and from variable nodes to check nodes x_{vc} at iteration l, computed by means of density evolution using the Gaussian approximation [CRU01], are given by

$$x_{vc}^{(l-1)} = 1 - \sum_{j=2}^{d_{cmax}} \rho_j J((j-1)J^{-1}(1 - x_{cv}^{(l-1)})) \; , \tag{4.13}$$

$$x_{vc}^{(l)} = \sum_{j=1}^{N_m} \sum_{k=1}^{N_p} \sum_{i=2}^{d_{vmax}} \lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k} J(\frac{2}{\sigma_j^2} + (i-1)J^{-1}(x_{cv}^{(l-1)})) \; . \tag{4.14}$$

The update rule for the messages from check nodes to variable nodes is not affected by the division of the variable node degree distribution into sub-distributions since the check node degree distribution is constant for all protection classes and modulation classes. Equations (4.13) and (4.14) can be combined to yield the mutual information evolution of the LDPC code

$$x_{vc}^{(l)} = F(\boldsymbol{\lambda}, \boldsymbol{\rho}, \boldsymbol{\sigma}^2, x_{vc}^{(l-1)}) \; . \tag{4.15}$$

If $x_{vc}^{(l)} > x_{vc}^{(l-1)}$ for any $x_{vc}^{(l-1)}$, then $\boldsymbol{\lambda}$ and $\boldsymbol{\rho}$ describe a code for which density evolution converges for the noise variance vector $\boldsymbol{\sigma}^2$. Since the variable node degree distributions give proportions of edges connected to variable nodes of certain degrees, the constraint

$$\sum_{j=1}^{N_m} \sum_{k=1}^{N_p} \sum_{i=2}^{d_{vmax}} \lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k} = 1 \tag{4.16}$$

must be fulfilled.

UEP capabilities may be obtained by optimising each protection class after another by linear programming, starting with the best protected class and fixing the degree distributions of the already optimised classes during the optimisation of the following classes, [PDF07]. It is well-known that a higher connectivity of a variable node leads to better protection. Thus, the optimisation target is to find a variable node degree distribution for the whole code that maximises the average variable node degree of the class being optimised. When the proportion of edges in the parity check matrix that are associated with a specific class is high, many messages will be passed and the local convergence of the graph is fast. However, in the limit when the number of decoding iterations tends to infinity, all messages have transit the whole graph and there should be no UEP capability left. This implies that the UEP capability theoretically should be decreasing with increasing number of iterations. However, our results show that for a reasonable number of iterations the UEP capability is not affected much by the number of decoding iterations. Figure 4.9 in Section 4.4 shows the BER as a function of the number of decoding iterations for the code design suggested in this paper. In [PDF07], the authors show that it is not only the average connection degree to be maximised but also the minimum variable node degree. Therefore, the second optimisation target of our code design is the maximisation of the minimum variable node degree of each protection class.

The aim of the code design is to reduce the overall BER by taking the different error probabilities of the modulation levels into account. The design algorithm should also give the possibility to trade overall BER for UEP capability. The natural way of assigning bits from modulation classes to protection classes to achieve UEP, is to use the best protected bits from the modulation, that is, modulation class $\mathcal{M}^1$, for protection class $\mathcal{P}^1$ and continue like that until all bits have been assigned to a protection class. However, this assignment is not necessarily expected to give a degree distribution with the lowest possible threshold, where the threshold is defined as the lowest E_b/N_0 for which density evolution converges. As discussed later, there is always a trade-off between a low threshold and good UEP capabilities. By distinguishing not only between degree polynomial coefficients for different protection classes but also for different modulation classes, linear programming may be used to assign bits from the modulation classes to the protection classes.

In order to obtain UEP capabilities, the optimisation target is to maximise the average variable node degree of the protection class currently optimised while maximising the minimum variable node degree of the class, [PDF07]. The target function for protection

class $\mathcal{P}^k$ can be formulated as

$$\max_{\boldsymbol{\lambda}} \sum_{j=1}^{N_m} \sum_{i=2}^{d_{vmax}} \lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k} \,. \tag{4.17}$$

This target function will ensure a maximised average variable node degree, but in order to also maximise the minimum variable node degree of the class, the optimisation is first performed for a high minimum variable node degree. If no degree distribution for which density evolution converges can be found, the optimisation is repeated for a lower minimum variable node degree as in [PDF07]. Repeated optimisation with the target function (4.17) results in a degree distribution with UEP capabilities, but the target function does not account for the fact that variable nodes connected to the first modulation class are more valuable than those from worse modulation classes. Therefore, we introduce a scaling factor k_j for each modulation class which decreases with the modulation class index, i.e., $k_1 > k_2 > \ldots > k_{N_m} > 0$.

$$\max_{\boldsymbol{\lambda}} \sum_{j=1}^{N_m} k_j \sum_{i=2}^{d_{vmax}} \lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k} \tag{4.18}$$

The choice of this scaling factor affects how valuable a better modulation class is compared to a modulation class with higher equivalent noise variance. Although differences in the degree distribution occur for different choices of k_j, the differences in BER are negligible and, for simplicity, k_j might be chosen as $k_j = N_m - j + 1$. The factor k_j has the effect that the linear programming algorithm, if possible while fulfilling all constraints, will use modulation classes with low noise variance for the protection classes being optimised.
Besides the optimisation constraints given in Equations (4.15) and (4.16), there are some more constraints which have to be fulfilled by the optimised $\boldsymbol{\lambda}$. The variable node degree distribution is connected to the check node degree distribution and the code rate by

$$R = 1 - \frac{\sum_{j=2}^{d_{cmax}} \rho_j / j}{\sum_{i=2}^{d_{vmax}} \lambda_i / i} \,. \tag{4.19}$$

Furthermore, the proportion vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ impose the following two constraints on $\boldsymbol{\lambda}$

$$N_{\mathcal{P}^k} = \alpha_k \cdot R \cdot N \quad k = 1, \ldots, N_p - 1, \quad \text{and} \tag{4.20}$$

$$N_{\mathcal{M}^j} = \beta_j \cdot N \tag{4.21}$$

where $N_{\mathcal{P}^k}$ and $N_{\mathcal{M}^j}$ denote the total number of variable nodes belonging to protection class $\mathcal{P}^k$ and to modulation class $\mathcal{M}^j$, respectively. $N_{\mathcal{P}^k}$ and $N_{\mathcal{M}^j}$ are connected to $\boldsymbol{\lambda}$ by

$$N_{\mathcal{P}^k} = \sum_{j=1}^{N_m} \sum_{i=2}^{d_{vmax}} \frac{\lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k}}{i} \cdot N \cdot (1-R) / \left(\sum_{i=2}^{d_{cmax}} \frac{\rho_i}{i} \right) \quad \text{and} \tag{4.22}$$

$$N_{\mathcal{M}^j} = \sum_{k=1}^{N_p} \sum_{i=2}^{d_{vmax}} \frac{\lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k}}{i} \cdot N \cdot (1-R) / \left(\sum_{i=2}^{d_{cmax}} \frac{\rho_i}{i} \right) , \tag{4.23}$$

respectively.

When designing good LDPC code ensembles, the stability condition which ensures convergence of the density evolution for mutual information close to one should be fulfilled [RSU01]. The stability condition gives an upper bound on the number of degree-2 variable nodes. For a BPSK scheme, where all bits are affected by the same noise variance, we have [RSU01]

$$\frac{1}{\lambda'(0)\rho'(1)} > e^{-r} = \int_{\mathbb{R}} P_0(x) e^{-\frac{x}{2}} dx = e^{-\frac{1}{2\sigma^2}} \tag{4.24}$$

with $P_0(x)$ being the message density corresponding to the received values and $\lambda'(x)$ and $\rho'(x)$ being the derivatives of the degree polynomials. It is straightforward to see that $\lambda'(0) = \sum_{j=1}^{N_m} \sum_{k=1}^{N_p} \lambda_{\mathcal{M}^j,2}^{\mathcal{P}^k}$ and $\rho'(1) = \sum_{m=2}^{d_{cmax}} \rho_m \cdot (m-1)$. In our case, the bits are affected by channel noise with different equivalent noise variances σ_j^2 (see (4.12)) and thus different densities. We use the average density, which is given by utilising the modulation class proportions $\boldsymbol{\beta}$,

$$e^{-r} = \int_{\mathbb{R}} \sum_{j=1}^{N_m} \beta_j \cdot P_{0,j}(x) e^{-\frac{x}{2}} dx = \sum_{j=1}^{N_m} \beta_j \cdot e^{-\frac{1}{2\sigma_j^2}} . \tag{4.25}$$

We note that this is an approximation, but we assume it to be sufficient for the ensemble of code constructions with given $\boldsymbol{\beta}$.
All above given constraints will be used in the algorithm, rewritten only containing $\boldsymbol{\lambda}$,

$\boldsymbol{\rho}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\sigma}^2$, and R, i.e., without using N and K.

4.3.2 Optimisation Algorithm

The optimisation algorithm proposed here is a modification of the hierarchical optimisation algorithm presented in [PDF07] for HOCs. The optimisation is performed at $E_b/N_0 = \delta + \epsilon$, which will be the threshold of the optimised code, where δ is the lowest possible threshold in dB for the given $\boldsymbol{\rho}$ and $d_{v_{max}}$, and ϵ is the offset from the lowest threshold that provides more flexibility in the choice of $\boldsymbol{\lambda}$ for the code design. For $\epsilon = 0$, the designed code has some inherent UEP by assigning higher degree variable nodes to the more protected classes. On the contrary, by introducing $\epsilon > 0$, we allow for more freedom in the code design, assigning even higher degrees or larger fractions of edges to the high degrees of the first class and accepting worse properties for the less important protection classes. Hence, $\epsilon > 0$ leads to increased UEP compared to the inherent UEP of the 'minimum threshold code'.

The algorithm can be divided into an outer loop and an inner loop. For a given E_b/N_0, the outer loop runs over the protection classes, starting from the first. At this point, the degree distribution of the code is designed while optimising the corresponding protection class. As mentioned before, there are two target functions to be maximised, i.e., the average connection degree and the minimum connection degree of the class' variable nodes. Since we are using LP with only a single objective function, we choose it to be the maximisation of the average connection degree. The maximisation of the minimum variable node degree is performed by the inner loop which runs over different values for the minimum degree, starting from some maximum value and successively reducing it during the procedure. Once a valid solution is found for the current protection class and minimum degree, the next protection class $\mathcal{P}^k$ is optimised, assuming that classes $\mathcal{P}^1$, ..., $\mathcal{P}^{k-1}$ have already been optimised and that $\boldsymbol{\lambda}_{\mathcal{M}^j}^{\mathcal{P}^1}$, ..., $\boldsymbol{\lambda}_{\mathcal{M}^j}^{\mathcal{P}^{k-1}}$, $\forall j$, are fixed.

The optimisation algorithm can be stated as follows.

I) Fix $E_b/N_0 = \delta + \epsilon$ and calculate $\boldsymbol{\sigma}^2$.

II) Find $\boldsymbol{\lambda}$ by performing the inner optimisation loop for each protection class.

For the optimisation of class $\mathcal{P}^k$, a linear-programming routine is executed, which requires the definition of the check node degree distribution $\boldsymbol{\rho}$, $E_b/N_0 = \delta + \epsilon$ in dB, the maximum variable node degree $d_{v_{max}}$, and the code rate R. The optimisation algorithm is presented on page 65.

4.3.3 Code Construction

When the optimal degree distribution of the variable nodes is found, a parity-check matrix is constructed by a modification of the approximate cycle extrinsic (ACE) message degree algorithm [TJVW04]. The ACE algorithm constructs a parity-check matrix following a given variable node degree distribution while selectively avoiding small cycle clusters that are isolated from the rest of the graph. The ACE algorithm has good performance in the error floor region for irregular LDPC codes.

The original ACE algorithm [TJVW04] only ensures a certain variable node degree distribution. However, the design algorithm optimises the variable node degree distribution given a certain check node degree distribution and a parity-check matrix with degrees given by $\boldsymbol{\lambda}$ and $\boldsymbol{\rho}$ is desired. The modified ACE construction of the parity-check matrix also ensures that the check node degree distribution equals $\boldsymbol{\rho}$. Whereas the ones in a column are located at random positions in the original ACE algorithm, we allow only those positions where a one does not violate the defined check node degree distribution.

4.4 Simulation Results

In this section, simulation results for examples with 8-PSK and 64-QAM are presented. We denote our scheme by "HOC-UEP", which is a UEP-LDPC code optimised for the different σ_j^2 from the modulation. The HOC-UEP scheme is compared to a UEP-LDPC code optimised for BPSK [PDF07]. This scheme, that is denoted by "UEP", designs the code for the comparable σ_{BPSK}^2 (see Section 4.2.2) and assign the best bits from modulation to the first protection class and vice versa. Our scheme is also compared to a design without UEP similar to the design proposed in [SSN04]. The variable node degree distributions are optimised for $R = 1/2$, $N_p = 3$, $\boldsymbol{\alpha} = [0.3,\ 0.7]$, $d_{v_{max}} = 30$, and $\rho(x) = 0.00749x^7 + 0.99101x^8 + 0.00150x^9$, which is found by numerical optimisation

1) Initialisation $d_{v_{min}}^{(k)} = d_{v_{max}}$

2) While optimisation failure

a) Under the constraints $[\mathcal{P}^1] - [\mathcal{P}^6]$, optimise

$$\max_{\boldsymbol{\lambda}} \sum_{j=1}^{N_m} k_j \sum_{i=2}^{d_{vmax}} \lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k} \tag{4.26}$$

$[\mathcal{P}^1]$ Rate constraint

$$\sum_{j=1}^{N_m} \sum_{k=1}^{N_p} \sum_{i=2}^{d_{vmax}} \frac{\lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k}}{i} = \frac{1}{1-R} \sum_{i=2}^{d_{cmax}} \frac{\rho_i}{i} \tag{4.27}$$

$[\mathcal{P}^2]$ Proportion distribution constraints

i)
$$\sum_{j=1}^{N_m} \sum_{k=1}^{N_p} \boldsymbol{\lambda}_{\mathcal{M}^j}^{\mathcal{P}^k\ T} \mathbf{1} = 1 \tag{4.28}$$

ii) $\forall k \in \{1, \ldots, N_p - 1\}$,

$$\sum_{j=1}^{N_m} \sum_{i=2}^{d_{vmax}} \frac{\lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k}}{i} = \alpha_k \frac{R}{1-R} \sum_{i=2}^{d_{cmax}} \frac{\rho_i}{i} \tag{4.29}$$

iii) $\forall j \in \{1, \ldots, N_m - 1\}$,

$$\sum_{k=1}^{N_p} \sum_{i=2}^{d_{vmax}} \frac{\lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k}}{i} = \beta_j \frac{1}{1-R} \sum_{i=2}^{d_{cmax}} \frac{\rho_i}{i} \tag{4.30}$$

$[\mathcal{P}^3]$ Convergence constraint, see (4.15)

$$F(\boldsymbol{\lambda}, \boldsymbol{\rho}, \boldsymbol{\sigma}^2, x) > x \tag{4.31}$$

$[\mathcal{P}^4]$ Stability condition, see (4.24) and (4.25)

$$\sum_{j=1}^{N_m} \sum_{k=1}^{N_p} \lambda_{\mathcal{M}^j,2}^{\mathcal{P}^k} < \left[\sum_{j=1}^{N_m} \beta_j e^{-1/2\sigma_j^2} \cdot \sum_{m=2}^{d_{cmax}} \rho_m (m-1) \right]^{-1} \tag{4.32}$$

$[\mathcal{P}^5]$ Minimum variable node degree constraint

$$\forall i < d_{v_{min}}^{(k)},\ \forall j :\ \lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k} = 0 \tag{4.33}$$

$[\mathcal{P}^6]$ Previous optimisation constraints

$$\forall k' < k,\ \forall j :\ \boldsymbol{\lambda}_{\mathcal{M}^j}^{\mathcal{P}^{k'}} \quad \text{is fixed} \tag{4.34}$$

b) $d_{v_{min}}^{(k)} = d_{v_{min}}^{(k)} - 1$

End (While)

Table 4.1: Degree distributions for the UEP scheme with $N_p = 3$ protection classes.

	$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\epsilon = 0$ dB	$\lambda_7 = 0.0799$	$\lambda_3 = 0.1790$	$\lambda_2 = 0.2103$
	$\lambda_8 = 0.0948$	$\lambda_6 = 0.0737$	$\lambda_3 = 0.0181$
	$\lambda_{30} = 0.3029$	$\lambda_7 = 0.0414$	
$\epsilon = 0.1$ dB	$\lambda_{11} = 0.1783$	$\lambda_3 = 0.2041$	$\lambda_2 = 0.1841$
	$\lambda_{12} = 0.1184$	$\lambda_4 = 0.0393$	$\lambda_3 = 0.0575$
	$\lambda_{30} = 0.2183$		

in [RSU01] to be a good check node degree distribution for $d_{v_{max}} = 30$.

4.4.1 Results for 8-PSK

For Gray-label ed 8-PSK, the noise vector $\boldsymbol{\sigma}^2$ is calculated according to (4.12), with $N_m = 2$ and $\boldsymbol{\beta} = [2/3,\ 1/3]$. Table 4.1 shows the degree distributions given by the UEP scheme. We arbitrarily choose $\epsilon = 0.1$ dB to allow for some increased UEP. The resulting degree distributions $\boldsymbol{\lambda}^{\mathcal{P}^k}$ are given for each protection class $\mathcal{P}^k$. For comparison, the degree distribution for the minimum threshold, that is $\epsilon = 0$ dB, is also shown. This degree distribution corresponds to the code design with inherent UEP only. The degree distributions of the HOC-UEP scheme, $\boldsymbol{\lambda}^{\mathcal{P}^k}_{\mathcal{M}^j}$, are given in Table 4.2.

In the algorithm, we have chosen $k_j = N_m - j + 1$ in (4.26) and this means that the best modulation class is favoured when designing the best protection class, which should give more UEP capability. For comparison we also design a code with $k_j = 1$ and only one information class without UEP capability, which is similar to the approach in [SSN04]. However, the approach in [SSN04] does not separate the degree distributions for information bits and parity bits. The degree distributions are shown in Table 4.3.

Finite-length codeword simulations with $N = 4096$ and 50 decoding iterations are performed using the equivalent BPSK channels. Simulations verify that 8-PSK modulation and demodulation give almost exactly the same results as the equivalent BPSK channels. We assume that a soft demapper provides the message passing decoder with the channel log-likelihood ratios (LLRs) in any case using HOC modulation and demodulation. Note that the channel LLRs are computed using the appropriate noise variances σ_j^2 of the modulation classes.

Table 4.2: Degree distributions for the HOC-UEP scheme for 8-PSK with $N_p = 3$ protection classes and $N_m = 2$ modulation classes.

		$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\epsilon = 0$ dB	$\mathcal{M}^1$	$\lambda_9 = 0.1703$ $\lambda_{10} = 0.0555$ $\lambda_{30} = 0.1811$	$\lambda_3 = 0.1673$	$\lambda_2 = 0.1240$
	$\mathcal{M}^2$	$\lambda_{30} = 0.0854$	$\lambda_4 = 0.0225$ $\lambda_5 = 0.0738$ $\lambda_7 = 0.0117$	$\lambda_2 = 0.0878$ $\lambda_3 = 0.0022$ $\lambda_4 = 0.0183$
$\epsilon = 0.1$ dB	$\mathcal{M}^1$	$\lambda_{12} = 0.3290$ $\lambda_{30} = 0.1782$	$\lambda_3 = 0.1070$	$\lambda_2 = 0.1585$
	$\mathcal{M}^2$		$\lambda_3 = 0.0396$ $\lambda_4 = 0.1026$ $\lambda_5 = 0.0165$	$\lambda_2 = 0.0547$ $\lambda_3 = 0.0137$
$\epsilon = 0.2$ dB	$\mathcal{M}^1$	$\lambda_{15} = 0.4840$ $\lambda_{30} = 0.0327$	$\lambda_3 = 0.0848$	$\lambda_2 = 0.1733$
	$\mathcal{M}^2$		$\lambda_3 = 0.0782$ $\lambda_4 = 0.0940$	$\lambda_2 = 0.0414$ $\lambda_3 = 0.0115$
$\epsilon = 0.3$ dB	$\mathcal{M}^1$	$\lambda_{16} = 0.4993$	$\lambda_3 = 0.0268$	$\lambda_2 = 0.2163$
	$\mathcal{M}^2$	$\lambda_{16} = 0.0345$	$\lambda_3 = 0.1849$ $\lambda_4 = 0.0291$	$\lambda_2 = 0.0092$

Fig. 4.5 shows the overall BER after 50 decoding iterations averaged over all protection classes for both $\epsilon = 0$ dB and $\epsilon = 0.1$ dB. It is seen that the overall BERs of the HOC-UEP codes are lower than for the UEP codes. For an overall BER of 10^{-4}, there is a gain of around 0.7 dB by the HOC-UEP scheme which is due to the consideration of the different σ_j^2 in contrast to only assuming an average σ_{BPSK}^2. The overall BER for the design similar to the approach in [SSN04] is shown for comparison even though this scheme has no UEP capability. By design, the overall BERs for the codes with $\epsilon \neq 0$ dB are higher than for the corresponding codes with $\epsilon = 0$ dB, because the thresholds of the codes are increased in order to allow an increased average variable node degree of the most protected classes. However, our results for the HOC-UEP scheme show a slightly lower overall BER for the case with $\epsilon = 0.1$ dB compared to $\epsilon = 0$ dB. Reasons for this are discussed later in this section.

Table 4.3: Degree distributions for the design without UEP for 8-PSK.

		Information bits		Parity bits
$\epsilon = 0$ dB	$\mathcal{M}^1$	$\lambda_3 = 0.1474$	$\lambda_4 = 0.0008$	$\lambda_2 = 0.1237$
		$\lambda_5 = 0.0021$	$\lambda_6 = 0.0008$	$\lambda_3 = 0.0154$
		$\lambda_7 = 0.0023$	$\lambda_8 = 0.0005$	
		$\lambda_9 = 0.2204$	$\lambda_{10} = 0.0040$	
		$\lambda_{11} = 0.0008$	$\lambda_{12} = 0.0004$	
		$\lambda_{13} = 0.0002$	$\lambda_{14} = 0.0002$	
		$\lambda_{15-25} = 0.0013$	$\lambda_{26} = 0.0002$	
		$\lambda_{27} = 0.0003$	$\lambda_{28} = 0.0004$	
		$\lambda_{29} = 0.0011$	$\lambda_{30} = 0.1751$	
	$\mathcal{M}^2$	$\lambda_3 = 0.0019$	$\lambda_4 = 0.0509$	$\lambda_2 = 0.0881$
		$\lambda_5 = 0.0581$	$\lambda_6 = 0.0049$	$\lambda_3 = 0.0005$
		$\lambda_7 = 0.0008$	$\lambda_8 = 0.0057$	
		$\lambda_9 = 0.0017$	$\lambda_{10} = 0.0006$	
		$\lambda_{11} = 0.0003$	$\lambda_{12} = 0.0002$	
		$\lambda_{13} = 0.0001$	$\lambda_{14} = 0.0001$	
		$\lambda_{15-25} = 0.0008$	$\lambda_{26} = 0.0001$	
		$\lambda_{27} = 0.0002$	$\lambda_{28} = 0.0003$	
		$\lambda_{29} = 0.0005$	$\lambda_{30} = 0.0869$	

The BER performance of the individual protection classes $\mathcal{P}^1$ and $\mathcal{P}^2$ for the UEP scheme are shown in Fig. 4.6. Note that we do not show protection class $\mathcal{P}^3$ because it contains parity bits only and is of little interest for possible applications. As expected, the UEP capability, i.e., the difference in BER between class $\mathcal{P}^1$ and $\mathcal{P}^2$, is increased slightly with increasing ϵ. Generally, the UEP capability is accomplished by assignment of high degree variable nodes to the most protected classes.

Fig. 4.7 shows the BER performance of protection classes $\mathcal{P}^1$ and $\mathcal{P}^2$ for the HOC-UEP scheme. The results show that the HOC-UEP $\epsilon = 0.1$ dB code has lower BER for both information classes than the HOC-UEP $\epsilon = 0$ dB code. Note that the differences in BER are partly balanced by protection class $\mathcal{P}^3$ which is not shown here. By comparing Fig. 4.6 and Fig. 4.7, we see that the shown BERs of the HOC-UEP scheme are much lower than those of the UEP scheme for higher SNR. The gain of the HOC-UEP scheme

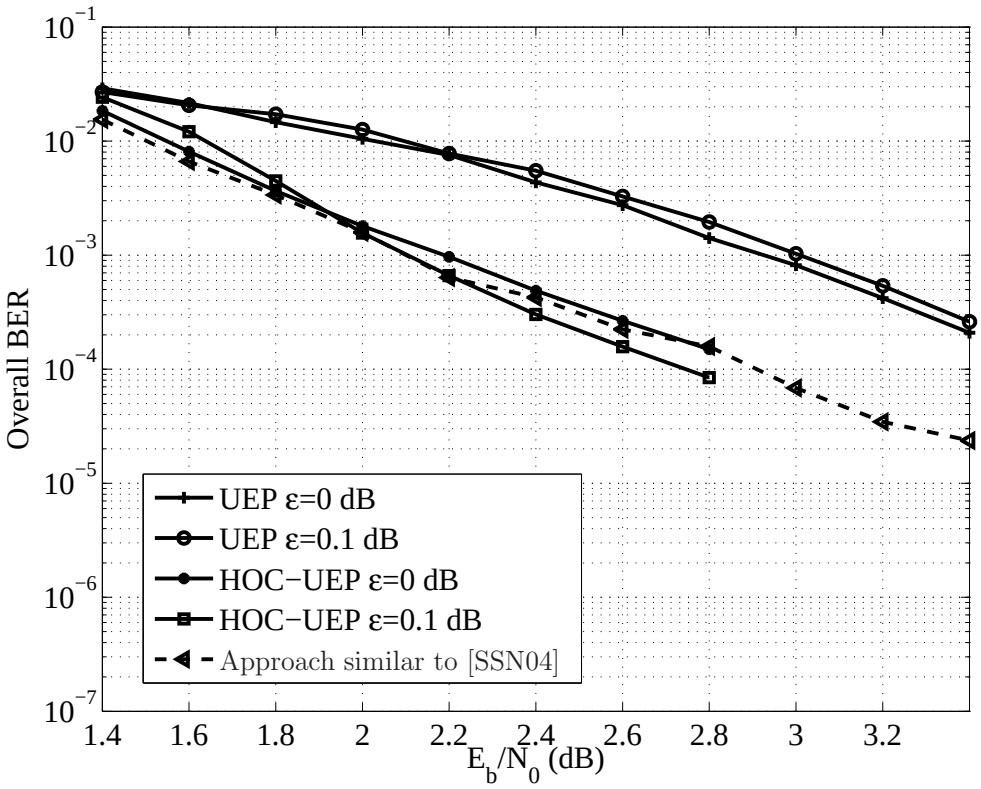


Figure 4.5: Overall bit-error rate performance of the UEP scheme and the HOC-UEP scheme for $\epsilon = 0$ dB and $\epsilon = 0.1$ dB.

compared to the UEP scheme is 0.8 dB for both protection classes $\mathcal{P}^1$ and $\mathcal{P}^2$, at a BER of 10^{-6} and 10^{-4}, respectively. We also see that the difference in performance between the protection classes is slightly higher for the HOC-UEP scheme than for the UEP scheme, especially for high SNR. Because of the large difference in BER of the protection classes and also the knowledge of the proportion of bits in the classes, we know that it is mainly $\mathcal{P}^2$ that governs the performance in terms of the overall BER. By comparing also with Fig. 4.5, it is seen that the overall BER of the approach similar to [SSN04] without UEP is almost equal to the overall BER of the HOC-UEP scheme with $\epsilon = 0$ dB and worse than the HOC-UEP scheme with $\epsilon = 0.1$ dB. When comparing these schemes, remember also that the HOC-UEP scheme provides a BER significantly lower than the overall BER for $\mathcal{P}^1$. Fig. 4.8 shows the effect of different threshold offsets in the HOC-UEP scheme, with BER as a function of ϵ for $E_b/N_0 = 2.4$ dB. It is seen that $\epsilon = 0.1$ dB is a good choice for our example, since the BERs for both classes are lower than for code design without increased UEP ($\epsilon = 0$ dB). For higher values of ϵ, the first

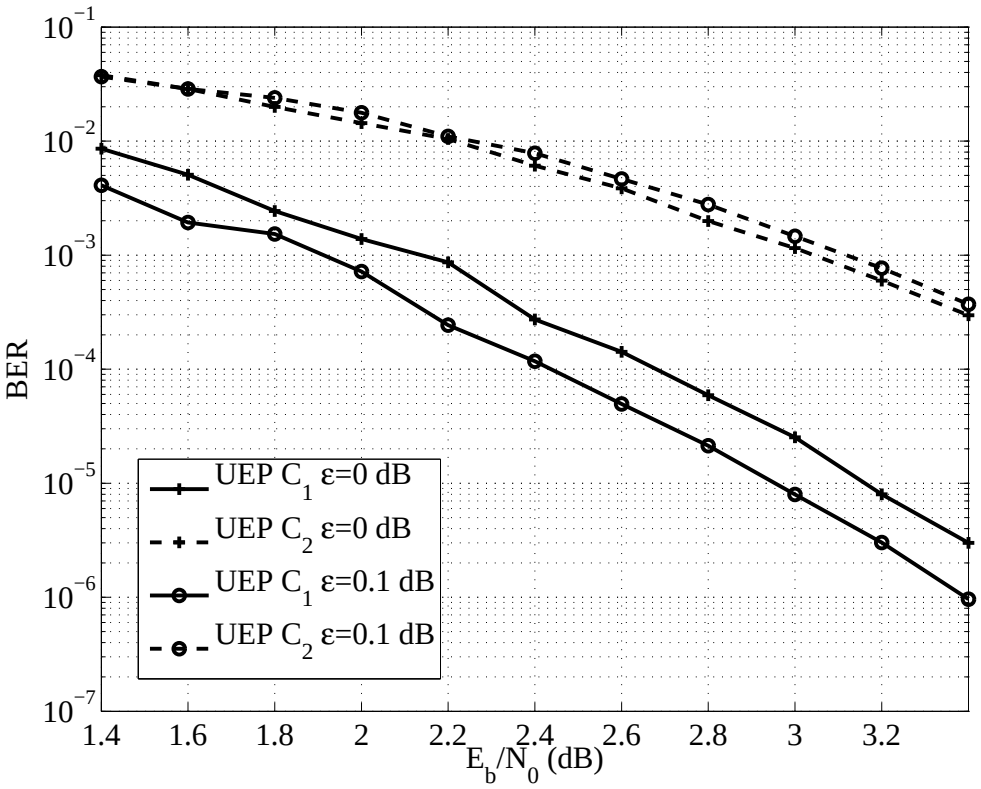


Figure 4.6: Bit-error rate performance of protection class $\mathcal{P}^1$ and $\mathcal{P}^2$ for the UEP scheme and 8-PSK.

class is assigned even more edges and the second and third classes have concentrated low degrees. With increasing ϵ, which implies increasing the code threshold, the global convergence of the code gets worse and this affects the performance of all classes.

In the limit where the number of decoding iterations approaches infinity, the UEP capability of the code itself theoretically should vanish. However, Fig. 4.9 shows that for at least up to a maximum of 200 iterations, the UEP capability is not affected much with increasing number of iterations. The reason for this behaviour will be discussed in Chapter 5.

4.4.2 Results for 64-QAM

In this section, we give an example of a UEP-LDPC code designed for and used with 64-QAM. We design a code with the parameters n, k, R, N_p, $\boldsymbol{\alpha}$, $\boldsymbol{\rho}$, and $d_{v_{max}}$ as in

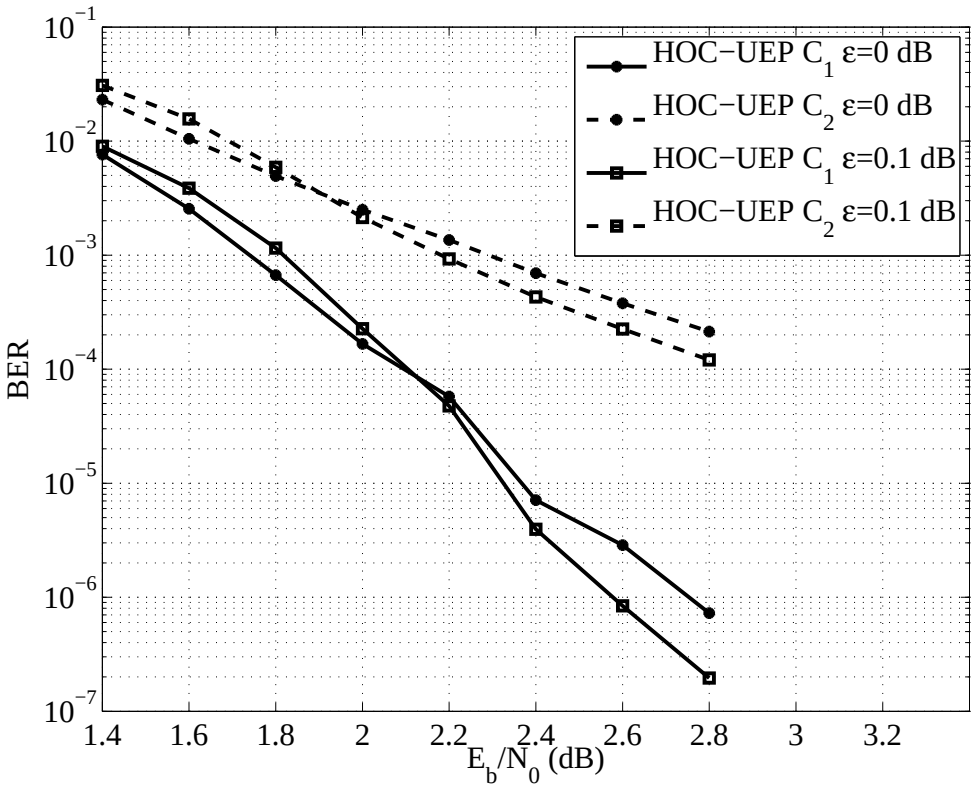


Figure 4.7: Bit-error rate performance of protection class $\mathcal{P}^1$ and $\mathcal{P}^2$ for the HOC-UEP scheme and 8-PSK.

the previous section. 64-QAM yields three modulation classes of equal size, and thus, $N_m = 3$ and $\boldsymbol{\beta} = [1/3,\ 1/3,\ 1/3]$. Fig. 4.10 shows the BER performance of protection classes $\mathcal{P}^1$ and $\mathcal{P}^2$ of the 64-QAM HOC-UEP code compared to a UEP code designed for BPSK, both with $\epsilon = 0.1$ dB. The plot shows that for low SNR, the BERs of the HOC-UEP scheme are worse but outperform the UEP scheme for an SNR higher than approximately 1.7 dB. For a BER of 10^{-5}, HOC-UEP $\mathcal{P}^1$ has an E_b/N_0 gain of almost 1.5 dB compared to UEP $\mathcal{P}^1$.

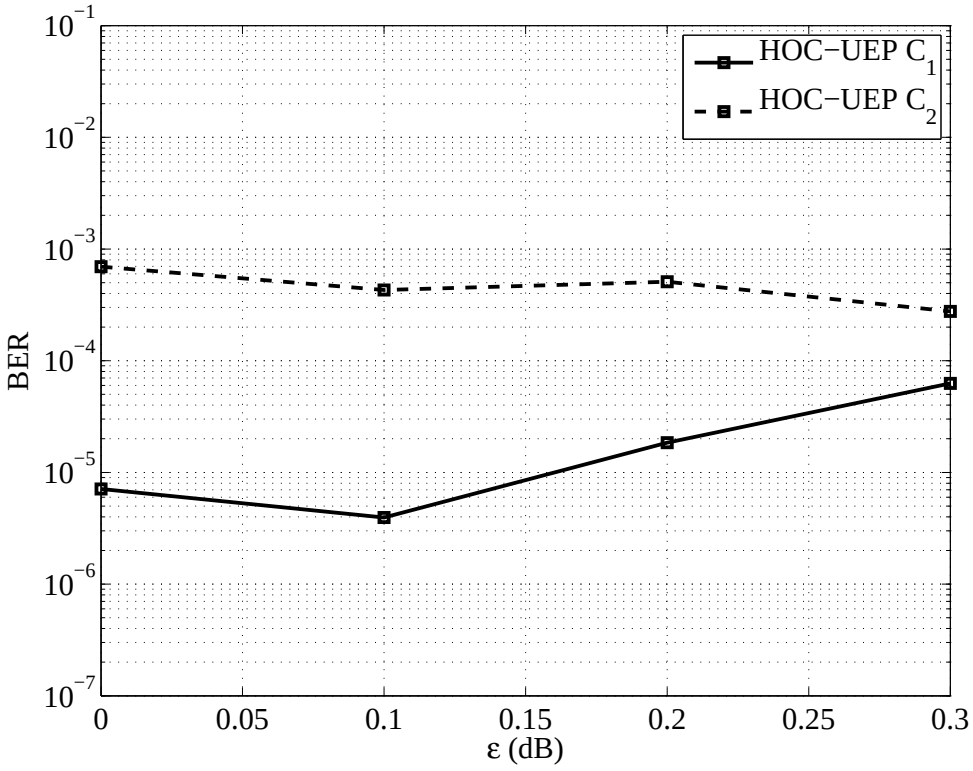


Figure 4.8: Performance of the HOC-UEP scheme for $E_b/N_0 = 2.4$ dB for various values of ϵ and 8-PSK.

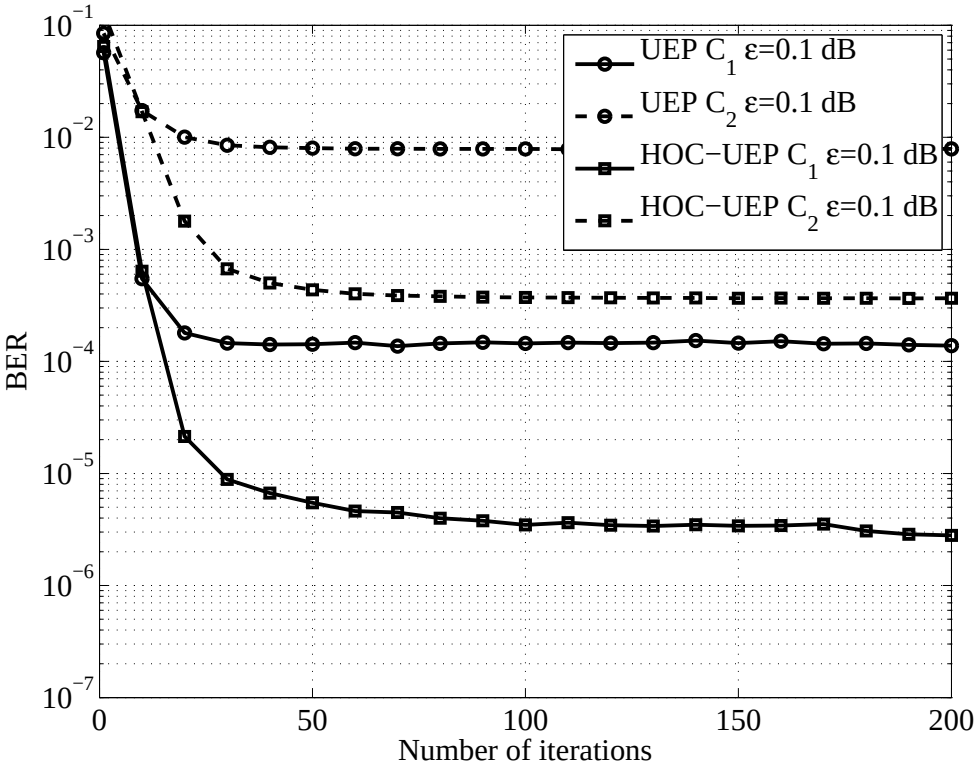


Figure 4.9: Bit-error rate performance as a function of the number of decoding iterations for $E_b/N_0 = 2.4$ dB and 8-PSK.

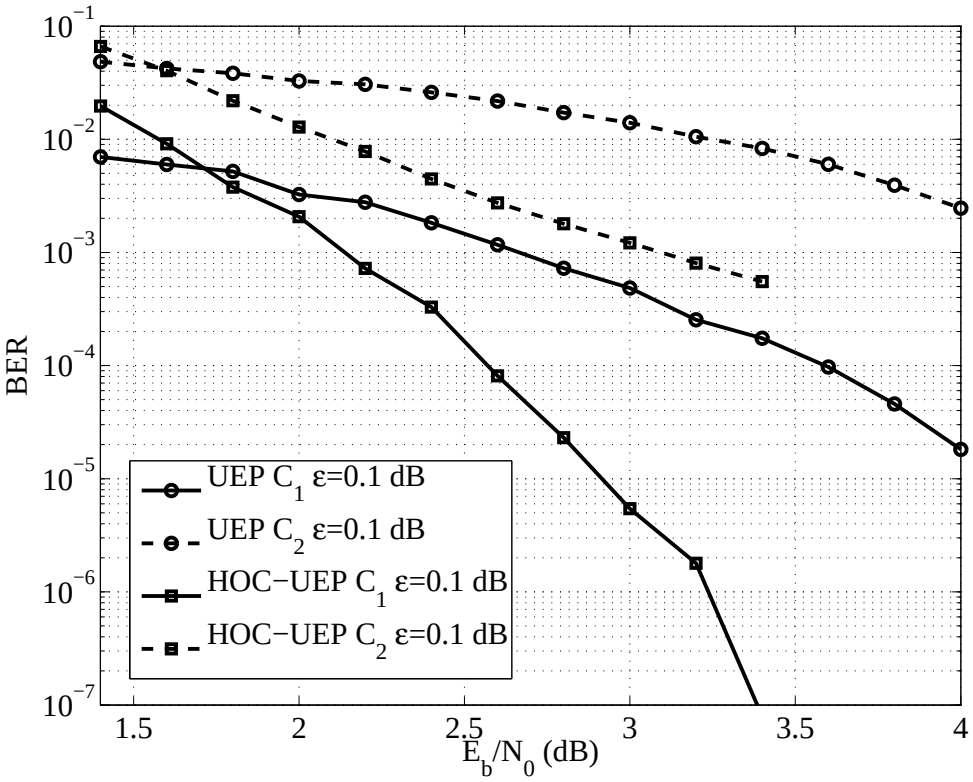


Figure 4.10: Bit-error rate performance of the UEP and HOC-UEP schemes for 64-QAM modulation.

Chapter 5

On the UEP Capabilities of Several LDPC Construction Algorithms

This chapter analyses the UEP capabilities of different well-known construction algorithms for the parity-check matrix of LDPC codes. We consider irregular UEP-LPDC codes with degree distributions designed by the algorithm described in [PDF07] or Chapter 4. Several publications apply different algorithms for obtaining irregular UEP-LDPC codes. However, their results disagree. For example, [RPNF07] shows significant UEP capabilities after 200 message-passing iterations, while [KM06] argues that no UEP gradation can be detected for irregular UEP-LDPC codes after 50 iterations. In this part of the thesis, the reasons behind the disagreeing results are explained by analysing different construction algorithms with respect to how the graph properties of the corresponding codes affect the UEP capabilities.

Once a degree distribution is obtained, a parity-check matrix $\mathbf{H}$ has to be constructed according to the degree profile. Many construction algorithms, such as the approximate cycle extrinsic message degree (ACE) algorithm [TJVW04], the progressive edge-growth (PEG) algorithm [HEA05], etc., have been developed. Random construction, following the approach of [RSU01], was typically used a few years ago. However, several authors have suggested construction algorithms with better BER performance, especially in the error-floor region, mainly by avoiding small cycles in the Tanner graph. We consider five different algorithms for the construction of the parity-check matrix. All of the

obtained codes belong to the same code ensemble, i.e., they have the same variable and check node degree distributions.

In this chapter we confirm by simulation that the design of an irregular variable node degree distribution provides UEP capability for a low number of message-passing iterations regardless of the construction algorithm used. However, the results also show that the choice of the construction algorithm is critical when good UEP properties are desired after a moderate or high number of iterations. UEP capability after many iterations is important since this enables considerably lower error rates than a low number of iterations. To ensure UEP capability of the code regardless of the choice of construction algorithm, the decoder must typically be interrupted after only 10 iterations, resulting in performance losses.

The chapter is organised as follows. Several construction algorithms are presented in Section 5.1. In Section 5.2, we shortly describe the ensemble design and discuss performance results of the algorithms and their differences. Section 5.3 discusses properties of parity-check matrices $\mathbf{H}$ which are relevant for the UEP behaviour. In Section 5.4, we modify one of the construction algorithms to enhance the UEP capability of the resulting code.

5.1 Construction Algorithms

In this section, we will describe the basic concepts of several well-known construction algorithms, such as random construction, the progressive edge-growth (PEG) algorithm, zigzag construction, the approximate cycle extrinsic message degree (ACE) algorithm, and an ACE-constrained PEG algorithm. For the sake of brevity, we only introduce the algorithms. For details, the reader is referred to the references given in the respective sections.

5.1.1 Random Construction

The random construction algorithm places the edges randomly in the graph, according to the given variable node and check node degree distributions. We consider a random construction where only length-4 cycles between degree-2 variable nodes are avoided. The resulting Tanner graph has girth 4.

5.1.2 Progressive Edge-Growth (PEG) Construction

The PEG algorithm is an efficient algorithm for the construction of parity-check matrices with large girth by progressively connecting variable nodes and check nodes [HEA05]. Starting with the lowest-degree variable nodes, edges are chosen according to an edge selection procedure. The selection procedure aims at minimising the impact of each new edge on the girth. This is done by expanding a tree from the current variable node down to a certain depth. If possible, the algorithm connects the variable node to a check node that is not yet reached at this depth of the tree. If all check nodes are reached, the check node that will create the largest cycle through the current variable node is selected. If several possible choices exist, a check node with the lowest check node degree under the current graph setting is chosen. Finally, if there is more than one possible lowest-degree check node left, one of them is chosen randomly. Thereby, the resulting Tanner graph automatically has a concentrated check node degree distribution. This is shown in [RSU01] to be advantageous for good convergence.

5.1.3 Zigzag Construction

The zigzag construction algorithm connects the edges of degree-two variable nodes in a zigzag manner, according to [HEA01]. In principle, the part of the parity-check matrix corresponding to the degree-two variable nodes will be bi-diagonal. This algorithm avoids all cycles involving only variable nodes of degree two, assuming that the number N_2 of degree-two variable nodes is smaller than the number of check nodes, i.e., $N_2 < N - K$. The remaining edges may be connected randomly in the same way as described for the random algorithm (zigzag-random) or according to the PEG algorithm (zigzag-PEG). The zigzag construction is of interest both due to the easy encoding, that is the result of the almost diagonal shape of the part of the parity-check matrix that corresponds to the parity bits, and the avoidance of cycles involving only degree-two variable nodes, which have been shown to degrade the BER performance, see e.g. [TJVW04].

5.1.4 Approximate Cycle Extrinsic Message Degree (ACE) Construction

For finite block lengths, Tian *et al.* [TJVW04] proposed an efficient graph conditioning algorithm called the approximate cycle extrinsic message degree (ACE) algorithm. The aim of the ACE construction is to lower the error floor by emphasising both the connectivity as well as the length of cycles. The ACE algorithm avoids small cycle clusters that are isolated from the rest of the graph and is shown to lower the error-floors of irregular LDPC codes significantly, while only slightly degrading the performance in the waterfall region. The approximate cycle extrinsic message degree of a length-$2d$ cycle is $\sum_i (d_i - 2)$, where d_i is the degree of the ith variable node in the cycle. An LDPC code has parameters (d_{ACE}, η) if all the cycles whose lengths are $2d_{ACE}$ or less have an ACE of at least η. The ACE algorithm is an efficient Viterbi-like algorithm with linear complexity proposed to detect and avoid harmful short cycles during code construction. Cycles with low ACE are considered to be harmful, since these cycles include variable nodes with low degrees and the cycles have low connectivity to nodes outside the cycle.

Given the variable node degree distribution $\lambda(x)$, columns of the parity-check matrix are generated one at a time starting from low-weight columns. The edges of every new column are generated randomly and the ACE algorithm checks whether the (d_{ACE}, η) requirement is met. If not, this column is generated again. This procedure is repeated until the whole parity check matrix is generated. In our implementation, we add extra constraints to construct a graph according to the given check node degree distribution $\rho(x)$.

The performance of highly irregular LDPC codes is of great interest when considering irregular UEP-LDPC codes, since the UEP properties are obtained by assigning variable nodes with high degree to the best protected class. However, with highly irregular codes it is difficult to ensure a high girth and the ACE algorithm is of importance.

5.1.5 PEG-ACE Construction

The PEG-ACE algorithm is a generalisation of the popular PEG algorithm, that is shown to generate good LDPC codes with short and moderate block lengths having large girth. In the PEG-ACE algorithm [VS08], the sequential methodology of the PEG algorithm is employed with a modified check node selection procedure. If the creation of

cycles cannot be avoided while adding an edge, the PEG-ACE construction algorithm chooses an edge that creates the longest possible cycle with the largest possible ACE value. The algorithm constructs a code with the largest possible ACE constraints, which is assumed to reduce the number of trapping sets significantly. As the PEG algorithm, the PEG-ACE construction algorithm usually picks a check node among its candidates which has the lowest check node degree. It has been shown that a PEG-ACE code performs better than an ACE code in the error-floor region [VS08].

5.2 Simulation Results

5.2.1 Ensemble Design

We consider the UEP-LDPC ensemble design proposed in [PDF07], which is based on a hierarchical optimisation of the variable node degree distribution for each protection class. The algorithm maximises the average variable node degree within one class at a time while guaranteeing a minimum variable node degree as high as possible. The optimisation can be stated as a linear programming problem and can, thus, be easily solved. To keep the overall performance of the UEP-LDPC code reasonably good, the search for UEP codes is limited to degree distributions whose convergence thresholds lie within a certain range ϵ of the minimum threshold of a code with the same parameters. We fix ϵ to 0.1 dB, which is shown in [PDF07] to give a good trade-off between the performances of the protection classes.

The UEP-LDPC ensemble design algorithm is initialised with a maximum variable node degree $d_{v_{max}}$, the code rate R, and a check node degree distribution. The bits of the codeword are divided into protection classes $\mathcal{P}^j$ according to their protection requirements. We design a UEP-LDPC code with $N_p = 3$ protection classes with rate $1/2$, $d_{v_{max}} = 30$, and $\rho(x) = 0.00749\,x^7 + 0.99101\,x^8 + 0.00150\,x^9$, which is found by numerical optimisation in [RSU01] to be a good check node degree distribution for $d_{v_{max}} = 30$. The proportions of the classes are chosen such that $\mathcal{P}^1$ contains 20% of the information bits and $\mathcal{P}^2$ contains 80%. The third class ($\mathcal{P}^3$) contains all parity bits. Therefore, we are mainly interested in the performances of classes $\mathcal{P}^1$ and $\mathcal{P}^2$. The resulting variable node degree distribution is defined by the coefficients $\lambda_i^{(\mathcal{P}^j)}$ which denote the fractions of edges incident to degree-i variable nodes of protection class $\mathcal{P}^j$. The overall degree distribution is therewith given by $\lambda(x) = \sum_{j=1}^{N_p} \sum_{i=2}^{d_{v_{max}}} \lambda_i^{(\mathcal{P}^j)} x^{i-1}$.

Table 5.1 summarizes the optimized variable node degree distribution for the resulting UEP-LDPC code.

Table 5.1: Variable node degree distribution of the UEP-LDPC ensemble.

$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\lambda_{18}^{(\mathcal{P}^1)} = 0.2521$	$\lambda_{3}^{(\mathcal{P}^2)} = 0.0786$	$\lambda_{2}^{(\mathcal{P}^3)} = 0.2130$
$\lambda_{19}^{(\mathcal{P}^1)} = 0.0965$	$\lambda_{4}^{(\mathcal{P}^2)} = 0.2511$	$\lambda_{3}^{(\mathcal{P}^3)} = 0.0141$
$\lambda_{30}^{(\mathcal{P}^1)} = 0.0946$		

5.2.2 Performance Comparison

UEP-LDPC codes with length $N = 4096$ are constructed using the construction algorithms described in Section 5.1. All codes belong to the ensemble described above. From each construction algorithm, we consider one code realisation in the following. It should be noted that the differences between several code realisations constructed with the same construction algorithm are small. We present simulation results for BPSK transmission over the AWGN channel. Figure 5.1 shows the FER and the BER as a function of E_b/N_0 for the random and ACE code after 100 decoder iterations. They both show good UEP properties, but the ACE code outperforms the random code for $\mathcal{P}^2$ and $\mathcal{P}^3$. We also see that the ACE code has a lower error-floor than the random code. Figure 5.2 shows the FER and the BER for the zigzag-random and PEG-ACE code after 100 decoder iterations. The zigzag-random code shows moderate UEP capabilities, while the PEG-ACE code does not show any UEP at all in FER and very little in BER. Simulation results for the PEG code and the zigzag-PEG code are omitted here since they show almost exactly the same performance and UEP capability as the PEG-ACE code. For simplicity, we will summarise the construction algorithms into the following two groups: non-UEP algorithms and UEP-capable algorithms. The non-UEP construction algorithms are the PEG, the zigzag-PEG, and the PEG-ACE construction. The UEP-capable construction algorithms are the random, the ACE, and the zigzag-random construction.

For standard code design, i.e. without UEP, the PEG-ACE construction has been shown to lower the error-floor while the loss in the waterfall-region is minimal [VS08].

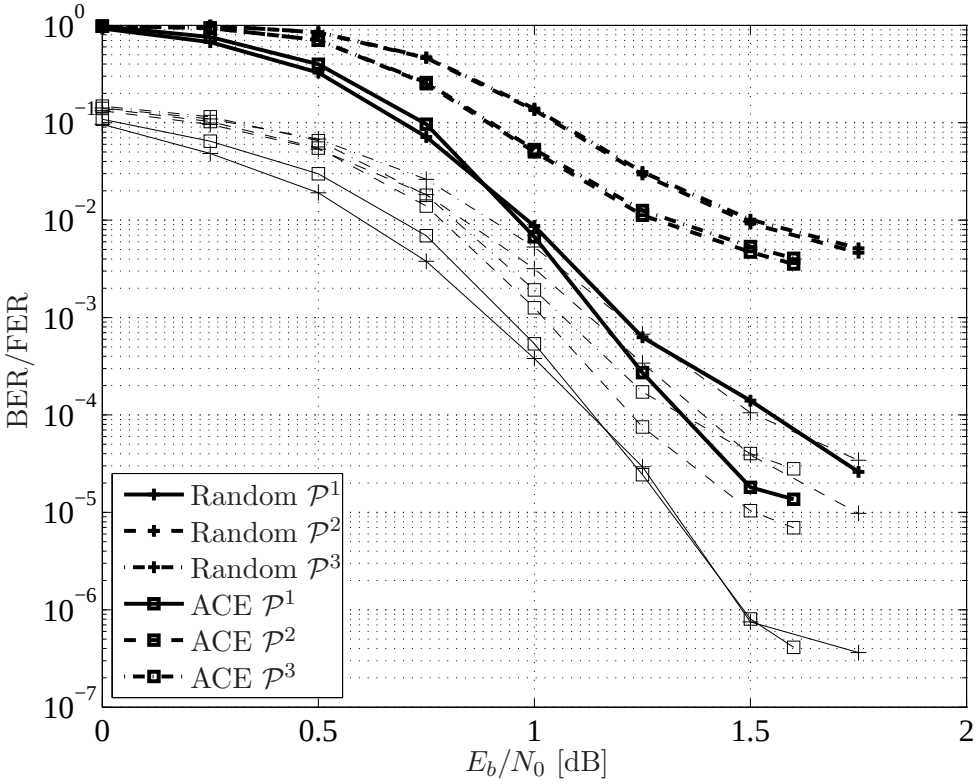


Figure 5.1: FER and BER of the random code and the ACE code as a function of E_b/N_0, after 100 iterations. The bold curves show FER and the thin curves show BER. Both codes show good UEP capabilities, but the ACE code outperforms the random code for $\mathcal{P}^2$ and $\mathcal{P}^3$.

The results in Fig. 5.2 show the same behaviour. Remarkably, the PEG-ACE code shows almost no difference in performance between the classes. The PEG-ACE construction does not lower the error-floors of all classes compared to the random construction as may be expected, but it removes the UEP capability by improving C^2 and $\mathcal{P}^3$ while degrading $\mathcal{P}^1$. Also, the loss in the waterfall-region is slightly higher than shown for the standard code design, while the gain in the error-floor region is substantial since all classes have low error floors.

Figure 5.3 shows the BER as a function of the number of decoder iterations at $E_b/N_0 = 1.25$dB for one UEP-capable code and one non-UEP code, that is, the ACE and the PEG-ACE code. The other algorithms in both groups show similar results, corresponding to the respective group. Both groups have similar performance for the first 10 iterations where there is a significant difference in BER between $\mathcal{P}^1$ and

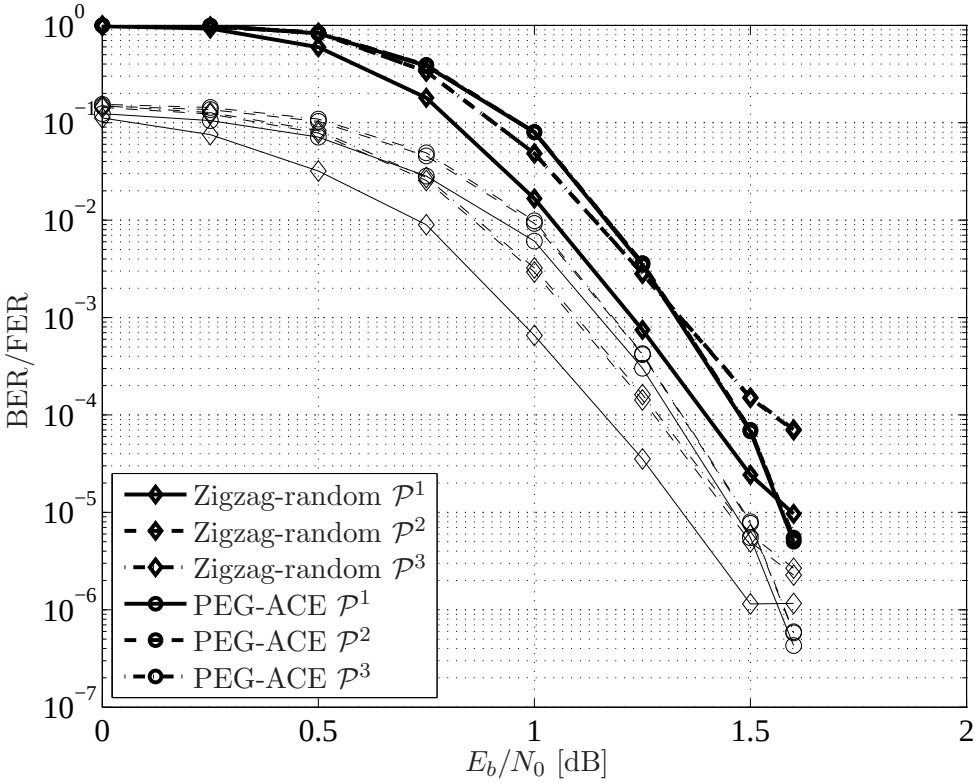


Figure 5.2: FER and BER of the zigzag-random code and the PEG-ACE code as a function of E_b/N_0, after 100 iterations. The bold curves show FER and the thin curves show BER. The zigzag-random code shows moderate UEP capabilities, while the PEG-ACE code does not show any UEP at all in FER and very little in BER.

$\mathcal{P}^2$. However, limiting the number of iterations to less than 10 would cause a huge performance loss. For a high number of iterations, it is seen that the PEG-ACE code has almost no UEP. For low E_b/N_0, all classes of the PEG-ACE code perform fairly bad compared to the other codes, see Fig. 5.2. However, in the error-floor region, $\mathcal{P}^2$ and $\mathcal{P}^3$ of the PEG-ACE code have much lower error rates than the other codes.

The results presented in this section suggest the use of the PEG-ACE code for high E_b/N_0. At $E_b/N_0 = 1.6$dB, all classes of PEG-ACE have the same performance as the best class of ACE. Good performance of all classes is of course even better than UEP capability with only good performance of the most protected class. However, for low E_b/N_0, the PEG-ACE code performs badly and the ACE code with UEP capability is a better choice.

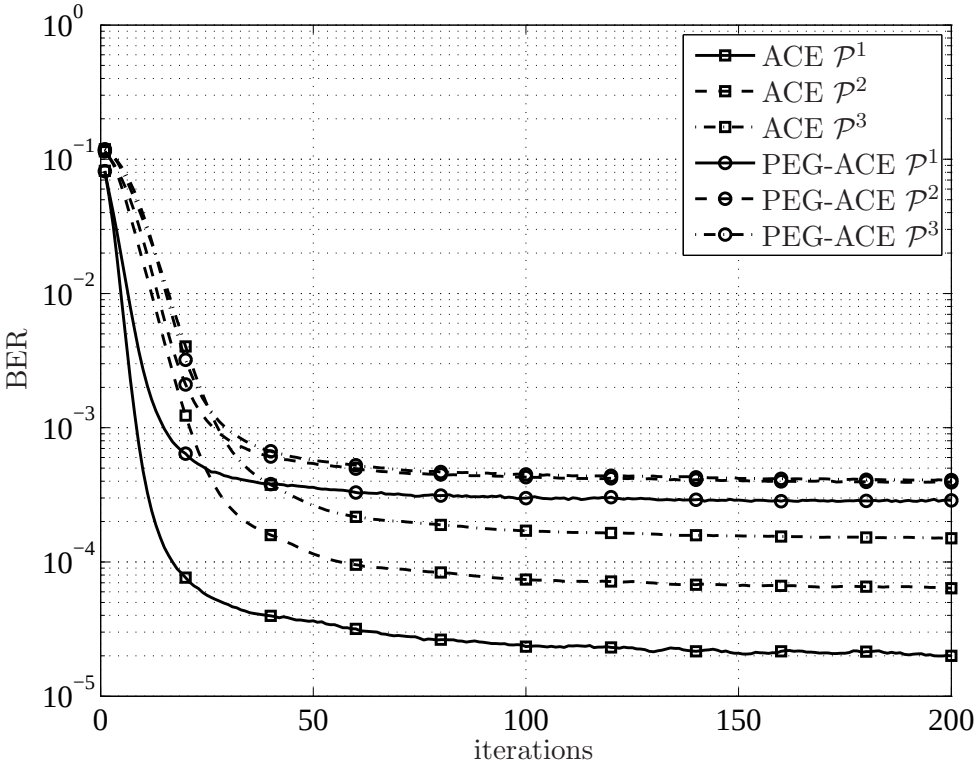


Figure 5.3: BER as a function of the number of decoder iterations for the ACE and the PEG-ACE code at $E_b/N_0 = 1.25$ dB. Both codes show good UEP properties after a small number of iterations (< 10), but after a large number of iterations all classes of the PEG-ACE code perform almost the same.

5.3 Relevant Graph Properties

In this section, we present properties of the Tanner graph which are relevant for the UEP behaviour of the code. These properties concern the connectivities between variable nodes of different protection classes. It should also be mentioned that the UEP capability is mainly independent of finite-length issues. We investigated cycles and trapping sets and found differences between the algorithms. However, later results will show that there is no direct connection between cycles and the UEP capability.

To compare the algorithms in terms of cycles we count the cycles encountered from each variable node and sum over all variable nodes in the class. This means that each cycle will be counted several times (2 times for cycles of length 4, etc.). Table 5.2 gives the number of cycles of lengths 4 and 6 that involve at least one variable node from

the respective class.

Table 5.2: The number of cycles of length 4 and 6 involving at least one variable node from the respective class.

	Length 4			Length 6		
	$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$	$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
Random	4 909	37	10	196 594	35 143	17 463
ACE	3 582	236	48	183 279	31 065	11 516
Zigzag-random	2 020	0	0	195 297	19 683	12 535
PEG	0	0	0	119 226	18 874	5 203
PEG-ACE	0	0	0	115 892	10 698	943

The table shows that the UEP-capable codes (random, ACE and zigzag-random) contain many cycles of length 4 in $\mathcal{P}^1$ and very few in the other classes. The non-UEP codes all have girth 6 and thus contain no cycles of length 4. Interestingly, the cycles in $\mathcal{P}^1$ do not seem to affect the performance of $\mathcal{P}^1$ much, since the BERs of $\mathcal{P}^1$ for the UEP-capable codes are still lower than for the non-UEP codes. This is in agreement with the observations in [TJVW04] that only isolated short cycles are harmful. Since the degrees of variable nodes in $\mathcal{P}^1$ are relatively high, the cycles in $\mathcal{P}^1$ are not isolated but have high ACE values. On the other hand, short cycles facilitate the existence of trapping sets in a graph. Trapping sets have been shown to govern the error-floor performance of LDPC codes [Ric03]. The number of cycles is therefore of importance for the error-floor performance.

5.3.1 Connectivity Between Protection Classes

Since the degree distributions $\lambda(x)$ and $\rho(x)$ are equal for all codes, we investigate how the variable nodes of different protection classes are connected through the check nodes. It is especially interesting to see how the incident variable nodes of a check node are spread between the classes. Let us consider a certain check node and how the edges of this check node are connected to the protection classes. We investigate if the edges are uniformly distributed to the protection classes or if a majority of edges is connected to a certain protection class.
Generally, a check node degree distribution may be defined from the node's perspective

as

$$\tilde{\rho}(x) = \sum_{i=2}^{d_{c_{max}}} \tilde{\rho}_i x^{i-1} \, .$$

The coefficients $\tilde{\rho}_i$ correspond to the fraction of degree-i check nodes. In order to account for connections to different protection classes, we define detailed check node degree distributions for the protection classes C^j,

$$\tilde{\rho}^{(\mathcal{P}^j)}(x) = \sum_{i=0}^{d_{c_{max}}} \tilde{\rho}_i^{(\mathcal{P}^j)} x^{i-1} \, , \quad i = 1 \ldots N_p \, . \tag{5.1}$$

The coefficients $\tilde{\rho}_i^{(\mathcal{P}^j)}$ correspond to the fraction of check nodes with i edges connected to class-$\mathcal{P}^j$ variables nodes. Note that i is not the overall degree of the check nodes but only the number of edges which are connected to class-$\mathcal{P}^j$ variable nodes. For example, $\tilde{\rho}_4^{(\mathcal{P}^1)}$ is the number of all check nodes with exactly 4 edges connected to $\mathcal{P}^1$, divided by $N-K$, where the remaining edges of these check nodes may be arbitrarily connected to the other classes. By definition we have $\sum_{i=0}^{d_{c_{max}}} \tilde{\rho}_i^{(\mathcal{P}^j)} = 1, \; j = 1, \ldots, N_p$. This detailed check node degree distribution is similar to the detailed representation described in [KSS03b], but we consider the degree distribution from the node's perspective while [KSS03b] considers the edge's perspective. The representation in [KSS03b] is also more detailed than necessary for our purpose since it defines connections to nodes of certain degrees instead of certain protection classes. Table 5.3 presents the coefficients of the detailed check node degree distributions for the ACE, the zigzag-random, and the PEG-ACE code. Note that the maximum check node degree of the whole code is $d_{c_{max}} = 10$.

Most of the coefficients of the ACE code are non-zero for all three protection classes, while the PEG-ACE code has only a few non-zero coefficients. That is, the PEG-ACE code only has a few different types of check nodes, and the numbers of connections to the protection classes are very similar for all nodes: The PEG-ACE coefficients $\tilde{\rho}_4^{(\mathcal{P}^1)}$, $\tilde{\rho}_3^{(\mathcal{P}^2)}$ and $\tilde{\rho}_2^{(\mathcal{P}^3)}$ are all large, which shows that most check nodes have 4 edges connected to $\mathcal{P}^1$, 3 edges to $\mathcal{P}^2$, and 2 edges to $\mathcal{P}^3$. This means that the variable nodes of a protection class are generally well connected to other protection classes through the check nodes. In contrast, the ACE code exhibits many different kinds of check nodes. Some of the check nodes are mainly connected to one protection class, having only one or two edges going to other protection classes. There even exist check nodes having 10 edges to a protection class, which means that they are solely connected to this class. This property seems to be important for the capability of providing UEP.

With more non-zero coefficients the classes are more isolated and the propagation of messages between the classes will be slower. If most check nodes have several edges to all protection classes, reliable and unreliable messages from different classes may proceed to other classes more easily and affect their performance.

The detailed check node degree distributions of the zigzag-random code have few non-zero coefficients for $\mathcal{P}^3$, while the distributions for $\mathcal{P}^1$ and $\mathcal{P}^2$ are similar to the ACE code. Many check nodes have two edges connected to $\mathcal{P}^3$, while the connectivities to the other classes vary. The connectivity between the classes is therefore higher than for the ACE code, but lower than for the PEG-ACE code. This agrees with the UEP capabilities, since the zigzag-random code shows less UEP than the ACE code and more than the PEG-ACE code. The detailed distributions of the other codes are omitted here since the distribution of the random code is very similar to that of the ACE code and all non-UEP codes have almost the same detailed check node degree distributions. In order to visualise the observations, Fig. 5.4 shows the detailed check node degree distributions of the ACE, the zigzag-random, and the PEG-ACE code.

Table 5.3: Detailed check node degree distributions.

	ACE			Zigzag-random			PEG-ACE		
	$\mathcal{P}^1$	$\mathcal{P}^2$	C^3	$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$	$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\tilde{\rho}_0^{(\mathcal{P}^j)}$	0.04102	0.02588	0	0.04248	0.06152	0	0	0	0
$\tilde{\rho}_1^{(\mathcal{P}^j)}$	0.05273	0.12158	0.48193	0.05664	0.14258	0.04199	0	0.00537	0
$\tilde{\rho}_2^{(\mathcal{P}^j)}$	0.10742	0.23584	0.24268	0.10889	0.23193	0.87842	0.00049	0.12109	0.95752
$\tilde{\rho}_3^{(\mathcal{P}^j)}$	0.14795	0.27686	0.14648	0.14746	0.22363	0.07568	0.10498	0.77832	0.04248
$\tilde{\rho}_4^{(\mathcal{P}^j)}$	0.23926	0.21289	0.06445	0.21240	0.14258	0.00342	0.79980	0.09473	0
$\tilde{\rho}_5^{(\mathcal{P}^j)}$	0.21094	0.08594	0.03516	0.23145	0.10693	0.00049	0.09424	0.00049	0
$\tilde{\rho}_6^{(\mathcal{P}^j)}$	0.14453	0.03027	0.01611	0.14258	0.05664	0	0.00049	0	0
$\tilde{\rho}_7^{(\mathcal{P}^j)}$	0.05029	0.00879	0.00537	0.05566	0.03174	0	0	0	0
$\tilde{\rho}_8^{(\mathcal{P}^j)}$	0.00586	0.00195	0.00391	0.00244	0.00244	0	0	0	0
$\tilde{\rho}_9^{(\mathcal{P}^j)}$	0	0	0.00293	0	0	0	0	0	0
$\tilde{\rho}_{10}^{(\mathcal{P}^j)}$	0	0	0.00098	0	0	0	0	0	0

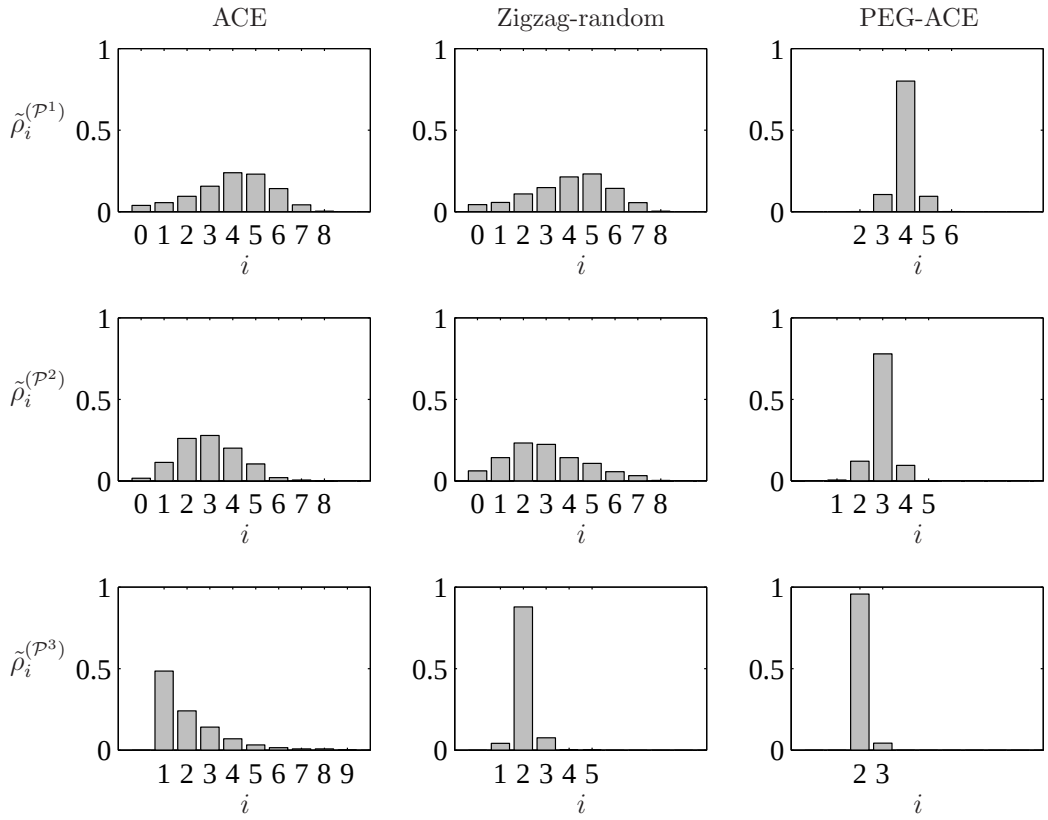


Figure 5.4: Detailed check node degree distributions of the ACE, the zigzag-random, and the PEG-ACE code. The UEP-capable ACE code has many non-zero coefficients, while the non-UEP PEG-ACE code has few non-zero coefficients and thereby better connectivity between the classes.

5.3.2 Detailed Mutual Information Evolution

The effect of the connectivity between the classes on the performance may be analysed by calculating the mutual information (MI) functions of the different codes. By calculating the theoretical MI functions, the effect of connectivity is isolated from effects due to cycles, trapping sets, and codeword length, since the calculation is based on the corresponding cycle-free graph. Typically, the MI functions are calculated from the degree distributions $\lambda(x)$ and $\rho(x)$ of a code. However, in our case all codes have the same overall degree distributions $\lambda(x)$ and $\rho(x)$. To observe the differences between the

algorithms, a detailed computation of MI may be performed by considering the edge-based MI messages traversing the graph instead of node-based averages. This has been done for protographs in [LC07]. We follow the same approach, but use the parity-check matrix instead of the protograph base matrix. See [tB01b, tBKA04] for more details on MI analysis.

Let I_{Av} be the *a priori* MI between one input message and the codeword bit associated to the variable node. I_{Ev} is the extrinsic MI between one output message and the codeword bit. Similarly on the check node side, we define I_{Ac} (I_{Ec}) to be the *a priori* (extrinsic) MI between one check node input (output) message and the codeword bit corresponding to the variable node providing (receiving) the message. The evolution is initialised by the MI between one received message and the corresponding codeword bit, denoted by I_{ch}, which corresponds to the channel capacity. For the AWGN channel, it is given by $I_{ch} = J(\sigma_{ch})$, where

$$\sigma_{ch}^2 = 8R\frac{E_b}{N_0} \tag{5.2}$$

and E_b/N_0 is the signal-to-noise ratio at which the analysis is performed. The function $J(\cdot)$ is defined by

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\sigma^2/2)^2}{2\sigma^2}} \log_2(1+e^{-y})dy \tag{5.3}$$

and computes the MI based on the noise variance. For a variable node with degree d_v, the extrinsic MI between the sth output message and the corresponding codeword bit is [LC07]

$$I_{Ev|s} = J\left(\sqrt{\sum_{l=1,l\neq s}^{d_v} [J^{-1}(I_{Av|l})]^2 + [J^{-1}(I_{ch})]^2}\right), \tag{5.4}$$

where $I_{Av|l}$ is the *a priori* MI of the message received by the variable node on its lth edge. The extrinsic MI for a check node with degree d_c may be written as

$$I_{Ec|s} = 1 - J\left(\sqrt{\sum_{l=1,l\neq s}^{d_c} [J^{-1}(1-I_{Ac|l})]^2}\right), \tag{5.5}$$

where $I_{Ac|l}$ is the *a priori* MI of the message received by the check node on its lth edge. Note that the MI functions are subject to the Gaussian approximation (see [CRU01]) and are not exact.

The following algorithm describes the MI analysis of a given parity-check matrix. We denote element (i, j) of the parity-check matrix by $h_{i,j}$.

1) Initialisation
   $I_{ch} = J(\sigma_{ch})$

2) Variable to check update

   a) For $i = 1, \ldots, N-K$ and $j = 1, \ldots, N$, if $h_{i,j} = 1$, calculate

$$I_{Ev}(i,j) = J\left(\sqrt{\sum_{s \in \mathcal{C}_i, s \neq i} [J^{-1}(I_{Av}(s,j))]^2 + (J^{-1}(I_{ch}))^2}\right) , \tag{5.6}$$

      where $\mathcal{C}_i$ is the set of check nodes incident to variable node i.

   b) If $h_{i,j} = 0$, $I_{Ev}(i,j) = 0$.

   c) For $i = 1, \ldots, N-K$ and $j = 1, \ldots, N$, set $I_{Ac}(i,j) = I_{Ev}(i,j)$.

3) Check to variable update

   a) For $i = 1, \ldots, N-K$ and $j = 1, \ldots, N$, if $h_{i,j} = 1$, calculate

$$I_{Ec}(i,j) = 1 - J\left(\sqrt{\sum_{s \in \mathcal{V}_j, s \neq j} [J^{-1}(1 - I_{Ac}(i,s))]^2}\right) , \tag{5.7}$$

      where $\mathcal{V}_j$ is the set of variable nodes incident to check node j.

   b) If $h_{i,j} = 0$, $I_{Ec}(i,j) = 0$.

   c) For $i = 1, \ldots, N-K$ and $j = 1, \ldots, N$, set $I_{Av}(i,j) = I_{Ec}(i,j)$.

4) *A posteriori* check node MI
   For $i = 1, \ldots, N-K$, calculate

$$I_{APPc}(i) = 1 - J\left(\sqrt{\sum_{s \in \mathcal{V}_j} [J^{-1}(1 - I_{Ac}(i,s))]^2}\right) . \tag{5.8}$$

5) *A posteriori* variable node MI
   For $j = 1, \ldots, N$, calculate

$$I_{APPv}(j) = J\left(\sqrt{\sum_{s \in \mathcal{C}_j} [J^{-1}(I_{Av}(s,j))]^2 + [J^{-1}(I_{ch})]^2}\right) . \tag{5.9}$$

6) Repeat 2)-5) until $I_{APPv} = 1$ for $j = 1, \ldots, N$.

The *a posteriori* MI of the check nodes I_{APPc} at $E_b/N_0 = 0.7$dB is shown in Fig. 5.5 for the ACE and PEG-ACE code. An average I_{APPc} of each class is calculated as an average I_{APPc} of all edges incident to variable nodes of the corresponding class. The figure shows that the average I_{APPc} of all classes are almost equal for the PEG-ACE code, while they differ for the ACE code. This behaviour may be explained by (5.8) and the connections of the check nodes to the different protection classes described by the detailed check node degree distributions in Table 5.3.

For the PEG-ACE code, almost all check nodes are connected to 4 variable nodes from $\mathcal{P}^1$, 3 variable nodes from $\mathcal{P}^2$ and 2 variable nodes from $\mathcal{P}^3$. Even if the extrinsic information from the variable nodes ($I_{Ev} = I_{Ac}$) differs (due to the irregular variable node degree distribution), almost all check nodes combine the same number of nodes from each class. This gives almost equal I_{Ec} for all check nodes according to (5.8), which means that the performance of different classes will be averaged over the whole codeword in this step.

For the ACE code on the other hand, the number of variable nodes from different classes that are connected to a check node differs much more. This allows for having some check nodes with higher MI than others and the difference in MI of the check nodes will increase the UEP capability of the code.

Remember that for all codes we consider here, there will be some UEP capability strictly depending on the irregularity of the LDPC code. In the first iteration, all I_{Ec} will be almost equal (due to the concentrated check node degree distribution) and the only difference in I_{APPv} between the classes will depend on the variable node degrees. Figure 5.6 shows the variable node *a posteriori* MI I_{APPv} over the number of iterations. It is shown in [tB01b] that small differences in MI may lead to significant differences in BER for MI values near to 1. Therefore, the figure shows the distance of the MI to its maximum value, i.e. $1 - I_{APPv}$, on a logarithmic scale.

The figure shows that the convergence speeds of the protection classes are farther apart for the ACE algorithm. Protection class $\mathcal{P}^1$ of the ACE code converges faster than that of the PEG-ACE code, while the other classes take more iterations to converge for the ACE code than for the PEG-ACE code. Theoretically, the MI may approach 1 arbitrarily close and there will still be UEP. The actual amount of UEP depends on the convergence speeds of the classes, [PDF07]. In practice, the accuracy is limited by the numerical precision of the computations and, for example, approximations of the J-function. However, using the approximation of the error probability based on mutual

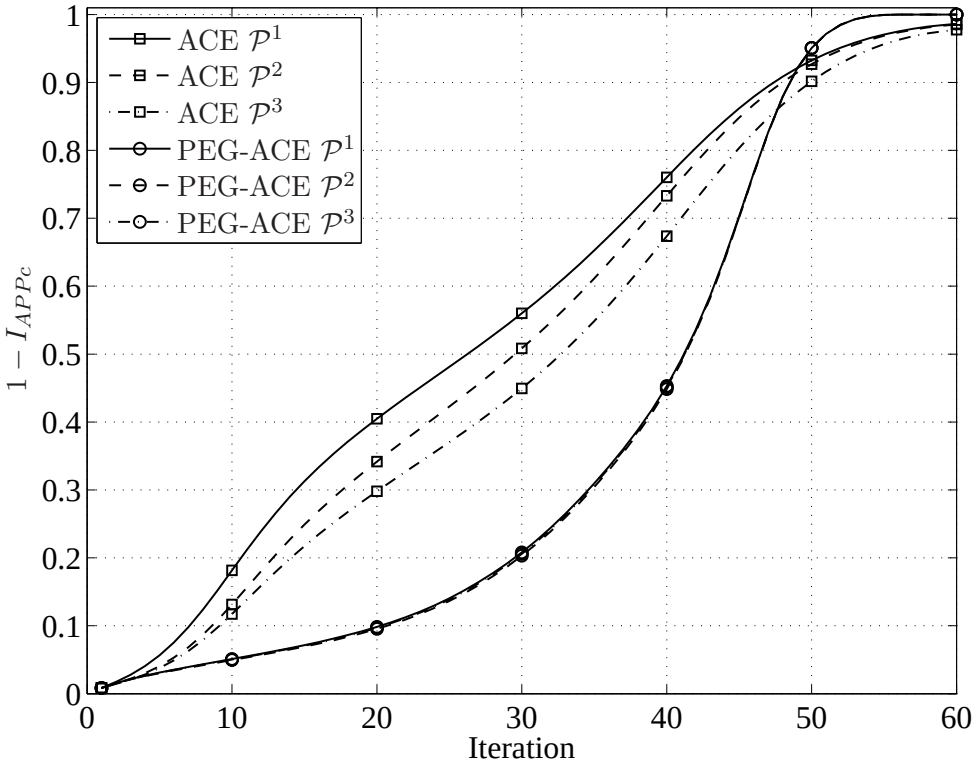


Figure 5.5: Check node *a posteriori* MI as a function of the number of decoder iterations at $E_b/N_0 = 0.7$dB. The average I_{APPc} of all classes are almost equal for the PEG-ACE code, while they differ for the ACE code.

information from [tB01b]

$$P_b \approx \frac{1}{2}\,\mathrm{erfc}\left(\frac{\sqrt{8R\frac{E_b}{N_0} + J^{-1}(I_{Av})^2 + J^{-1}(I_{Ev})^2}}{2\sqrt{2}}\right)\,, \tag{5.10}$$

the error probabilities of the protection classes may be estimated. Note that the results are not exact for finite-length codes and become more inexact with lower error rates. Nevertheless, we observe significant differences in BER between the classes if their MI values are close to 1 and only differ in the fifth decimal position.

The above discussion shows that the connectivity of the protection classes, defined by the detailed check node degree distribution in Table 5.3 plays an important role to the UEP capability. However, in comparing the theoretical MI from above with the true values obtained by measuring the actual decoding LLRs, we observe deviations. The

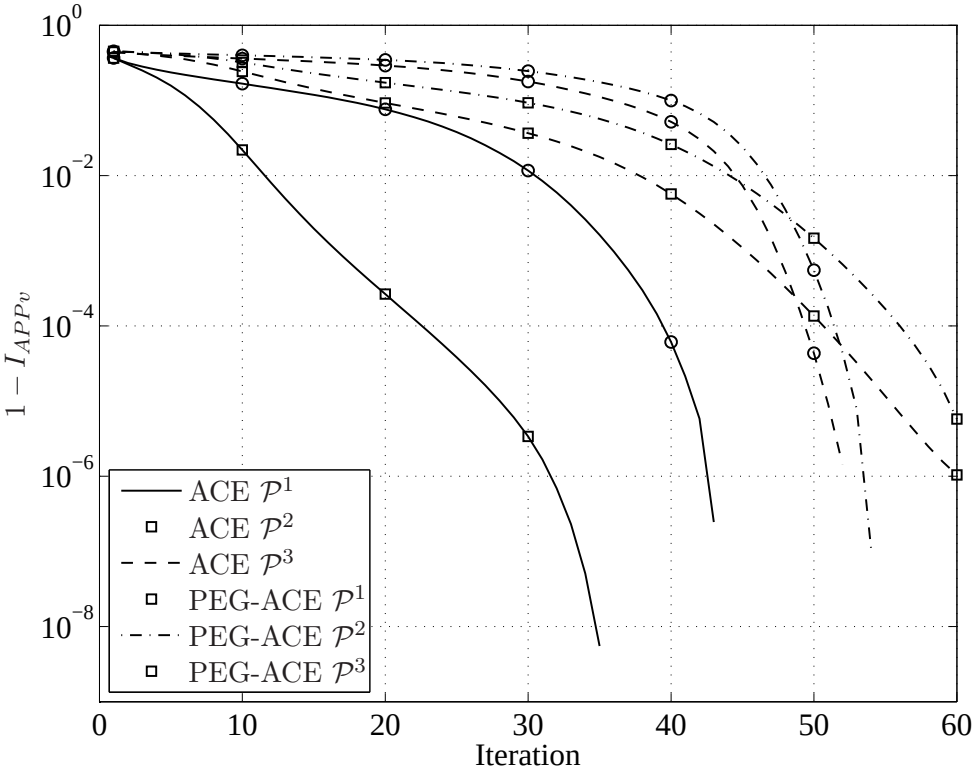


Figure 5.6: Distance of variable node *a posteriori* MI to the maximum MI, as a function of the number of decoder iterations, at $E_b/N_0 = 0.7$dB. Protection class $\mathcal{P}^1$ of the ACE code converges faster than that of the PEG-ACE code, while the other classes take more iterations to converge for the ACE code than for the PEG-ACE code.

measured MI values are lower than the theoretical ones which is due to finite-length issues such as cycles and trapping sets. Nevertheless, the UEP properties are valid for both theoretical and measured observations.

5.4 Modified PEG-ACE Construction with Increased UEP Capability

The above sections discussed the connectivity of check nodes to the protection classes and differences between the UEP codes and the non-UEP codes were found. In order to verify that the presented argument indeed is the reason for the differences in UEP capability, we modify the (non-UEP) PEG-ACE construction algorithm to yield codes

Table 5.4: Detailed check node degree distribution of the modified PEG-ACE code.

	Modified PEG-ACE		
	$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\tilde{\rho}_0^{(\mathcal{P}^j)}$	0	0.03711	0
$\tilde{\rho}_1^{(\mathcal{P}^j)}$	0.03662	0.15137	0.48291
$\tilde{\rho}_2^{(\mathcal{P}^j)}$	0.17529	0.23340	0.22510
$\tilde{\rho}_3^{(\mathcal{P}^j)}$	0.16211	0.25098	0.14648
$\tilde{\rho}_4^{(\mathcal{P}^j)}$	0.24707	0.16650	0.05908
$\tilde{\rho}_5^{(\mathcal{P}^j)}$	0.20020	0.07666	0.08496
$\tilde{\rho}_6^{(\mathcal{P}^j)}$	0.12939	0.05078	0.00146
$\tilde{\rho}_7^{(\mathcal{P}^j)}$	0.04443	0.02686	0
$\tilde{\rho}_8^{(\mathcal{P}^j)}$	0.00488	0.00635	0
$\tilde{\rho}_9^{(\mathcal{P}^j)}$	0	0	0
$\tilde{\rho}_{10}^{(\mathcal{P}^j)}$	0	0	0

with a similar detailed check node degree distribution as the UEP-capable ACE code.

The algorithm is modified in such a way that it only allows check nodes for the candidate list which do not violate certain detailed check node degree distributions $\tilde{\rho}^{(\mathcal{P}^j)}(x)$. Thereby, the parity-check matrix is forced to have detailed check node profiles similar to the ACE code instead of its natural distribution. By doing so, the detailed check node degree distribution given in Table 5.4 is obtained. Notice that this is very similar to the detailed distributions of the ACE code given in Table 5.3.

Fig. 5.7 shows the BER and FER of the modified PEG-ACE code in comparison to the original PEG-ACE code. It shows that by changing the detailed check node degree distribution of the original PEG-ACE algorithm, it is possible to enhance its UEP capability significantly. Instead of equal error rates for all classes, the modification has improved the BER of $\mathcal{P}^1$, while slightly degrading $\mathcal{P}^2$ and $\mathcal{P}^3$. Compared to the ACE algorithm, the modified PEG-ACE even shows more UEP capability, since the performance of $\mathcal{P}^1$ is better for the PEG-ACE code while $\mathcal{P}^2$ and $\mathcal{P}^3$ perform worse. Furthermore, it should be mentioned that the modified PEG-ACE code is even capable of increasing the differences in BER between $\mathcal{P}^2$ and $\mathcal{P}^3$ compared to the UEP-capable codes presented in Section 5.1.

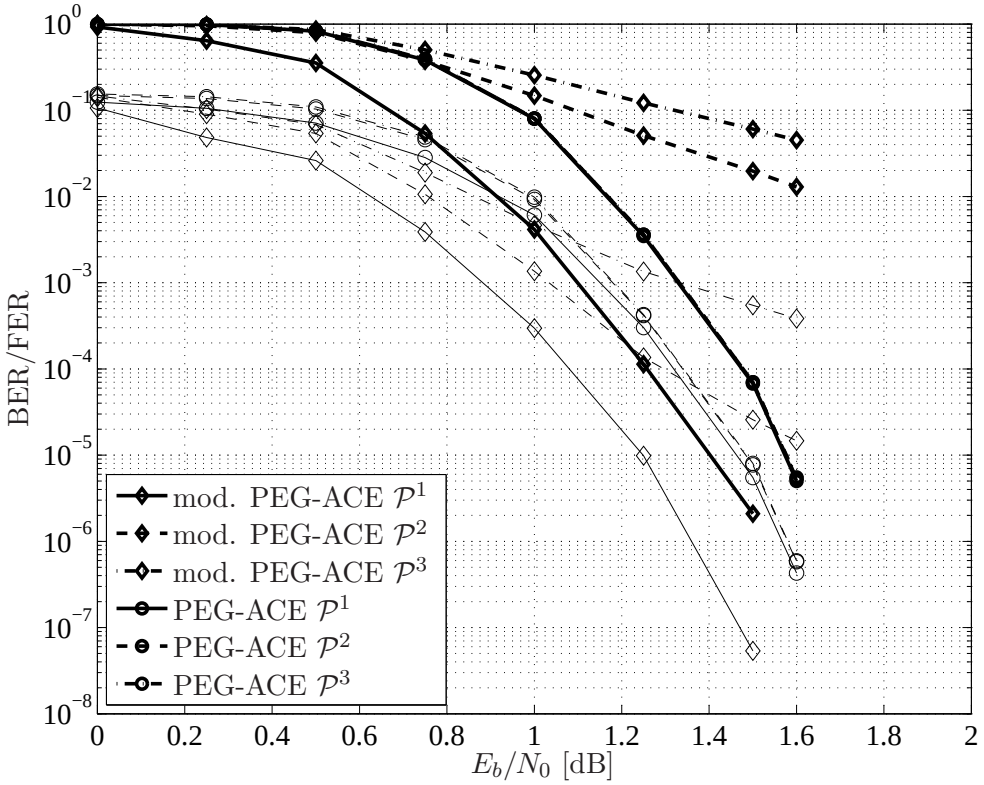


Figure 5.7: BER and FER of the modified PEG-ACE code in comparison to the original PEG-ACE code. The modification of the PEG-ACE code has increased its UEP capability significantly.

These results are meant to further support the above discussion that the differences in UEP capability between different LDPC construction algorithms are due to the connectivity between the classes, that is, the detailed check node degree distribution.

The modified PEG-ACE code has girth 6. Counting the cycles as in Section 5.3 yields 177 122 cycles involving at least one variable node from $\mathcal{P}^1$, 8 931 cycles in $\mathcal{P}^2$ and 838 cycles in $\mathcal{P}^3$. The numbers of cycles in the modified PEG-ACE graph are similar to the number of cycles in the PEG-ACE graph. Still the two algorithms have totally different UEP behaviour. These results suggest that cycles do not affect the UEP capability.

With the modified PEG-ACE algorithm, it is possible to enhance the UEP capability of a code by modifying its detailed check node degree distribution. Optimisation of the detailed check node degree distribution is outside the scope of this work. However, the detailed check node degree distribution may be used as a tool to move between codes

with good UEP capability but lower overall performance and codes with good overall performance but less UEP capability.

Chapter 6

Conclusions and Outlook

In this thesis, several approaches for obtaining unequal error protection with modern coding schemes have been investigated and developed. We have improved existing techniques as well as developed new approaches in order to enable efficient transmission of mutlimedia data.

In Chapter 2, we introduced an efficient and flexible construction method for unequal error protection convolutional codes and Turbo codes. Instead of puncturing code bits, information bits are fixed. Besides its flexibility in terms of rate compatibility and rates, it may increase the free distance of the mother code if designed properly. Furthermore, we propse an optimisation strategy for maximising the free distance of the pruned code for a given code rate, or maximising the code rate for a given free distance. The appendix contains tables of good pruning patterns as results of an exhaustive computer search.

Further research in this area might be the investigation of combined pruning and puncturing and the effects on the distance properties of a code. Furthermore, the exact impact of pruning on the mutual information may be investigated, especially the evolution of the extrinsic information due to increased a-priori information.

In Chapter 3, we designed an unequal error protection multilevel coding scheme where the waterfall regions of the different protection levels can directly be chosen by trading the code rates of the partitioning levels. By varying the code rates on the levels, it is possible to generate controllable, individual operating points on each level. The application of non-uniform signal constellations further improves the flexibility of the

design compared to standard signal constellations. When dealing with progressively encoded source files, truncating least important information bits makes the approach even more flexible, which is a common approach in the context of scalable data processing. Therewith, we have designed a flexible and very easy-to-control UEP-MLC scheme.

This topic still offers interesting research regarding the optimisation of the individual operating points. Depending on the source encoding algorithm, the performance may vary significantly and the individual operating points should be traded carefully. The results obtained in this thesis were based on intuition. However, there might be ways to optimise the strategy based on more detailed information or requirements from higher protocol layers. Also, non-uniform signal constellations and the code rate design could be jointly optimised.

Chapter 4 presents a flexible design method for UEP-LDPC codes with higher order constellations which is applicable to arbitrary signal constellations and an arbitrary number and proportions of the protection classes. For an example with 8-PSK, it is shown that the overall BER is reduced by the proposed method and there is a gain of 0.7 dB at BER 10^{-4} compared to using the UEP-LDPC codes designed for BPSK. An example with 64-QAM shows an even higher gain. The results show that the UEP capability is increased compared to standard UEP-LDPC codes. The proposed code design is also compared to a design method for higher order constellations without UEP and it is shown that the two methods have similar overall BER even though the presented scheme provides a protection class with BER significantly lower than the overall BER.

One open problem regarding this topic is the control of UEP by sacrifycing overall performance through the distance of the design threshold to the minimum threshold for given code parameters, also called threshold offset. Clearly, there is a trade-off between the performance and the threshold offset. On the one hand, if the offset is chosen too small, the amount of UEP may not be sufficient. On the other hand, if the offset is chosen too large, the overall performance of the code will decrease which may not be desirable either. An analysis on the influence which the offset has on the results is still open. This problem is connected to optimisatoin theory and requires detailed information about the shape of the search space.

In Chapter 5, we discussed the UEP properties of parity-check matrices constructed with different existing algorithms, i.e., the random, the ACE, the PEG, the Zigzag, and

the PEG-ACE construction algorithm. It turned out that these algorithms produce matrices with dramatical differences in capabilities of providing unequal error protection. We show that the connectivities between variable nodes from different protection classes is crucial to the UEP performance. The relevant properties of the parity-check matrix can be analysed by a detailed check node degree distribtion which confirms the theory and helps designing an improved algorithm with better UEP properties. The observations in this chapter are not only relevant for the analysis and improvement of UEP properties of a code but may be generally interesting when considering behavioural differences between several construction algorithms.

In general, the research on UEP still offers many open issues. One of the main problems is, however, not the design of a specific system, but the interaction between the physical layer and higher protocol layers. The problem of how to transfrom perceptive quality requirements into well-defined features of a communication system has not been solved, yet. Up to now, it is not clear how the relations and requirements of different protection classes should be chosen. Results so far have usually been based on intuitive guesses rather than well-defined theory.

Appendix A

List of Good Pruning Patterns for Convolutional Codes

This appendix contains lists of good codes with constraint lengths $L_c = 3$, $L_c = 4$, and $L_c = 5$, and rate-1/2 convolutional mother code. Given are the code rates R_{CC} and R_{TC} of the corresponding convolutional and Turbo code, the pruning pattern, the SNR (dB) where the Turbo codes converge, the offset δ from the Shannon limit (dB), the area between the EXIT curves, and the free distance of the convolutional code $d_{min,CC}$.

Table A.1: Generator matrix $\mathbf{G}(D) = \left(1 \;\; D^2/(1+D+D^2)\right)$, $L_c = 3$.

R_{CC}	R_{TC}	pruning pattern	E_s/N_0 [dB]	δ [dB]	area	$d_{free,CC}$
0.5	0.333		-3.73	1.35	0.25	4
0.417	0.278	$\begin{pmatrix} 0 & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-4.24	1.84	0.255	4
0.4	0.267	$\begin{pmatrix} 0 & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-4.27	2.05	0.263	4
0.333	0.222	$\begin{pmatrix} 0 & 0 & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-4.76	3.12	0.258	4
0.2	0.133	$\begin{pmatrix} 0 & 0 & 0 & \cdot & \cdot \end{pmatrix}$	-8.53	1.47	0.257	6
0.167	0.111	$\begin{pmatrix} 0 & 0 & 0 & \cdot & 0 & \cdot \end{pmatrix}$	-8.78	2.14	0.266	6

Table A.2: Generator matrix $\mathbf{G}(D) = \left(1 \;\; (D + D^3)/(1 + D + D^2)\right)$, $L_c = 4$.

R_{CC}	R_{TC}	pruning pattern	E_s/N_0 [dB]	δ [dB]	area	$d_{free,CC}$
0.5	0.333		-4.73	0.35	0.077	5
0.417	0.278	$\begin{pmatrix} 0 & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-6.04	0.05	0.168	5
0.4	0.267	$\begin{pmatrix} 0 & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-5.87	0.46	0.123	5
0.333	0.222	$\begin{pmatrix} 0 & 0 & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-7.16	0.767	0.219	5
0.3	0.2	$\begin{pmatrix} 0 & 0 & \cdot & \cdot & \cdot \end{pmatrix}$	-7	0.92	0.194	6
0.3	0.2	$\begin{pmatrix} 0 & \cdot & 0 & \cdot & \cdot \end{pmatrix}$	-7	0.92	0.183	6
0.2	0.133	$\begin{pmatrix} 0 & 0 & 0 & \cdot & \cdot \end{pmatrix}$	-9.13	0.87	0.226	7
0.167	0.111	$\begin{pmatrix} 0 & \cdot & 0 & 0 & 0 & \cdot \end{pmatrix}$	-9.37	1.547	0.236	8

Table A.3: Generator matrix $\mathbf{G}(D) = \left(1 \;\; (1 + D + D^4)/(1 + D^2)\right)$, $L_c = 5$.

R_{CC}	R_{TC}	pruning pattern	E_s/N_0 [dB]	δ [dB]	area	$d_{free,CC}$
0.5	0.333		-4.73	0.35	0.086	5
0.417	0.278	$\begin{pmatrix} 0 & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-5.64	0.44	0.228	5
0.4	0.267	$\begin{pmatrix} 0 & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-5.07	1.26	0.2	5
0.375	0.25	$\begin{pmatrix} 0 & \cdot & \cdot & \cdot \end{pmatrix}$	-5.2	1.63	0.201	5
0.333	0.222	$\begin{pmatrix} 0 & 0 & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$	-6.76	1.167	0.263	5
0.3	0.2	$\begin{pmatrix} 0 & 0 & \cdot & \cdot & \cdot \end{pmatrix}$	-5.80	2.12	0.202	5
0.25	0.167	$\begin{pmatrix} 0 & 0 & \cdot & \cdot \end{pmatrix}$	-6.27	2.4	0.248	8
0.2	0.133	$\begin{pmatrix} 0 & 0 & 0 & \cdot & \cdot \end{pmatrix}$	-6.73	3.27	0.306	8
0.167	0.111	$\begin{pmatrix} 0 & 0 & 0 & 0 & \cdot & \cdot \end{pmatrix}$	-7.18	3.74	0.354	9
0.1	0.067	$\begin{pmatrix} 0 & 0 & 0 & 0 & \cdot \end{pmatrix}$	-8.47	5.36	0.412	11

Appendix B

Decoder Scheduling of Hybrid Turbo Codes

In this appendix, we present an analysis of the decoding process of hybrid concatenated codes. The decoding is more complicated than of parallel or serial concatenations and is a combination of both. However, it is not clear in which order the decoders of the component codes have to be activated to achieve maximum possible mutual information or minimum decoding complexity. We will call these hybrid concatenated codes with interleavers and Turbo-decoding *hybrid Turbo codes*.

We first describe the system model of the encoding and decoding of hybrid concatenated codes, and the correct arrangement of all single decoders will be shown. Afterwards, we present a detailed description of the information extraction and processing between all component decoders. Moreover, we discuss the optimisation issues and a description of the scheduling optimisation obtained by multiple EXIT charts.

System Model

A hybrid concatenation is defined by a combination of a parallel and a serial concatenation, i.e., a parallel concatenation of serial concatenated codes. Figure B.1 presents a simple example with two parallel branches with two serial concatenated component codes each. The parallel branches as well as the serial concatenated component codes are each separated by interleavers of appropriate size. In this example, the outer codes

are of rate $R_{11} = R_{21} = 1/2$ and the inner code rates are $R_{12} = R_{22} = 2/3$.

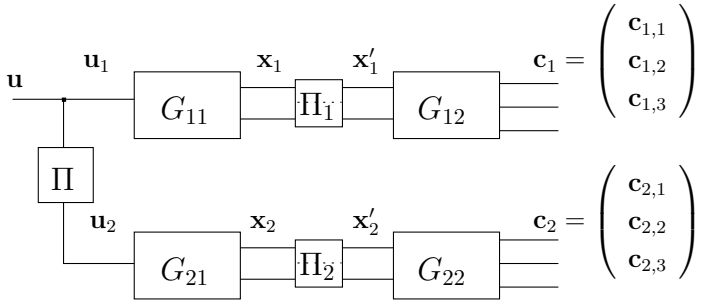


Figure B.1: Encoder structure of a hybrid concatenated code with interleavers

In the following, all component codes are assumed to be recursive systematic convolutional (RSC) codes. We define the interleavers in the upper and lower branch, Π_1 and Π_2, not to mix their upper and lower input bit streams, such that the whole hybrid Turbo code is systematic and the information bits only have to be transmitted once. The coded bit stream is composed as follows

$$\begin{aligned}\mathbf{c} \;=\; \big(&c_{1,1}(1)\; c_{1,2}(1)\; c_{1,3}(1)\; c_{2,2}(1)\; c_{2,3}(1)\\ &c_{1,1}(2)\; c_{1,2}(2)\; c_{1,3}(2)\; c_{2,2}(2)\; c_{2,3}(2)\;\; \ldots\big)\,,\end{aligned}$$

where $c_{1,1}(1) = u(1)$, $c_{1,1}(2) = u(2)$ and so on. The decoding structure of such a hybrid Turbo codes is complicated and represents a combination of the decoding structures of parallel and serial concatenations. Thus, we will first present the decoding structures of these simple concatenations and later explain how to combine them.

For the parallel concatenation with two component encoders and an interleaver Π_p in between, we define the information sequence of the whole encoder to be $\mathbf{u}$ and the information sequences of the component encoders as $\mathbf{u}_1 = \mathbf{u}$ and $\mathbf{u}_2 = \mathbf{u}(\Pi_p)$. The respective coded sequences are called $\mathbf{c}_1$ and $\mathbf{c_2}$. As component decoders, we use APP decoders (a-posteriori probability, e.g. BCJR, Log-MAP) which have two inputs and two outputs in form of log-likelihood ratios or L-values. In Figures B.2 and B.3, these decoders are represented by boxes with two inputs at the left and two outputs at the right. The decoders of the parallel concatenation receive information from the channel, called intrinsic information. This intrinsic information can be interpreted as a-priori information concerning the coded bit stream and is, thus, called $L_a(\hat{\mathbf{c}}_i)$ (upper input). The second (lower) input represents the a-priori information concerning the uncoded bit stream, denoted by $L_a(\hat{\mathbf{u}}_i)$. Accordingly, the decoder outputs two log-likelihood ratios

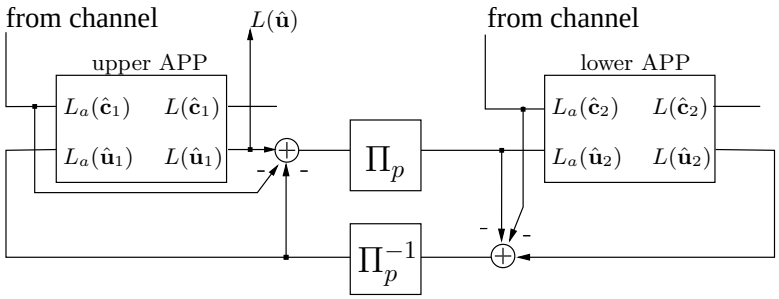


Figure B.2: Decoder structure of a parallel concatenated code with an interleaver

corresponding to the estimated coded (upper output) and the uncoded (lower output) bit streams denoted by $L(\hat{\mathbf{c}}_i)$ and $L(\hat{\mathbf{u}}_i)$, respectively. For systematic encoders, the output value of each bit is composed of the two a-priori values and the so-called extrinsic information, which is gained by the decoding process. In order to avoid statistical dependencies, the two decoders only exchange the extrinsic L-values corresponding to the uncoded bit stream $L_e(\hat{\mathbf{u}}_i)$. Thus, the a-priori values are subtracted from the estimated values before passing them as a-priori information to the next decoder. Figure B.2 shows the decoding structure of such a parallel concatenation.

For a serial concatenation with interleaver Π_s, let $\mathbf{u}$ and $\mathbf{x}$ be the input and output of the outer encoder, and $\mathbf{x}'$ and $\mathbf{c}$ be the input and output of the inner encoder, respectively, similar to the notations of the upper or lower branch in Fig. B.1. The decoder structure is similar to that of the parallel concatenation, except for two differences. The first difference is the input and output of the outer decoder. The second, i.e., outer decoder receives no intrinsic information directly from the channel but from the estimated values of the inner decoder. The a-priori information concerning the uncoded input of the outer decoder is zero all the time, since there is no information from this side of the decoder. Furthermore, the outer decoder does not pass information corresponding to the uncoded but to the coded bits to the inner decoder, since the (interleaved) coded output of the outer encoder corresponds to the uncoded input of the inner encoder. The second difference is that the inner and the outer decoder only subtract the a-priori knowledge about the uncoded and the coded data, respectively. All in all, the inner decoder estimates the uncoded data and the outer one estimates its coded data. Both decoders only subtract from their estimated values what they received from the other decoder.

Like hybrid concatenated encoders are a mixture of a parallel and a serial concatenation,

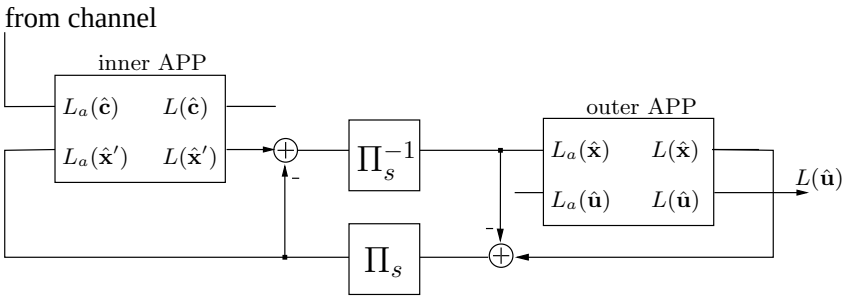


Figure B.3: Decoder structure of a serial concatenated code with an interleaver

also the decoder of a hybrid structure is a combination of both decoders. In this case, the single APP decoders in Fig. B.2 denoted by 'upper APP' and 'lower APP' will each contain a whole serial decoding structure like in Fig. B.3. Thereby, the uncoded a-priori input of the parallel decoder, $L_a(\hat{\mathbf{u}}_i)$, will be connected to the uncoded input of the outer decoder of Fig. B.3, then denoted by $L_a(\hat{\mathbf{u}}_i)$. The corresponding coded input in Fig. B.2 will be connected to the intrinsic input of Fig. B.3. The two outputs of the respective APP decoders in Fig. B.2 are connected to the coded output of the inner component decoder and to the uncoded output of the outer component decoder of Fig. B.3. These connections are depicted in Fig. B.4, which presents the serial decoding inside the upper APP decoder of a parallel concatenation.

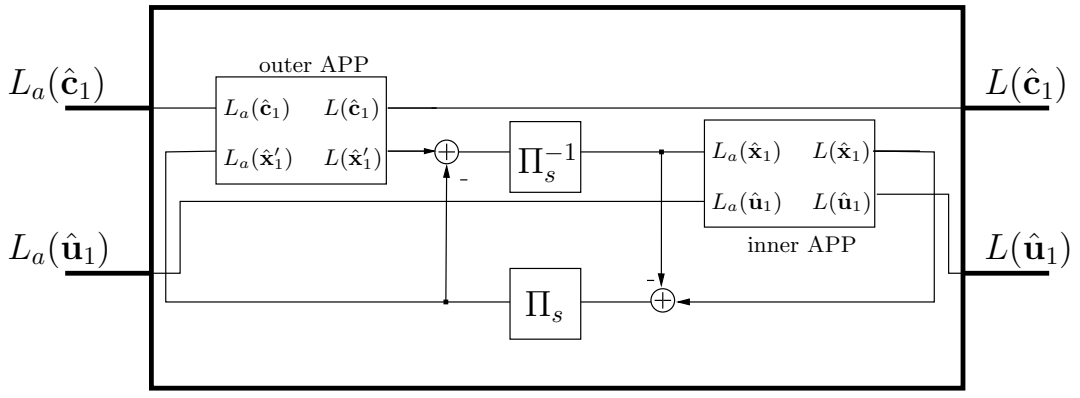


Figure B.4: Decoder structure of a serial concatenation inside an APP decoder

Information Transfer of Decoding

In the last section, we described how the component decoders are connected to each other. In this section, we explain the information processing during the decoding process. The decoding can most clearly be illustrated using the encoding structure, see Fig. B.5. The channel provides information about the outputs of the two inner encoders, called $L_a(\hat{\mathbf{c}}_1)$ and $L_a(\hat{\mathbf{c}}_2)$. We assume the upper branch to be decoded first. Thus, the upper inner decoder starts by estimating $\mathbf{x}_1'$ and passing the estimated L-values to the upper outer decoder. After estimating $\mathbf{x}_1$ and $\mathbf{u}_1$, this decoder subtracts its a-priori L-values $L_a(\hat{\mathbf{x}}_1)$ (provided by the upper inner decoder) from its estimated L-values $L(\hat{\mathbf{x}}_1)$ and passes them again to the upper inner decoder. This proceeding is performed for a certain number of iterations, denoted by $n_{it,1}$. The upper branch now subtracts the initial incoming channel information $L_a(\hat{\mathbf{c}}_1)$ from the estimated input sequence $L(\hat{\mathbf{u}}_1) = L(\hat{\mathbf{u}})$ and passes it to the lower branch.

The decoding of the lower branch starts with the activation of the outer decoder, which receives a-priori information from the upper branch $L_a(\hat{\mathbf{u}}_2)$ and from the channel $L_a(\hat{\mathbf{x}}_2)$ (note that the inner encoder is systematic and the channel information can be passed to the outer decoder without activation of the inner decoder). The decoder subtracts the a-priori values $L_a(\hat{\mathbf{x}}_2)$ from its estimated values $L(\hat{\mathbf{x}}_2)$ and passes the information to the lower inner decoder. This decoder subtracts the a-priori values $L_a(\hat{\mathbf{x}}_2')$ from its estimated values $L(\hat{\mathbf{x}}_2')$. The lower branch is decoded with $n_{it,2}$ iterations ending with an activation of the outer decoder, which subtracts the initial channel information $L_a(\hat{\mathbf{c}}_2)$ and the a-priori values coming from the upper branch $L_a(\hat{\mathbf{u}}_2)$ from its estimation $L(\hat{\mathbf{u}}_2)$. This whole process is executed for $n_{it,g}$ iterations, called global iterations.

For the analysis of the decoding, it is possible to construct EXIT charts [tB01b] corresponding to the local as well as to the global iterations. Although these different EXIT charts interact, the connections cannot easily be seen. Thus, we propose to use only the global EXIT chart. Hereto, we consider each serial decoding structure of a branch as one component decoder in a parallel concatenation. The global EXIT chart contains mutual information concerning the a-priori values $L_a(\hat{\mathbf{u}}_i)$ and the extrinsic values $L_e(\hat{\mathbf{u}}_i) = L(\hat{\mathbf{u}}_i) - L_a(\hat{\mathbf{u}}_i) - L_a(\hat{\mathbf{c}}_i)$. For finitely long sequences of length N, mutual information between a data sequence $\mathbf{u}$ and the corresponding L-values $L_{a/e}(\hat{\mathbf{u}})$ can be

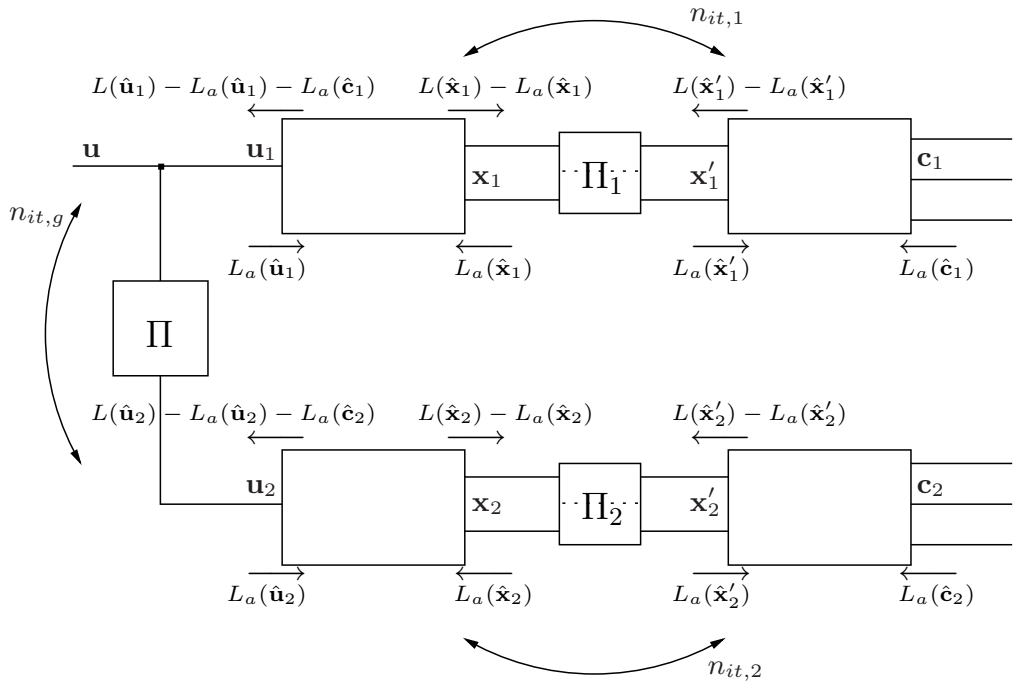


Figure B.5: Decoder structure of a hybrid Turbo decoder

calculated by the following two expectation values

$$I_a = I(\mathbf{u}; L_a(\hat{\mathbf{u}})) = \mathrm{E}\left\{1 - \log_2\left(1 + e^{-u_l \cdot L_a(\hat{u}_l)}\right)\right\} \tag{B.1}$$

and

$$I_e = I(\mathbf{u}; L_e(\hat{\mathbf{u}})) = \mathrm{E}\left\{1 - \log_2\left(1 + e^{-u_l \cdot L_e(\hat{u}_l)}\right)\right\} \ . \tag{B.2}$$

These expressions are computed for each parallel branch, where the extrinsic information corresponds to the L-values achieved after the whole number of local iterations $n_{it,1/2}$.

An important new approach to the analysis is not only including the transfer curves of a fixed number of local iterations in the global EXIT chart but for a range of numbers of local iterations. This means, the EXIT chart contains one curve per parallel branch for the information transfer corresponding to one local iteration, another curve representing the information transfer after the second local iteration etc.

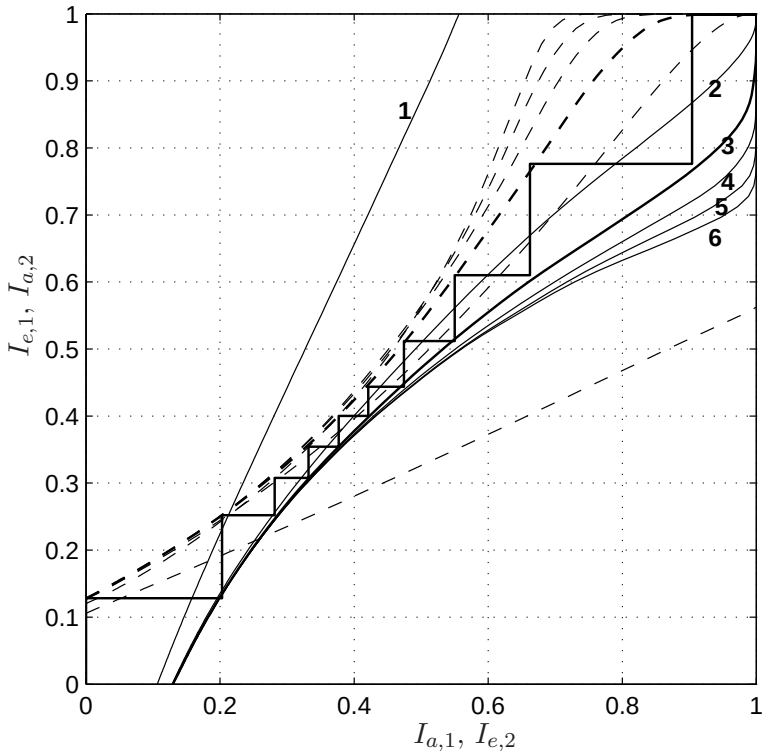


Figure B.6: Multiple EXIT chart for various numbers of local iterations at $E_b/N_0 = 0.17$dB.

Figure B.6 depicts such a multiple EXIT chart at $E_b/N_0 = 0.17$ dB for example codes given by

$$G_{11} = G_{21} = \begin{pmatrix} 1 & \frac{D^2}{1+D+D^2} \end{pmatrix} \tag{B.3}$$

and

$$G_{12} = G_{22} = \begin{pmatrix} 1 & 0 & \frac{1+D+D^2}{1+D^2} \\ 0 & 1 & \frac{1}{1+D^2} \end{pmatrix} . \tag{B.4}$$

The EXIT chart contains the transfer curves for $n_{it,1} = 1, \ldots, 6$ as dashed lines and $n_{it,2} = 1, \ldots, 6$ as solid lines. We additionally inserted the simulated decoding trajectories for a hybrid decoding scheme with $n_{it,1} = n_{it,2} = 3$ and $n_{it,g} = 10$. It is obvious that the edges of the trajectories lie on the transfer curves corresponding to $n_{it,1} = n_{it,2} = 3$, marked as bold lines.

Scheduling Optimisation

In the last section, we described the decoding of a hybrid Turbo code for fixed numbers of local and global iterations $n_{it,1}$, $n_{it,2}$, and $n_{it,g}$. From the EXIT chart in Fig. B.6, we can see the number of local iterations $n_{it,1}$ and $n_{it,2}$ which are required in order to pass the bottleneck between the two transfer curves.

For $n_{it,1} = n_{it,2} = 1$, the mutual information would get stuck at $(I_{a,1}; I_{e,1}) = (I_{a,2}; I_{e,2}) \approx 0.18$ and for $n_{it,1} = n_{it,2} = 2$, the maximum achievable mutual information would be $(I_{a,1}; I_{e,1}) = (I_{a,2}; I_{e,2}) \approx 0.36$. However, when choosing $n_{it,1}$ and $n_{it,2}$ differently, e.g. $n_{it,1} = 3$ and $n_{it,2} = 2$, the bottleneck just opens at 0.17 dB.

Of course, we can also vary the number of local iterations each time, one of the branches is decoded. We could, e.g., decode the upper branch with $n_{it,1} = 3$ iterations, jump to the lower branch and decode it with $n_{it,2} = 4$ iterations, jump back to the upper branch and decode with $n_{it,1} = 2$ iterations etc. Thus, the numbers of local iterations may be described by vectors of length $n_{it,g}$ where the ith component represents the number of local iterations during global iteration i. The information transfer of such a variable decoding process can also be visualised in a multiple EXIT chart shown in Fig. B.6. The edges of the trajectories now do not touch only one of the curves on each side but jump between several of the transfer curves. The decoding of the hybrid Turbo code example from above is shown in Fig. B.7 for the number of global iterations $n_{it,g} = 11$ and the local iteration vectors

$$n_{it,1} = [1\ 1\ 2\ 2\ 3\ 3\ 3\ 3\ 3\ 3\ 4]$$

and

$$n_{it,2} = [1\ 2\ 2\ 2\ 3\ 3\ 3\ 3\ 3\ 3\ 2]\ .$$

When considering the EXIT chart in Fig. B.7, we can also minimise the computational complexity of decoding. Once through the bottleneck, there are several possibilities of reaching the point $(I_a; I_e) = 1$. By applying a search algorithm, we would be able to find the decoding schedule which converges with lowest complexity. Hereto, we consider the complexity estimation from [CRWC05], where the number of equivalent additions is given for the decoding of a convolutional code with m memory elements. For a Log-MAP decoder, the complexity was computed as

$$N_{conv}(m) = 48 \cdot 2^m - 13\ . \tag{B.5}$$

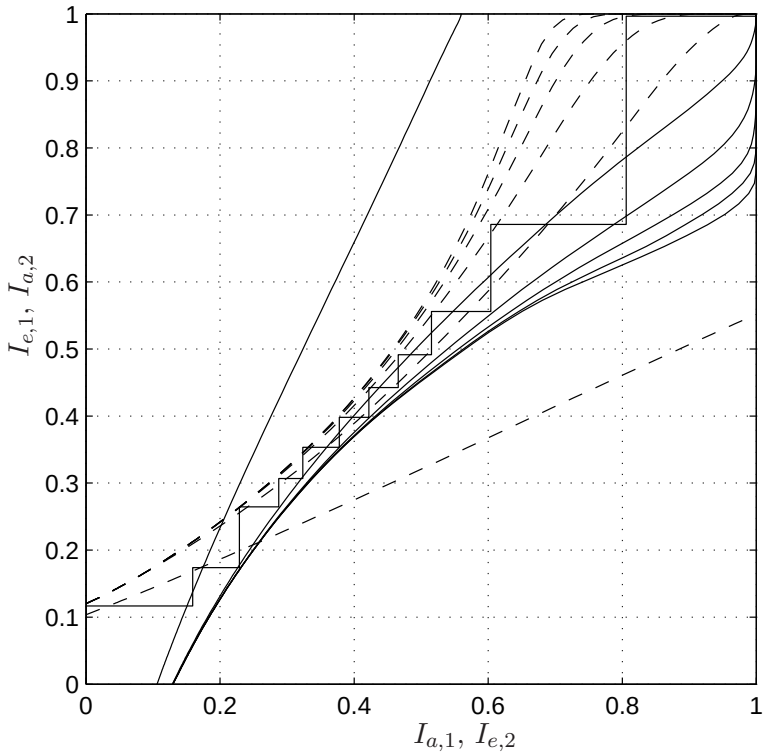


Figure B.7: Multiple EXIT chart for varying numbers of local iterations.

For a complete decoding of a hybrid Turbo code, we perform $n_{it,g}$ global decoding iterations, where each iteration contains the decoding of both branches, which in turn consist of $n_{it,1/2}$ and $n_{it,1/2}-1$ decodings of the outer and inner codes, respectively, plus one additional decoding of the very first inner decoder. Thus, the overall complexity is given by

$$\begin{aligned} N_{TC} &= n_{it,g} \cdot \Big(n_{it,1} \cdot N_{conv}(m_{11}) + (n_{it,1} - 1) \cdot \ldots \\ &\quad N_{conv}(m_{12}) + n_{it,2} \cdot N_{conv}(m_{21}) + (n_{it,2} - 1) \cdot \ldots \\ &\quad N_{conv}(m_{22}) \Big) + N_{conv}(m_{11}) \,, \end{aligned} \tag{B.6}$$

where m_{ij} denotes the number of memory elements of encoder G_{ij}. For the above example codes (B.3) and (B.4), the numbers of memory elements are $m_{11} = m_{21} = m_{12} = m_{22} = 2$ which leads to

$$N_{TC} = n_{it,g} \cdot 358 \cdot (n_{it,1} + n_{it,2} - 1) + 179 \tag{B.7}$$

for constant $n_{it,1}$ and $n_{it,2}$. For the same codes and variable numbers of local iterations,

the number of equivalent additions can be determined by

$$N_{TC} = \left(1 + \sum_{i=1}^{n_{it,g}} 2 \cdot n_{it,1}(i) + 2 \cdot n_{it,2}(i) - 2\right) \cdot 179 \; . \tag{B.8}$$

We now compare the complexity of the two decoding schedules that lead to a converging decoding process. First, we give the complexity of a decoding with constant and equal numbers of local iterations, i.e., $n_{it_1} = n_{it,2} = 3$. From Fig. B.6, we can see that $n_{it,g} = 10$ is required to converge. With Eq. (B.7), we compute $N_{TC,3-3} = 18079$ or equivalently, the number of decoder activations is 101. When improving the decoding by applying the decoding schedule given in Fig. B.7, the number of equivalent additions is $N_{TC,var} = 15931$ and the decoding only needs 89 decoder activations, which is a reduction of almost 12%. Thus, we are able to reduce the decoding complexity of such a hybrid Turbo decoding by means of the new presented decoding analysis. Furthermore, we found that the bit-error rate is not affected by this improvement.

Appendix C

Optimised LDPC Variable Node Degree Distributions for Higher Order Constellations

This section presents optimised degree distributions for rate-1/2 UEP-LDPC codes for higher order constellations. Given are tables for 8-PSK and 64-QAM modulation schemes, with different threshold offsets $\epsilon = \{0,\ 0.1\ , 0.2\ , 0.3\}$ dB. The check node degree polynomial is defined by $\rho(x) = 0.00749x^7 + 0.99101x^8 + 0.00150x^9$ which was shown to yield good results [RSU01]. Results are given for protection class proportions $\boldsymbol{\alpha} = [0.3,\ 0.7]$ as well as $\boldsymbol{\alpha} = [0.1,\ 0.9]$.

Table C.1: Degree distributions for UEP-LDPC codes optimised for 8-PSK modulation and class proportions $\boldsymbol{\alpha} = [0.3,\ 0.7]$.

		$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\epsilon = 0$ dB	$\mathcal{M}^1$	$\lambda_9 = 0.1703$ $\lambda_{10} = 0.0555$ $\lambda_{30} = 0.1811$	$\lambda_3 = 0.1673$	$\lambda_2 = 0.1240$
	$\mathcal{M}^2$	$\lambda_{30} = 0.0854$	$\lambda_4 = 0.0225$ $\lambda_5 = 0.0738$ $\lambda_7 = 0.0117$	$\lambda_2 = 0.0878$ $\lambda_3 = 0.0022$ $\lambda_4 = 0.0183$
$\epsilon = 0.1$dB	$\mathcal{M}^1$	$\lambda_{12} = 0.3290$ $\lambda_{30} = 0.1782$	$\lambda_3 = 0.1070$	$\lambda_2 = 0.1585$
	$\mathcal{M}^2$		$\lambda_3 = 0.0396$ $\lambda_4 = 0.1026$ $\lambda_5 = 0.0165$	$\lambda_2 = 0.0547$ $\lambda_3 = 0.0137$
$\epsilon = 0.2$dB	$\mathcal{M}^1$	$\lambda_{15} = 0.4840$ $\lambda_{30} = 0.0327$	$\lambda_3 = 0.0848$	$\lambda_2 = 0.1733$
	$\mathcal{M}^2$		$\lambda_3 = 0.0782$ $\lambda_4 = 0.0940$	$\lambda_2 = 0.0414$ $\lambda_3 = 0.0115$
$\epsilon = 0.3$dB	$\mathcal{M}^1$	$\lambda_{16} = 0.4993$	$\lambda_3 = 0.0268$	$\lambda_2 = 0.2163$
	$\mathcal{M}^2$	$\lambda_{16} = 0.0345$	$\lambda_3 = 0.1849$ $\lambda_4 = 0.0291$	$\lambda_2 = 0.0092$

Table C.2: Degree distributions for UEP-LDPC codes optimised for 8-PSK modulation and class proportions $\boldsymbol{\alpha} = [0.1,\ 0.9]$.

		$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\epsilon = 0$ dB	$\mathcal{M}^1$	$\lambda_8 = 0.0211$ $\lambda_9 = 0.0119$ $\lambda_{30} = 0.3171$	$\lambda_3 = 0.2015$	$\lambda_2 = 0.1332$
	$\mathcal{M}^2$	$\lambda_7 = 0.0306$ $\lambda_8 = 0.1095$ $\lambda_{30} = 0.0231$	$\lambda_4 = 0.0047$ $\lambda_5 = 0.0474$	$\lambda_2 = 0.0786$ $\lambda_4 = 0.0213$
$\epsilon = 0.1$dB	$\mathcal{M}^1$	$\lambda_{30} = 0.3336$	$\lambda_3 = 0.1588$ $\lambda_4 = 0.0564$ $\lambda_6 = 0.0673$	$\lambda_2 = 0.1178$
	$\mathcal{M}^2$		$\lambda_7 = 0.1381$ $\lambda_8 = 0.0167$	$\lambda_2 = 0.0913$ $\lambda_3 = 0.0200$
$\epsilon = 0.2$dB	$\mathcal{M}^1$	$\lambda_{30} = 0.3336$	$\lambda_3 = 0.1635$ $\lambda_5 = 0.0894$ $\lambda_6 = 0.0671$	$\lambda_2 = 0.1072$
	$\mathcal{M}^2$		$\lambda_7 = 0.1157$	$\lambda_2 = 0.0985$ $\lambda_3 = 0.0250$
$\epsilon = 0.3$dB	$\mathcal{M}^1$	$\lambda_{30} = 0.3336$	$\lambda_3 = 0.1578$ $\lambda_5 = 0.0343$ $\lambda_6 = 0.1595$	$\lambda_2 = 0.1023$
	$\mathcal{M}^2$		$\lambda_6 = 0.0842$	$\lambda_2 = 0.1038$ $\lambda_3 = 0.0246$

Table C.3: Degree distributions for UEP-LDPC codes optimised for 64-QAM modulation and class proportions $\boldsymbol{\alpha} = [0.3,\ 0.7]$.

		$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\epsilon = 0$ dB	$\mathcal{M}^1$	$\lambda_3 = 0.0638$	$\lambda_2 = 0.0191$	
		$\lambda_{30} = 0.2764$	$\lambda_3 = 0.1024$	
	$\mathcal{M}^2$		$\lambda_2 = 0.0169$	$\lambda_2 = 0.1314$
	$\mathcal{M}^3$	$\lambda_{30} = 0.0867$	$\lambda_4 = 0.0230$	$\lambda_2 = 0.0464$
			$\lambda_7 = 0.0326$	$\lambda_3 = 0.0310$
			$\lambda_8 = 0.1223$	$\lambda_4 = 0.0480$
$\epsilon = 0.1$dB	$\mathcal{M}^1$	$\lambda_{13} = 0.3902$	$\lambda_3 = 0.1017$	$\lambda_2 = 0.0196$
		$\lambda_{28} = 0.0125$		
	$\mathcal{M}^2$			$\lambda_2 = 0.1482$
	$\mathcal{M}^3$	$\lambda_{29} = 0.0754$	$\lambda_3 = 0.0540$	$\lambda_2 = 0.0479$
		$\lambda_{30} = 0.0090$	$\lambda_5 = 0.1234$	$\lambda_3 = 0.0114$
			$\lambda_6 = 0.0077$	
$\epsilon = 0.2$dB	$\mathcal{M}^1$	$\lambda_{14} = 0.4422$	$\lambda_3 = 0.0956$	$\lambda_2 = 0.0200$
		$\lambda_{23} = 0.0160$		
	$\mathcal{M}^2$			$\lambda_2 = 0.1482$
	$\mathcal{M}^3$	$\lambda_{25} = 0.0109$	$\lambda_3 = 0.0583$	$\lambda_2 = 0.0476$
		$\lambda_{26} = 0.0167$	$\lambda_5 = 0.1229$	$\lambda_3 = 0.0099$
			$\lambda_6 = 0.0117$	
$\epsilon = 0.3$dB	$\mathcal{M}^1$	$\lambda_{15} = 0.4797$	$\lambda_3 = 0.0274$	$\lambda_2 = 0.0661$
	$\mathcal{M}^2$			$\lambda_2 = 0.1482$
	$\mathcal{M}^3$	$\lambda_{15} = 0.0207$	$\lambda_3 = 0.1652$	$\lambda_2 = 0.0026$
			$\lambda_6 = 0.0819$	$\lambda_3 = 0.0082$

Table C.4: Degree distributions for UEP-LDPC codes optimised for 64-QAM modulation and class proportions $\boldsymbol{\alpha} = [0.1,\ 0.9]$.

		$\mathcal{P}^1$	$\mathcal{P}^2$	$\mathcal{P}^3$
$\epsilon = 0$ dB	$\mathcal{M}^1$	$\lambda_{30} = 0.2764$	$\lambda_3 = 0.1662$	$\lambda_2 = 0.0191$
	$\mathcal{M}^2$			$\lambda_2 = 0.1482$
	$\mathcal{M}^3$	$\lambda_{30} = 0.0572$	$\lambda_3 = 0.0180$	$\lambda_2 = 0.0464$
			$\lambda_4 = 0.0710$	$\lambda_3 = 0.0130$
			$\lambda_7 = 0.0326$	
			$\lambda_8 = 0.1223$	
$\epsilon = 0.1$dB	$\mathcal{M}^1$	$\lambda_{30} = 0.3336$	$\lambda_3 = 0.1891$	
	$\mathcal{M}^2$			$\lambda_2 = 0.1482$
	$\mathcal{M}^3$		$\lambda_4 = 0.0233$	$\lambda_2 = 0.0543$
			$\lambda_5 = 0.0855$	$\lambda_3 = 0.0297$
			$\lambda_9 = 0.0465$	
			$\lambda_{10} = 0.0898$	
$\epsilon = 0.2$dB	$\mathcal{M}^1$	$\lambda_{30} = 0.3336$	$\lambda_3 = 0.1853$	
			$\lambda_4 = 0.0052$	
	$\mathcal{M}^2$		$\lambda_4 = 0.0508$	$\lambda_2 = 0.1228$
				$\lambda_2 = 0.0672$
	$\mathcal{M}^3$		$\lambda_7 = 0.0564$	$\lambda_3 = 0.0486$
			$\lambda_8 = 0.1301$	
$\epsilon = 0.3$dB	$\mathcal{M}^1$	$\lambda_{30} = 0.3336$	$\lambda_3 = 0.1157$	
			$\lambda_4 = 0.0979$	
	$\mathcal{M}^2$		$\lambda_3 = 0.0385$	$\lambda_2 = 0.1225$
	$\mathcal{M}^3$		$\lambda_7 = 0.1180$	$\lambda_2 = 0.0694$
			$\lambda_8 = 0.0586$	$\lambda_3 = 0.0457$

Appendix D

List of Mathematical Symbols

Latin symbols

$\mathcal{A}$	signal constellation, symbol alphabet
$\mathcal{A}_{in}$, $\mathcal{A}_{out}$	channel input and output alphabet
C	channel capacity
C_i	channel capacity of level i (multilevel codes)
$\tilde{C}_i$	non-optimal channel capacity of level i
$\mathcal{C}$	code, set of codewords
c_i	ith check node
d_H	Hamming distance
$d_{v_{max}}$, $d_{c_{max}}$	maximum variable/check node degree
d_0, d_1	intersymbol distances in a constallation
E_b/N_0	signal-to-noise ratio w.r.t. bit energy
E_s/N_0	signal-to-noise ratio w.r.t. symbol energy
$g(D)$, $\mathbf{G}(D)$	generator polynomial/matrix of a code
$\mathbf{H}$	parity-check matrix
$I(Y;X)$	mutual information between random variables Y and X
K	number of information bits at the encoder input
L_c	constraint length
L_p	pruning period
l_m	number of bits per symbol
M	number of parity-check constraints
M_m	cardinality of a signal set
$\mathcal{M}^j$	modulation class j
N	number of code bits at the encoder ouput

N_p	number of protection classes
N_m	number of modulation classes
$N_{\mathcal{P}^i}$	total number of variable nodes in $\mathcal{P}^i$
$N_{\mathcal{M}^j}$	total number of variable nodes in $\mathcal{M}^j$
$\mathcal{N}_{v_i}^l$	set of check nodes which are covered by a tree of depth l and with root variable node v_i
$\bar{\mathcal{N}}_{v_i}^l$	set of check nodes which are not covered by a tree of depth l and with root variable node v_i
$\mathcal{N}(m, \sigma^2)$	Gaussian random variable with mean m and variance σ^2
P_s, P_b	symbol/bit-error probability
$\mathcal{P}^i$	protection class i
R	code rate
R_i	code rate of a subcode or of a level in a multilevel code
R_{CC}	code rate of a convolutional code
R_{TC}	code rate of a Turbo code
$\mathbf{u} = (u_0 \ u_1 \ldots)$	information sequence or word
v_i	ith variable node
w_H	Hamming weight
x_{cv}	mutual information from check node to variable node
x_{vc}	mutual information from variable node to check node

Greek symbols

$\alpha = [\alpha_1 \ \ldots \ \alpha_{N_p-1}]$	protection class proportions
$\beta = [\beta_1 \ \ldots \ \beta_{N_m}]$	modulation class proportions
δ	threshold of an LDPC code ensemble
ϵ	threshold offset
ε	erasure probability
$\lambda(x)$	variable node degree distribution polynomial
λ_i	fraction of edges connected to variable nodes of degree i
$\tilde{\lambda}_i$	fraction of degree-i variable nodes
$\lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k}$	coefficient corresponding to modulation class $\mathcal{M}^j$, protection class $\mathcal{P}^k$, and degree i
$\boldsymbol{\lambda}_{\mathcal{M}^j}^{\mathcal{P}^i}$	vector containing all degrees of $\lambda_{\mathcal{M}^j,i}^{\mathcal{P}^k}$
$\rho(x)$	check node degree distribution polynomial
ρ_i	fraction of edges connected to check nodes of degree i
$\tilde{\rho}_i$	fraction of degree-i check nodes
$\boldsymbol{\rho} = [\rho_2, \ldots, \rho_{d_{cmax}}]$	check node degree distribution coefficients
σ^2	noise variance
$\sigma^2 = [\sigma_1 \ \ldots \ \sigma_{N_m}]$	vector of noise variances per modulation class

Appendix E

List of Acronyms

ACE	approximate cycle extrinsic message degree
ARQ	automatic repeat request
AWGN	additive white Gaussian noise
BCJR	Bahl-Cocke-Jelinek-Raviv
BEC	binary erasure channel
BER	bit-error rate
BI-AWGN	binary input additive white Gaussian noise
BP	belief propagation
BPSK	binary phase shift keying
DE	density evolution
EMD	extrinsic message degree
EXIT	extrinsic information transfer
EZW	embedded zero-tree wavelet
GA	Gaussian approximation
HOC	higher order constellation
LDPC	low-density parity-check
LLR	log-likelihood ratio
LP	linear programming
MAP	maximum a-posteriori
MI	mutual information
ML	maximum likelihood
MLC	multilevel codes
MSD	multistage decoding
NSC	non-recursive nonsystematic convolutional
PSK	phase shift keying

PEG	progressive edge-growth
PCPC	path-compatible pruned convolutional
QAM	quadrature amplitude modulation
QPSK	quaternary phase shift keying
RCPC	rate-compatible punctured convolutional
RSC	recursive systematic convolutional
SNR	signal-to-noise ratio
UEP	unequal error protection

Own Publications

[HvD06] W. Henkel and N. von Deetzen. Path Pruning for Unequal Error Protection Turbo Codes. In *Proc. of the IEEE Intern. Zürich Seminar on Communications 2006*, February 2006.

[HvDH+07] W. Henkel, N. von Deetzen, K. Hassan, L. Sassatelli, and D. Declercq. Some UEP Concepts in Coding and Physical Transport. In *Proc. of the IEEE Sarnoff Symposium 2007*, April 2007.

[SvD08] S. Sandberg and N. von Deetzen. Design of Bandwidth-Efficient Unequal Error Protection LDPC Codes. Submitted to IEEE Trans. on Communications, 2008.

[vDH06] N. von Deetzen and W. Henkel. Decoder Scheduling of Hybrid Turbo Codes. In *Proc. of the IEEE Intern. Symposium of Information Theory (ISIT) 2006*, July 2006.

[vDH08a] N. von Deetzen and W. Henkel. On Code Design for Unequal Error Protection Multilevel Coding. In *Proc. of the 7th ITG Conference on Source and Channel Coding 2008*, January 2008.

[vDH08b] N. von Deetzen and W. Henkel. Unequal Error Protection Multilevel Coding and Hierarchical Modulation for Multimedia Transmission. In *Proc. of the IEEE Intern. Symposium of Information Theory (ISIT) 2008*, July 2008.

[vDS07] N. von Deetzen and S. Sandberg. Design of Unequal Error Protection LDPC Codes for Higher Order Constellations. In *Proc. of the IEEE Intern. Conference on Communications (ICC) 2007*, June 2007.

[vDSL08] N. von Deetzen, S. Sandberg, and James LeBlanc. On the UEP Capabilities of Several LDPC Construction Algorithms. Submitted to IEEE Trans. on Communications, 2008.

Bibliography

[AKtB02] A. Ashikhmin, G. Kramer, and S. ten Brink. Extrinsic Information Transfer Functions: A Model and Two Properties. In *Proceedings of the Conference on Information Sciences and Systems 2002*, pages 742–747, Princeton, NJ, USA, March 2002.

[AKtB04] A. Ashikhmin, G. Kramer, and S. ten Brink. Extrinsic information transfer functions: model and erasure channel properties. *IEEE Transactions on Information Theory*, 50:2657–2673, November 2004.

[BB99] G. Buch and F. Burkert. Concatenated Reed-Muller Codes for Unequal Error Protection. *IEEE Communications Letters*, 3(7):202–204, July 1999.

[BCJR74] L.R. Bahl, J. Cocke, F. Jelineka, and J. Raviv. Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate. In *IEEE Transactions on Information Theory*, volume 20, pages 248–287, March 1974.

[BGT93a] C. Berrou, A. Glavieux, and P. Thitimaijshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proceedings of the IEEE International Conference on Communications (ICC) 1993*, volume 2, pages 1064–1070, May 1993.

[BGT93b] C. Berrou, A. Glavieux, and T. Thitimajshima. Near Shannon limit error-correcting coding and decoding: turbo codes. In *Proceedings of the IEEE International Conference on Communications (ICC) 1993*, pages 1064–1070, May 1993.

[Cal93] A.R. Calderbank. Multilevel codes for unequal error protection. *IEEE Transactions on Information Theory*, 39:1234 – 1248, January 1993.

[CL96] G. Caire and G. Lechner. Turbo Codes with Unequal Error Protection. In *Electronics Letters*, volume 32, no.7, pages 629–631, March 1996.

[CRU01] S.-Y. Chung, T.J. Richardson, and R.L. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation. *IEEE Transactions on Information Theory*, 47(2):657–670, February 2001.

[CRWC05] I.A. Chatzigeorgiou, M.R. Rodrigues, I.J. Wassell, and R. Carrasco. A Comparison of Convolutional and Turbo Coding Schemes for Broadband FWA Systems. In *Proceedings of the 12th International Conference on Telecommunications*, May 2005.

[CT06] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2 edition, sep 2006.

[CTB98] G. Caire, G. Taricco, and E. Biglieri. Bit-interleaved coded modulation. *IEEE Transactions on Information Theory*, 44(3):927 – 946, May 1998.

[DLM05] F. Danesharan, M. Laddomada, and M. Mondin. An algorithm for the estimation of the minimum distance of LDPC codes. In *Proceedings of the IEEE Wireless Communications and Networking Conference 2005*, volume 2, pages 1046 – 1049, March 2005.

[DP97] D. Divsalar and F. Pollara. Hybrid Concatenated Codes an Iterative Decoding. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT) 1997*, Ulm, Germany, June 1997.

[DVB] ETS 300 744 rev 1.2.1, (1999-01) digital broadcasting systems for television, sound and data services (DVB-T); framing structure, channel coding and modulation for digital terrestrial.

[FR93] K. Fazel and M.J. Ruf. Combined multilevel coding and multiresolution modulation. In *Proceedings of the IEEE International Conference on Communications (ICC) 1993*, pages 1081–1085, May 1993.

[Gal62] R.G. Gallager. Low-density parity-check codes. *IEEE Transactions on Information Theory*, 8(1):21 – 28, January 1962.

[GZY03] X. Gaol, H. Zhang, and D. Yuan. Image transmission in multilevel coded wavelet packet multicarrier systems. In *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology*, pages 431–434, December 2003.

[Hag88] J. Hagenauer. Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications. *IEEE Transactions on Communications*, 36(4), April 1988.

[Hag04] J. Hagenauer. The EXIT Chart - Introduction to Extrinsic Information Transfer in Iterative Processing. In *Proceedings of 12th European Signal Processing Conference (EUSIPCO 2004)*, pages 1541–1548, Vienna, Austria, September 2004.

[HEA01] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold. Progressive edge-growth Tanner graphs. In *Proc. IEEE Global Telecommunications Conference 2001*, volume 2, pages 995–1001, November 2001.

[HEA05] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold. Regular and irregular progressive edge-growth Tanner graphs. *IEEE Trans. Inf. Theory*, 51(1):386–398, January 2005.

[HL06] C. Heneghan and Y. Liu. Optimizing Scalable Media Content Delivery using Hierarchical Modulation Techniques. In *Proceedings of the European Symposium on Mobile Media Delivery 2006*, September 2006.

[HSMP01] J. Hou, P.H. Siegel, L.B. Milstein, and D. Pfister. Multilevel coding with low-density parity-check component codes. In *Proceedings of the Global Communications Conference (Globecom) 2001*, volume 2, pages 1016–1020, November 2001.

[HSMP03] J. Hou, P.H. Siegel, L.B. Milstein, and H.D. Pfister. Capacity-approaching bandwidth-efficient coded modulation schemes based on low-density parity-check codes. *IEEE Transactions on Information Theory*, 49(9):2141 – 2155, September 2003.

[HvD06a] W. Henkel and N. von Deetzen. Path Pruning for Unequal Error Protection Turbo Codes. In *Proceedings of the International Zürich Seminar on Communications 2006*, Zürich, Switzerland, February 2006.

[HvD06b] W. Henkel and N. von Deetzen. Path Pruning for Unequal Error Protection Turbo Codes. In *Proceedings of the IEEE International Zürich Seminar on Communications 2006*, February 2006.

[HvDH+07a] W. Henkel, N. von Deetzen, K. Hassan, L. Sassatelli, and D. Declercq. Some UEP Concepts in Coding and Physical Transport. In *Proceedings of the IEEE Sarnoff Symposium 2007*, Princeton, NJ, USA, April 2007.

[HvDH+07b] W. Henkel, N. von Deetzen, K. Hassan, L. Sassatelli, and D. Declercq. Some Unequal Error Protection Concepts in Coding and Physical Transport. In *Proceedings of the IEEE Sarnoff Symposium 2007*, April 2007.

[IH77] H. Imai and S. Hirakawa. A new multilevel coding method using error correcting codes. *IEEE Transactions on Information Theory*, 23(2):371 – 377, May 1977.

[JZ99] R. Johannesson and K.S. Zigangirov. *Fundamentals of Convolutional Coding*. IEEE press, 1999.

[KFL01] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498 – 519, February 2001.

[KH93] M. Koch and W. Henkel. 90%-Rotationally Invariant Multilevel Convolutionally Encoded QAM. *ETT 1993*, 4(2):25–31, March 1993.

[KM06] V. Kumar and O. Milenkovic. On unequal error protection LDPC codes based on Plotkin-type constructions. *IEEE Transactions on Communications*, 54(6):994–1005, June 2006.

[KP01] J. Kim and G.J. Pottie. Unequal error protection TCM codes. *IEE Proceedings - Communications*, 148(5):265 – 272, October 2001.

[KSS03a] K. Kasai, T. Shibuya, and K. Sakaniwa. Detailed representation of irregular LDPC code ensembles and density evolution. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT) 2003*, page 121, June 2003.

[KSS03b] K. Kasai, T. Shibuya, and K. Sakaniwa. Detailedly Represented Irregular Low-Density Parity-Check Codes. *IEICE Trans. on Fundamentals*, E86-A(10):2435–2444, October 2003.

[LC07] G. Liva and M. Chiani. Protograph LDPC codes design based on EXIT analysis. In *Proc. IEEE GLOBECOM 2007*, pages 3250–3254, November 2007.

[LH06] Y. Liu and C. Heneghan. Optimizing scalable media content delivery using hierarchical modulation techniques. In *Proceedings of the European Symposium on Mobile Media Delivery (EuMob) 2006*, September 2006.

[LJ83] S. Lin and D. J. Costello Jr. *Error Control Coding*. Prentice Hall, 2nd edition, 1983.

[Lon07] Maja Lončar. *Taming of the BEAST*. Phd thesis, Lund University, Department of Electrical and Information Technology, October 2007.

[LR05] Y. Li and W.E. Ryan. Bit-reliability mapping in LDPC-coded modulation systems. *IEEE Communications Letters*, 9(1):1–3, January 2005.

[NSL06] Y. Nana, E. Sharon, and S. Lytsin. Improved decoding of LDPC coded modulation. *IEEE Communications Letters*, 10(5):375–377, May 2006.

[NVCC03] M. Neri, A. Vanelli-Coralli, and G.E. Corazza. Unequal Error Protection: A Turbo Multi Level Coding Approach. In *Proceedings of the Global Communications Conference (Globecom) 2003*, volume 22, pages 102–106, San Franciso, USA, December 2003.

[PDF04a] C. Poulliat, D. Declercq, and I. Fijalkow. Optimization of LDPC codes for UEP channels. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT) 2004*, page 451, June 2004.

[PDF04b] C. Poulliat, D. Declercq, and I. Fijalkow. Optimization of LDPC codes for UEP channels and application to scalable image transmission. Draft, 2004.

[PDF07] C. Poulliat, D. Declercq, and I. Fijalkow. Enhancement of unequal error protection properties of LDPC codes. *EURASIP Journal on Wireless Communications and Networking*, 2007:Article ID 92659, 9 pages, 2007. doi:10.1155/2007/92659.

[PNRF05] H. Pishro-Nik, N. Rahnavard, and F. Fekri. Nonuniform error correction using low-density parity-check codes. *IEEE Transactions on Information Theory*, 51(7):2702–2714, July 2005.

[Pro01] J.G. Proakis. *Digital communications*. McGraw-Hill, 2001.

[Ric03] T. Richardson. Error floors of LDPC codes. In *Proc. 41st Annual Allerton Conf. on Commun., Control, and Computing*, volume 41, pages 1426–1435, October 2003.

[RPNF07] N. Rahnavard, H. Pishro-Nik, and F. Fekri. Unequal error protection using partially regular LDPC codes. *IEEE Transactions on Communications*, 55(3):387–391, March 2007.

[RSU01] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, February 2001.

[SA07] Ali Sanaei and Masoud Ardakani. LDPC code design for nonuniform power-line channels. *EURASIP Journal on Advances in Signal Processing*, 2007:Article ID 76146, 9 pages, 2007. doi:10.1155/2007/76146.

[Seg00] A. Segger. Broadcast communication on fading channels using hierarchical coded modulation. In *Proceedings of the Global Communications Conference (Globecom) 2000*, November 2000.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27: part I, pp. 379-427; part II:623–656, 1948.

[SHD06] L. Sassatelli, W. Henkel, and D. Declercq. Check-irregular LDPC codes for unequal error protection under iterative decoding. In *Proceedings of the 4th International Symposium on Turbo Codes & Related Topics 2006*, April 2006.

[SSN04] H. Sankar, N. Sindhushayana, and K.R. Narayanan. Design of low-density parity-check (LDPC) codes for high order constellations. In *Proceedings of the IEEE Global Communications Conference (Globecom) 2004*, pages 3113–3117, November 2004.

[SvD08] S. Sandberg and N. von Deetzen. Design of Bandwidth-Efcient Unequal Error Protection LDPC Codes. Submitted to IEEE Transactions on Communications, 2008.

[SWBK03] A. Sezgin, D. Wübben, R. Bönke, and V. Kün. On EXIT-charts for space-time block codes. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT) 2003*, page 64, June 2003.

[Tan81] M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533 – 547, September 1981.

[tB01a] S. ten Brink. Convergence Behavior of Iteratively Decoded Parallel Concantenated Codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

[tB01b] S. ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

[tB01c] S. ten Brink. Exploiting The Chain Rule Of Mutual Information For The Design Of Iterative Decoding Schemes. In *Proceedings 39th Annual Allerton Conference on Commununication, Control, and Computing*, pages 293–300, Monticello, Urbana-Champaign, IL, USA, October 2001.

[tBKA04] S. ten Brink, G. Kramer, and A. Ashikhmin. Design of low-density parity-check codes for modulation and detection. *IEEE Trans. Commun.*, 52(4):670–678, April 2004.

[TJVW04] T. Tian, C.R. Jones, D.J. Villasenor, and R.D. Wesel. Selective avoidance of cycles in irregular LDPC code construction. *IEEE Transactions on Communications*, 52(8):1242–1247, August 2004.

[UC76] G. Ungerböck and I. Csajka. On improving data link performance by increasing channel alphabet and introducing sequence coding. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT) 1976*, June 1976.

[Ung82a] G. Ungerböck. Channel coding with multilevel/phase signal. *IEEE Transactions on Information Theory*, 28(1):55 – 67, January 1982.

[Ung82b] G. Ungerböck. Channel coding with multilevel/phase signals. *IEEE Transactions on Information Theory*, 28:55–67, January 1982.

[Ung82c] G. Ungerboeck. Channel Coding with Multilevel/Phase Signals. *IEEE Transactions on Information Theory*, IT-28:55–67, January 1982.

[Ung87a] G. Ungerböck. Trellis-coded modulation with redundant signal sets Part I: Introduction. *IEEE Communications Magazine*, 25(2):5–11, February 1987.

[Ung87b] G. Ungerböck. Trellis-coded modulation with redundant signal sets Part II: State of the art. *IEEE Communications Magazine*, 25(2):12–21, February 1987.

[Vit67] A.J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. In *IEEE Transactions on Information Theory*, volume IT-13, pages 260–269, April 1967.

[VS08] D. Vukobratovic and V. Senk. Generalized ACE constrained progressive edge-growth LDPC code design. *IEEE Communications Letters*, 12:32–34, January 2008.

[WC02] C.-H. Wang and C.-C. Chao. Path-Compatible Pruned Convolutional (PCPC) Codes. *IEEE Transactions on Communications*, 50(2), February 2002.

[Wei93a] L.-F. Wei. Coded modulation with unequal error protection. *IEEE Transactions on Communications*, 41(10):1439 – 1449, October 1993.

[Wei93b] L.-F. Wei. Coded modulation with unequal error protection. *IEEE Transactions on Communications*, 41:1439–1449, October 1993.

[WFH99] U. Wachsmann, R. Fischer, and J. Huber. Multilevel codes: Theoretical concepts and practical design rules. *IEEE Transactions on Information Theory*, 45(5):1361 – 1391, July 1999.

[ZV03] B. Zhao and M.C. Valenti. Convergence Analysis of a New Class of Flexible Hybrid Concatenated Codes. In *Proceedings of the 35th Southeastern Symposium on System Theory*, Morgantown, West Virginia, USA, March 2003.