TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

DISSERTATION

Efficient Decoding Techniques for LDPC Codes

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

eingereicht an der
Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von
Dipl.-Ing. Gottfried Lechner
Blumengasse 44/22, 1170 Wien
geboren in Wien am 1. August 1975
Matrikelnr. 9525633

Wien, im Juli 2007 .....

Supervisor

Univ.-Prof. Dr.-Ing. Dipl.-Ing. Markus Rupp

Institut für Nachrichtentechnik und Hochfrequenztechnik

Technische Universität Wien, Austria

Examiner

Prof. Rüdiger Urbanke

I&C School of Computer and Communication Sciences

EPFL, Lausanne, Switzerland

Kurzfassung

Effiziente Decodieralgorithmen für LDPC Codes gewinnen mehr und mehr an Bedeutung, da diese Gruppe von Codes mittlerweile in vielen Standards vertreten ist. Trotz ihrer beeindruckenden theoretischen Leistungsfähigkeit, treten bei der praktischen Implementierung von LDPC Decodern Probleme wie numerische Instabilität, begrenzte Speichergröße, usw. auf. Diese Arbeit beschäftigt sich mit Methoden, welche die Decodier-Komplexität reduzieren und gleichzeitig den Verlust gering halten. Um dies zu erreichen, untersuchen wir drei Punkte: die Vereinfachung der Komponenten eines LDPC Decoders, die Verwendung von harten Entscheidungen innerhalb des Decoders und die Kombination des LDPC Decoders mit anderen Aufgaben einer iterativen Empfängerstruktur.

Für die Vereinfachung der Komponenten analysieren wir den min-sum Algorithmus und entwickeln ein theoretisches Gerüst mit welchem bekannte heuristische Methoden zur Verbesserung dieser Approximation erklärt werden. Weiters adaptieren wir den Algorithmus für irreguläre LDPC Codes und erreichen eine Verbesserung nahe zum optimalen Algorithmus.

Die Einschränkung der internen Werte auf harte Entscheidungen führt zu einer Reduktionen der Speichererfordernisse und erlaubt die Implementierung von Decodern mit hohem Datendurchsatz, wie sie zum Beispiel in optischen Systemen verwendet werden. Wir verwenden extrinsic information transfer charts, um diese Gruppe von Decodern zu analysieren, wobei sich als Spezialfall der Algorithmus von Gallager ergibt. Wir verallgemeinern diesen Algorithmus für den Fall, daß der Kanal mehr Information als eine harte Entscheidung zur Verfügung stellt und verwenden die Analyse, um Schranken für diese Gruppe von Decodern herzuleiten. Weiters zeigen wir, wie Codes für diese Algorithmen optimiert werden können.

Zuletzt präsentieren wir die Optimierung eines LDPC Codes für den Fall, daß der Decoder im Kontext einer Empfängerstruktur betrachtet wird, wobei der Empfänger weitere Aufgaben wie Demapping oder Multi-User Detektion übernimmt. Wir zeigen, wie der LDPC Code effizient optimiert werden kann, wobei die Verteilungen der Symbol und Prüfknoten gemeinsam optimiert werden. Diese Optimierung des Codes erfordert nur die Kenntnis der Transfer Funktion der beteiligten Empfänger-Teile, welche entweder analytisch oder durch Simulation gewonnen werden kann. Nach einer allgemeinen Herleitung der Code Optimierung, wenden wir diese auf iteratives Demapping und iterative Multi-User Detektion an.

Abstract

Efficient decoding techniques for LDPC codes are in demand, since these codes are included in many standards nowadays. Although the theoretical performance of LDPC codes is impressive, their practical implementation leads to problems like numerical inaccuracy, limited memory resources, etc. We investigate methods that are suited to reduce the decoding complexity while still keeping the loss in performance small. We aim to reduce the complexity using three approaches: simplification of the component decoders, restricting the message passing algorithm to binary variables and combining the LDPC decoder with other receiver tasks like demapping or multi-user detection.

For the simplification of the component decoders, we analyze the min-sum algorithm and derive a theoretical framework which is used to explain previous heuristic approaches to improve the performance of this algorithm. Using this framework, we are able to modify the algorithm in order to achieve good performance for regular as well as irregular LDPC codes.

Restricting all internal messages of an LDPC decoder to binary variables, leads to a significant reduction of memory requirements and allows the implementation of high-throughput decoders which are used for example in optical communication systems. We analyze binary message passing decoders using a general framework which is based on extrinsic information transfer charts. As special cases, we rederive Gallagers bit-flipping algorithm. Our derivation allows to generalize these algorithms for the case where soft-information from the channel is available, while still using binary variables for all internal messages. The analysis is used to derive bounds and to optimize LDPC codes for binary message passing decoders.

Finally, we consider the optimization of an LDPC code where the decoder is not considered on its own, but in the context of a receiver structure which performs additional tasks like demapping or multi-user detection. We show how the code optimization can be performed efficiently by optimizing the degree distributions of variable and check nodes jointly. Our code optimization requires only knowledge of the extrinsic information transfer function of the receiver front-end, which can be obtained either analytically or via simulations. After a general derivation of the code optimization, we apply the optimization tools to iterative demapping and iterative multi-user detection.

Acknowledgment

It is not possible to name all people who contributed to my research and supported me during my work on this thesis. To name just a few, I would like to thank all my colleagues at ftw. for their support over the last years, especially Jossy Sayir, Katja Kapfer, Christoph Mecklenbräucker, and Thomas Zemen. Besides many fruitful discussion and suggestions, I also had a lot of fun while working on my thesis at ftw., where I found a friendly work environment and had support over the whole time. I also want to thank Markus Kommenda and Horst Rode for creating and maintaining this research center.

Furthermore, I would like to thank all my colleagues at the Vienna University of Technology, especially my supervisor Prof. Markus Rupp for critical reading and constructive comments.

Finally, I have to thank all my friends and my family for their non-scientific support over the whole time.

Contents

1 Introduction

In 1948, Claude Elwood Shannon introduced channel capacity and proved the existence of error correcting codes which enable systems to transmit at rates below channel capacity with arbitrary small error probability [Sha48]. Furthermore, it was shown that randomly generated codebooks are able to approach capacity with high probability if the length of the codewords tends to infinity. However, the computational complexity of the optimal decoder is exponential in the codeword length and therefore, coding techniques with some structure in the codebook were investigated.

During the following decades, a lot of coding families were invented, starting with Hamming codes, Golay codes, Reed Muller codes, convolutional codes, BCH codes and Reed Solomon codes—just to mention a few. The goal was to construct codes with good properties (i.e. large minimum distance) and to find low-complexity decoding algorithms, which are able to perform optimal decoding for these families.

The decoding algorithms of the these codes were well studied, many standards and applications include particularly convolutional and Reed Solomon codes, and there exist efficient and fast hardware implementations of these decoders.

With the presentation of turbo codes in 1993 by Berrou, Glavieux and Thitimajshima [BGT93], the coding community experienced a fundamental change of concepts. The concept of turbo codes was able to outperform all previous coding techniques by using very long codewords. This was enabled by the associated iterative decoding algorithm, which has a computational complexity that is linear in the codeword length.

In the following years, iterative algorithms were the main focus of the coding community. It was recognized that already in 1962, Gallager introduced low-density parity-check (LDPC) codes, that are also suited for iterative decoding algorithms but were mostly forgotten because of the absence of the required hardware (e.g. interconnections, required memory) at that time. With the reinvention of LDPC codes, their optimization, and the development of analysis techniques for iterative decoders, it was possible to design communication systems that are able to transmit at rates approaching channel capacity while still using decoding algorithms with a computational complexity that is linear in the codeword length.

After this fundamental theoretical work, turbo and LDPC codes moved into standards like DVB-S2, DSL, WLAN, etc. and are under consideration for others. When implementing iterative decoding algorithms, problems like numerical inaccuracy,

limited memory resources, etc. came up. Therefore, research was triggered to develop low-complexity iterative decoding algorithms. The aim of this thesis is to bridge some gaps between theoretical achievements and practical coding systems.

1.1 Outline

This section gives an overview of the thesis and the content of the individual chapters.

Chapter 2: Basics of Iterative Decoding

This chapter presents the basic building blocks of iterative decoding systems, which will be used throughout this thesis. After discussing channel models and component codes, we introduce turbo and LDPC codes, which are the most popular codes suited for iterative decoding.

Chapter 3: Analysis of Iterative Systems

The development of analysis tools for iterative systems enabled their optimization and these analysis techniques are an important tool for the simplification and optimization of sub-optimal low-complexity algorithms. This chapter gives an overview of existing analysis tools and discusses their assumptions and approximations.

Chapter 4: Simplifying Component Decoders

Although iterative decoders consist of simple component codes, decoding techniques for these codes are still challenging to implement in practice. By avoiding nonlinear functions, the decoding complexity can be reduced and numerical instabilities can be eliminated. In this chapter, we present an analysis of the approximation of a decoder for a single parity-check code, which is one component of an LDPC code. Using this analysis, we are able to improve the approximation and lower the gap between the optimal and sub-optimal decoder significantly. The theoretical analysis is based on [LS04, Lec04a, Lec04b, Lec05, LS06a, LS06b] and applications of the approximations can be found in [LSR04, LBSR04].

Chapter 5: Binary Message Passing Decoders

The quantization of messages significantly affects the performance of iterative decoders. For systems that require high data throughput, binary message passing

algorithms which use a single bit for all internal messages, are often the only algorithms that can be implemented in hardware. In this chapter we present an analysis of binary message passing decoders and extend the results to the case were a finer quantized channel observation is available, but the decoder is still restricted to use binary quantization for all internal messages. The main results of this chapter are based on [LPK07].

Chapter 6: Joint Receiver Tasks

Replacing a classical coding strategy (e.g. concatenation of convolutional codes and Reed Solomon codes) with an iterative decoder, often increases the amount of required hardware significantly. However, by combining the iterative decoding algorithm with other receiver tasks like demapping or multi-user detection, the overall receiver complexity can be reduced. This chapter presents examples of such receiver structures and a general optimization technique to improve their performance. This chapter is based on [LSL06, Lec06, LB07].

2 Basics of Iterative Decoding

This chapter introduces the basic building blocks and principles of iterative decoding systems, which will be used throughout this thesis. First, we present channel models and their unified description. Then we introduce component codes and the concept of concatenated codes in order to construct powerful codes that can be decoded by iterative algorithms. Finally, we describe turbo codes and low-density parity-check (LDPC) codes—the most famous codes suited for iterative decoding.

2.1 Channel Models

We will consider memoryless channels with binary input B taken from the alphabet $\mathcal{B} = \{+1, -1\}$ as shown in Figure 2.1. Let Y denote the output of the channel taking values from the output alphabet $\mathcal{Y}$ which can be discrete or continuous. The channel is defined by the channel transition probability density function $p(Y = y|B = b)$ where we allow for the existence of the Dirac distribution to include the case of a discrete output alphabet in the general framework. Furthermore, we will restrict

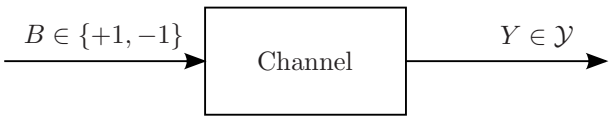


Figure 2.1: Binary input channel.

the channels to be symmetric, resulting in

$$p(Y = y|B = b) = p(Y = -y|B = -b). \tag{2.1}$$

We will call these channels binary input, symmetric output, memoryless channels (BISOMC). This family of channels includes discrete outputs like the binary erasure channel (BEC) and the binary symmetric channel (BSC) as well as channels with continuous output like the binary input additive white Gaussian noise (BIAWGN) channel.

The mutual information between B and Y can be written as

$$\begin{aligned} I(B;Y) &= h(Y) - h(Y|B) \\ &= -\int_{\mathcal{Y}} p(Y=y)\log p(Y=y)dy \\ &\quad + \sum_{b\in\mathcal{B}}\int_{\mathcal{Y}} p(B=b,Y=y)\log p(Y=y|B=b)dy. \end{aligned} \tag{2.2}$$

The input distribution that maximizes the mutual information between B and Y, and thus is equivalent to the channel capacity C, is the uniform distribution [CT91, Theorem 8.2.1] which allows to write the capacity as

$$C = \int_{\mathcal{Y}} -\frac{g(y)+g(-y)}{2}\log\frac{g(y)+g(-y)}{2} + g(y)\log g(y)dy, \tag{2.3}$$

where $g(y)$ is a shorthand notation for $p(Y=y|B=+1)$. For binary input, symmetric channels with equally likely input symbols, the output of the channel is completely described by $g(y)$.

2.1.1 Abstraction of the Channel

In order to develop decoding strategies that are not restricted to a certain output alphabet of the channel model, we introduce a framework that allows for a uniform description of the output of BISOMCs. We define an L-value [HOP96][1] as

$$L(B) \hat{=} \log\frac{p(B=+1)}{p(B=-1)}, \tag{2.4}$$

which is called *a-priori* L-value because it is not dependent on the channel output. Furthermore we define

$$L(B|y) \hat{=} \log\frac{p(B=+1|Y=y)}{p(B=-1|Y=y)}, \tag{2.5}$$

and

$$L(y|B) \hat{=} \log\frac{p(Y=y|B=+1)}{p(Y=y|B=-1)}, \tag{2.6}$$

which are called *a-posteriori* and *channel* L-value, respectively. The sign of the L-value determines whether $B=+1$ or $B=-1$ is more likely and the magnitude of the L-value is a measure for the certainty of this decision. Therefore, a maximum

[1] These quantities are often called *log-likelihood ratios*. We use the term L-value, since the enumerator and denominator are not always likelihood functions.

a-posteriori decoder decides for $B = +1$ if the a-posteriori L-value is larger than zero and for $B = -1$ otherwise. Using Bayes' law, the a-posteriori L-value can be separated into the sum of the a-priori and the channel L-value

$$L(B|y) = L(B) + L(y|B). \tag{2.7}$$

In the following, we will assume that the decoder has no a-priori information about the individual symbols, hence $L(B)$ will always be zero unless mentioned otherwise. Therefore, we can exchange $L(B|y)$ and $L(y|B)$ when needed. We will consider the computation of the channel L-value as being part of the channel, and the decoding algorithms presented and developed in the rest of this thesis will be independent of the channel model. However, note that the *performance* of the decoding algorithms is not independent of the channel.

There is always a unique conversion between L-values and (conditional) probabilities which allows us to exchange them when needed. For example, we can write the a-posteriori probability as

$$p(B = b|Y = y) = \frac{e^{b\frac{L(B|y)}{2}}}{e^{\frac{L(B|y)}{2}} + e^{-\frac{L(B|y)}{2}}}. \tag{2.8}$$

The probability density function of the L-values completely describes the channel, since the conditional channel transition probability can be derived from them. Let $q(L)$ denote the density of the L-values corresponding to $p(Y = y|B = +1)$. The capacity is then written as

$$C = \int_{\mathcal{L}} -\frac{q(l) + q(-l)}{2} \log \frac{q(l) + q(-l)}{2} + q(l) \log q(l) dl, \tag{2.9}$$

where $\mathcal{L}$ denotes the support of $q(l)$.

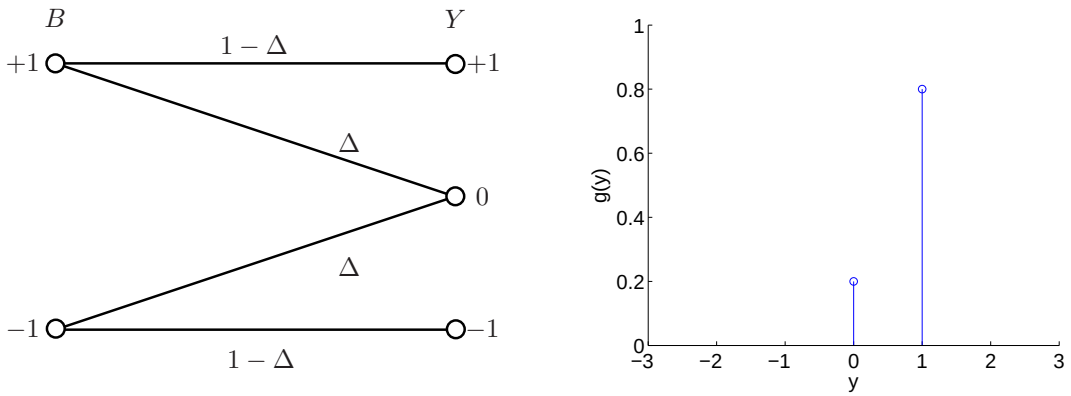
For every BISOMC, the density of the L-values $q(l)$ satisfies a symmetry condition [RSU01, Proposition 1]

$$q(-l) = e^{-l} q(l). \tag{2.10}$$

Therefore, we can simplify the computation of the capacity to

$$C = \int_{\mathcal{L};l>0} q(l) \left(e^{-l} \log \frac{2}{1 + e^{l}} + \log \frac{2}{1 + e^{-l}} \right) dl. \tag{2.11}$$

Since this is a linear function of $q(l)$, the capacity of a mixture of channels can be computed as the average of the capacity of the sub-channels.



(a) Channel Model.

(b) Conditional Output Probability $g(y)$ for $\Delta = 0.2$.

Figure 2.2: Binary erasure channel.

2.1.2 Binary Erasure Channel

A binary erasure channel with erasure probability Δ and the corresponding conditional output probability is shown in Figure 2.2(a) and 2.2(b), respectively. The output alphabet $\mathcal{Y}$ of this channel is the discrete set $\{+1, 0, -1\}$. The function $g(y)$ is given by

$$g(y) = (1 - \Delta)\delta(y - 1) + \Delta\delta(y). \tag{2.12}$$

Inserting $g(y)$ in (2.3) leads to the well known capacity of the BEC [CT91, Section 8.1.5]

$$C_{BEC}(\Delta) = 1 - \Delta. \tag{2.13}$$

The computation of the channel L-value simplifies to

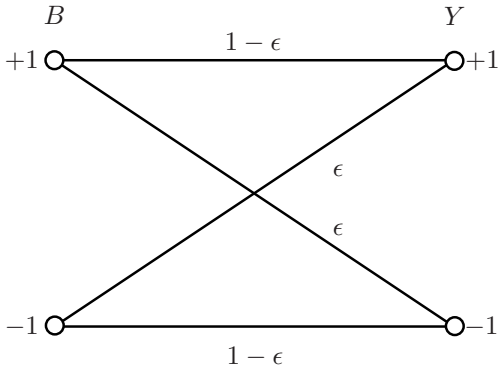
$$L(y|B) = \begin{cases} +\infty & ; \quad y = +1 \\ 0 & ; \quad y = 0 \\ -\infty & ; \quad y = -1 \end{cases}. \tag{2.14}$$

The fact that the magnitude is infinity for $y \in \{+1, -1\}$ means, that there is no uncertainty about the transmitted digit for these receive values.

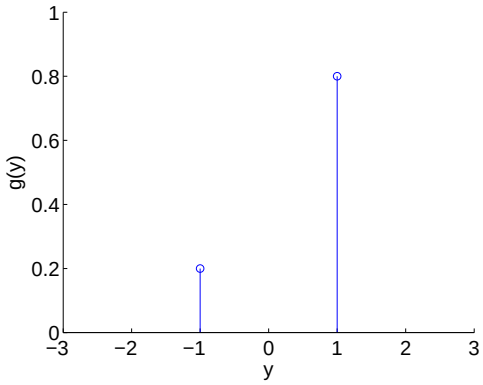
2.1.3 Binary Symmetric Channel

For the binary symmetric channel shown in Figure 2.3(a) with crossover probability ϵ and output alphabet $\{+1, -1\}$, $g(y)$ is given by

$$g(y) = (1 - \epsilon)\delta(y - 1) + \epsilon\delta(y + 1). \tag{2.15}$$



(a) Channel Model.



(b) Conditional Output Probability $g(y)$ for $\epsilon = 0.2$.

Figure 2.3: Binary symmetric channel.

An example for $\epsilon = 0.2$ is shown in Figure 2.3(b). This results in a capacity of the channel as [CT91, Section 8.1.4]

$$C_{BSC}(\epsilon) = 1 + \epsilon \log(\epsilon) + (1-\epsilon)\log(1-\epsilon) = 1 - h_b(\epsilon), \tag{2.16}$$

where $h_b(\epsilon)$ denotes the binary entropy function which is defined as

$$h_b(p) \hat{=} -p\log(p) - (1-p)\log(1-p). \tag{2.17}$$

The computation of the channel L-value simplifies to

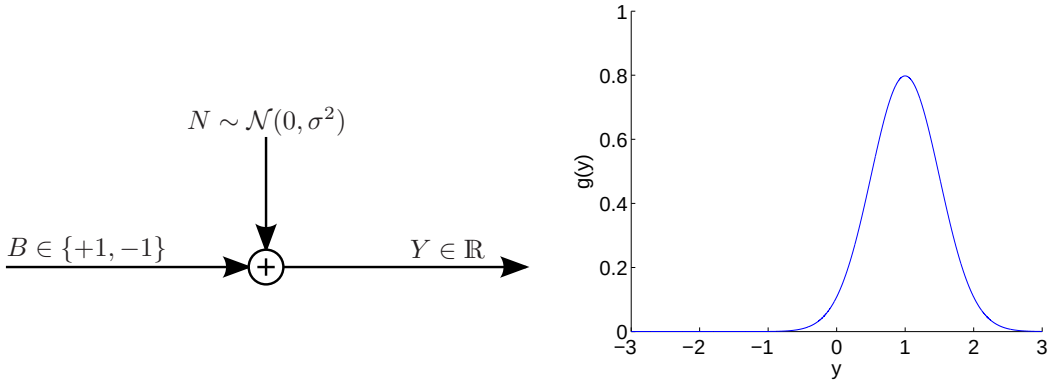
$$L(y|B) = \begin{cases} \log\frac{1-\epsilon}{\epsilon} & ; \quad y = +1 \\ \log\frac{\epsilon}{1-\epsilon} & ; \quad y = -1 \end{cases}. \tag{2.18}$$

As in the case of the BEC, the L-value alphabet contains only two nonzero values. However, since there is a nonzero crossover probability, the magnitude is finite and a function of ϵ.

2.1.4 Binary Input Additive White Gaussian Noise Channel

Figure 2.4(a) shows a BIAWGN channel with noise variance σ^2. The conditional output probability is shown in Figure 2.4(b). The output of this channel is real numbers ($\mathcal{Y} = \mathbb{R}$) and $g(y)$ is given by

$$g(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-1)^2}{2\sigma^2}}. \tag{2.19}$$



(a) Channel Model.

(b) Conditional Output Probability $g(y)$ for $\sigma = 0.5$.

Figure 2.4: Binary input additive white Gaussian noise channel.

The computation of the channel L-value simplifies to

$$L(y|B) = \frac{2}{\sigma^2} y, \tag{2.20}$$

which can take on all real values. The reliability of this L-value is a scaled version of the received value y, where the scaling factor is inverse proportional to the variance of the additive noise. The L-values are also distributed according to a Gaussian distribution with variance $\sigma_L^2 = 4/\sigma^2$ and mean $\mu_L = \sigma_L^2/2$.

The capacity of this channel cannot be written in closed form but has to be computed using numerical integration. Since this quantity is often used we define the J-function [AKtB04] as

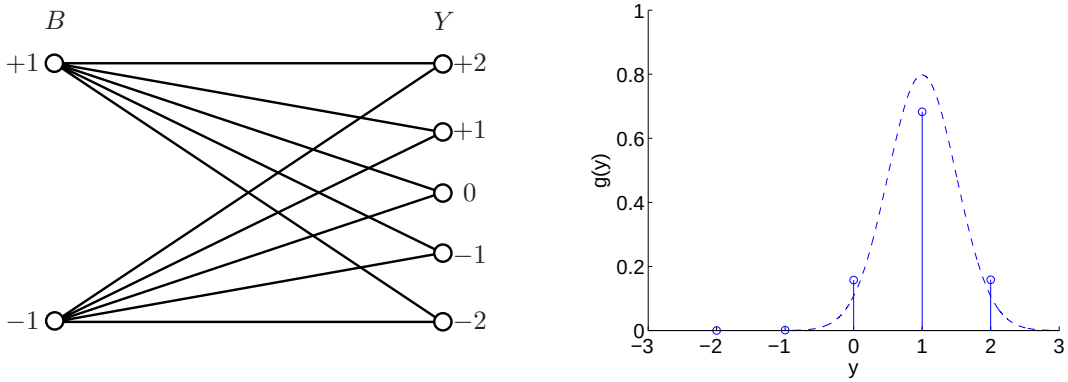
$$J(\sigma_L) \hat{=} 1 - \int_{-\infty}^{\infty} \frac{e^{-\frac{(\theta - \sigma_L^2/2)^2}{2\sigma_L^2}}}{\sqrt{2\pi}\sigma_L} \log\left(1 + e^{-\theta}\right) d\theta, \tag{2.21}$$

and the capacity of the BIAWGN channel can be written as

$$C_{BIAWGN}(\sigma) = J\left(\frac{2}{\sigma}\right) = J\left(\sigma_L\right). \tag{2.22}$$

The J-function maps values from the interval $[0, \infty]$ to the interval $[0, 1]$ and is strictly monotonically increasing. Therefore, there exists a well defined inverse which will be denoted by $J^{-1}(\cdot)$.

A BIAWGN channel with quantized output using five symbols at the output is shown in Figure 2.5(a). This channel is equivalent to parallel BSCs where the crossover probability of the BSCs and the probability that a certain BSC is used can



(a) Channel Model.

(b) Conditional Output Probability $g(y)$ for $\sigma = 0.5$.

Figure 2.5: BIAWGN with quantized outputs.

be computed as a function of the quantization intervals and the noise variance of the underlying BIAWGN channel. The subchannel that leads to a zero at the output (i.e. a BEC with $\Delta = 1$), can equivalently be described as a BSC with crossover probability 0.5, i.e. a capacity of zero [Lan05]. Each subchannel has a capacity C_i that contributes to the total capacity of the BIAWGN channel. This model is useful for analyzing quantization effects. The computation of the channel L-value reduces to the computation for the corresponding subchannel as

$$L(y|B) = \begin{cases} \log \frac{1-\epsilon_i}{\epsilon_i} & ; \quad y = +A_i \\ \log \frac{\epsilon_i}{1-\epsilon_i} & ; \quad y = -A_i \end{cases} . \tag{2.23}$$

where ϵ_i is the crossover probability of a subchannel and A_i the magnitude of the corresponding output.

2.1.5 Puncturing

In order to match the rate of the error correcting code to the capacity of the channel without changing the code parameters, many communication systems use puncturing. At the transmitter side, a certain fraction p of the code symbols is not transmitted and consequently the receiver has no information about those symbols, i.e. the receiver assigns an L-value of zero to the punctured symbols. We can model the output of the channel including the punctured symbols that are inserted as zeros at the receiver, as a multiplication of the original channel with a puncture sequence S_p, where the puncture sequence takes on the values zero and one with probability p and $1-p$, respectively. Let $g(y)$ denote the conditional transition probability of the

unpunctured channel. The conditional transition probability $g_p(y)$ after puncturing can then be written as

$$g_p(y) = g(y)p(S_p = 1) + \delta(y)p(S_p = 0) = g(y)(1-p) + \delta(y)p. \tag{2.24}$$

Using (2.3), we can relate the capacity of the punctured channel to the unpunctured channel leading to

$$C_p = (1-p)C \leq C, \tag{2.25}$$

where C and C_p denote the capacity without and with puncturing, respectively. An unpunctured system transmits one symbol per channel use over a channel with capacity C, while the punctured system transmits $\frac{1}{1-p} \geq 1$ symbols per channel use over a channel with capacity C_p. By adjusting the fraction p of punctured symbols, the system can transmit at rates approaching capacity without changing the error correcting code.

2.2 Components and Decoders

The idea behind iterative concepts is to solve a global problem by partitioning it into smaller problems that are easier to solve and iterate between these problems in order to converge to the global solution. In the case of iterative decoding, the component codes are repetition codes, single parity-check codes and convolutional codes. This section presents these codes, their definition and the associated decoding algorithms. The first two types of codes belong to the class of linear block codes which can be described by a parity-check matrix whereas convolutional codes are described using a trellis. All these codes are defined in the Galois field GF(2) with elements $X \in \{0, 1\}$. We define the mapping from X to B as

$$B = 1 - 2X, \tag{2.26}$$

which maps the values $\{0, 1\}$ from X to $\{+1, -1\}$ from B. Since this is a bijective mapping, we can exchange X and B if necessary without loss of information.

2.2.1 Linear Block Codes

Linear block codes can be described as the set of codewords $\boldsymbol{x}$ taken from a codebook $\mathcal{C}$ that satisfy a parity-check equation

$$\mathcal{C} = \left\{ \boldsymbol{x} \in \{0, 1\}^N : \boldsymbol{H}\boldsymbol{x} = 0 \right\}, \tag{2.27}$$

where all operations are performed in GF(2). The parity-check matrix $\boldsymbol{H}$ has N columns that correspond to the elements of the codeword and $M = N - K$ rows,

where each row defines a constraint. If we assume that the matrix has full rank, i.e. $\text{rank}(\boldsymbol{H}) = M$, then the degree of freedom for the information words is $N - M = K$. The rate of code R is defined as

$$R = \frac{K}{N} = 1 - \frac{M}{N}. \tag{2.28}$$

An encoder maps information blocks $\boldsymbol{u}$ of length K from $\{0,1\}^K$ to codewords $\boldsymbol{x}$ of length N from $\{0,1\}^N$, which we write as $\boldsymbol{x} = \mathcal{C}(\boldsymbol{u})$. An important property of linear codes is, that the all-zero codeword is always an element of the codebook. Furthermore, every codeword has the same properties, i.e. the same number of neighbors with the same distance in the codespace. Therefore, the probability of error averaged over all codewords is the same as the probability of error for a specific codeword, e.g. the all-zero word. These properties are important since they allow to perform the analysis for the all-zero codeword.

2.2.2 Maximum A-Posteriori Decoding

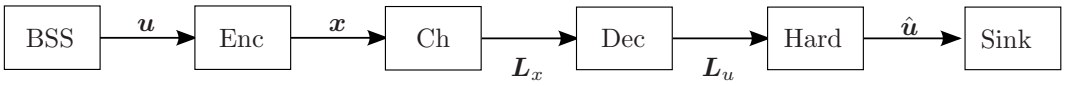


Figure 2.6: Coded transmission system.

Consider a transmission system as shown in Figure 2.6, where a vector $\boldsymbol{u}$ with elements from a binary symmetric source (BSS) is encoded to a vector $\boldsymbol{x}$, transmitted over a BISOMC and decoded at the receiver to obtain hard decisions of the elements of $\boldsymbol{u}$, denoted as $\hat{\boldsymbol{u}}$. The definition of the L-value in Section 2.1.1 was based on a memoryless channel were only one received value y contained information about the transmitted digit, i.e. we assumed that the transmitted symbols where independent of each other. For a coded system, there is a dependency between the inputs of the channel, which the decoder exploits in order to improve its decision. We therefore generalize the definition of the L-value by changing the condition $Y = y$ to the vector case. The a-posteriori L-value of the i$^{\text{th}}$ element x_i of $\boldsymbol{x}$ is then defined as

$$\begin{aligned} L(X_i|\boldsymbol{y}) &= \log \frac{p(X_i = 0|\boldsymbol{Y} = \boldsymbol{y})}{p(X_i = 1|\boldsymbol{Y} = \boldsymbol{y})} \\ &= \log \frac{p(Y_i = y_i|X_i = 0)}{p(Y_i = y_i|X_i = 1)} \\ &\quad + \log \frac{\sum_{\boldsymbol{x}:x_i=0} p(\boldsymbol{Y}_{[i]} = \boldsymbol{y}_{[i]}|\boldsymbol{X}_{[i]} = \boldsymbol{x}_{[i]})[\boldsymbol{x} \in \mathcal{C}]}{\sum_{\boldsymbol{x}:x_i=1} p(\boldsymbol{Y}_{[i]} = \boldsymbol{y}_{[i]}|\boldsymbol{X}_{[i]} = \boldsymbol{x}_{[i]})[\boldsymbol{x} \in \mathcal{C}]} \\ &= L(y_i|X_i) + L(\boldsymbol{y}_{[i]}|X_i), \end{aligned} \tag{2.29}$$

where the notation $\boldsymbol{y}_{[i]}$ denotes the vector $\boldsymbol{y}$ with the i$^{\text{th}}$ element removed and $[\cdot]$ is an indicator function that evaluates to one if its argument is true, and to zero if its argument is false. The a-posteriori L-value can be separated into two parts, the channel L-value $L(y_i|X_i)$ that is computed using the direct channel observation and an extrinsic part $L(\boldsymbol{y}_{[i]}|X_i)$ that includes all the information which is obtained from the other observations and the dependencies between the elements of $\boldsymbol{x}$. In order to compute the maximum a-posteriori decision for an element of $\boldsymbol{x}$, the decoder has to compute the sum of these two parts and base its decision on the sign of the resulting a-posteriori L-value.

In a similar way, the decoder can perform maximum a-posteriori decoding on the elements u_i of the information vector $\boldsymbol{u}$ as

$$\begin{aligned} L(U_i|\boldsymbol{y}) &= \log \frac{p(U_i = 0|\boldsymbol{Y} = \boldsymbol{y})}{p(U_i = 1|\boldsymbol{Y} = \boldsymbol{y})} \\ &= \log \frac{\sum_{\boldsymbol{u}:u_i=0} p(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{X} = \mathcal{C}(\boldsymbol{u}))}{\sum_{\boldsymbol{u}:u_i=1} p(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{X} = \mathcal{C}(\boldsymbol{u}))}, \end{aligned}$$

where the separation into two parts as in the previous case can only be performed if systematic encoding is used, i.e. $\boldsymbol{u}$ can be identified as a part of $\boldsymbol{x}$.

2.2.3 Repetition Code

A repetition code of length N can be defined by a parity-check matrix $\boldsymbol{H}$ with size $(N-1) \times N$ of the form

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 1 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \tag{2.30}$$

with a rate R that is a decreasing function in N, i.e. the rate goes to zero when the length of the repetition code goes to infinity

$$R = \frac{1}{N}. \tag{2.31}$$

The extrinsic L-value can be derived as

$$L(\boldsymbol{y}_{[i]}|X_i) = \sum_{j=1;j\neq i}^{N} L(y_j|X_j), \tag{2.32}$$

and therefore, the a-posteriori L-value is given by

$$L(X_i|\boldsymbol{y}) = \sum_{j=1}^{N} L(y_j|X_j). \tag{2.33}$$

2.2.4 Single Parity-Check Code

A single parity-check code of length N is defined by a parity-check matrix $\boldsymbol{H}$ with size $N \times 1$ which consists of a single row of ones

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}. \tag{2.34}$$

The rate R of the code is increasing with the length N and is given by

$$R = 1 - \frac{1}{N}, \tag{2.35}$$

i.e. the rate goes to one when the length of the single parity-check code goes to infinity. The extrinsic L-value can be derived as

$$L(\boldsymbol{y}_{[i]}|X_i) = 2 \tanh^{-1} \left(\prod_{j=1; j\neq i}^{N} \tanh \frac{L(y_j|X_j)}{2} \right), \tag{2.36}$$

and therefore, the a-posteriori L-value is given by

$$L(X_i|\boldsymbol{y}) = L(y_i|X_i) + 2 \tanh^{-1} \left(\prod_{j=1; j\neq i}^{N} \tanh \frac{L(y_j|X_j)}{2} \right). \tag{2.37}$$

2.2.5 Convolutional Code

Convolutional codes work on a stream of data. However, for iterative systems they are restricted to work on blocks of finite length. An optional termination, i.e. driving the encoder in a predefined state at the end of the block, can be implemented. We will use recursive systematic codes (RSC) where we use the notation (g_f, g_b) to define the feed-forward and feed-back connections of a memory m encoder in octal form.

Efficient decoding of convolutional codes is performed using the BCJR algorithm. For a description of this algorithm we refer to [BCJR74, WH00]. The algorithm accepts L-values of systematic and parity symbols. At the output, the algorithm provides the a-posteriori L-values of the systematic symbols as well as the extrinsic L-values as shown in Figure 2.7.

Figure 2.7: BCJR decoder.

2.3 Concatenated Systems

In order to be able to separate the decoding problem into smaller problems and iterate between them, we force the overall code to have a specific structure, i.e. the overall code is composed of two or more smaller codes. There are basically two methods to concatenate codes [Reg05]—parallel concatenation, where all encoders work on the same information block, and serial concatenation where each encoder works on the output of the previous encoder.

2.3.1 Parallel Concatenation

A parallel concatenation scheme is shown in Figure 2.8. We assume a memoryless binary symmetric source that produces information blocks $\boldsymbol{u}$ of length K. These information blocks are encoded using two different encoders and enter the channel as $\boldsymbol{v}$ and $\boldsymbol{w}$, respectively. A common method to implement the second encoder is to apply the same encoder as for the first path with an interleaver at the input. In practice these encoded blocks are multiplexed, transmitted over the channel and demultiplexed at the receiver. Since we are looking at memoryless channels, we can equivalently represent the system by two parallel channels. The maximum

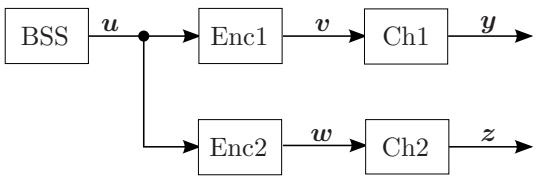


Figure 2.8: Parallel concatenation.

a-posteriori receiver has to compute the probability of every symbol given all its observations, or equivalently the L-value

$$\begin{aligned} L(U_i|\boldsymbol{y},\boldsymbol{z}) &= \log \frac{p(U_i=0|\boldsymbol{y},\boldsymbol{z})}{p(U_i=1|\boldsymbol{y},\boldsymbol{z})} \\ &= \log \frac{\sum_{\boldsymbol{u}:u_i=0} p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{u})}{\sum_{\boldsymbol{u}:u_i=1} p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{u})}, \end{aligned} \tag{2.38}$$

where we assumed that all information blocks are transmitted with the same probability and the channels are independent of each other.

The complexity to evaluate this expression is exponential in K and therefore not suitable for practical applications. However, if either $p(\boldsymbol{y}|\boldsymbol{u})$ or $p(\boldsymbol{z}|\boldsymbol{u})$ factors into the product of its marginals, there exist decoding algorithms with a complexity linear in K (e.g. the BCJR algorithm for decoding convolutional codes). We approximate $p(\boldsymbol{y}|\boldsymbol{u})$ and $p(\boldsymbol{z}|\boldsymbol{u})$ using two auxiliary variables $\boldsymbol{a}$ and $\boldsymbol{b}$ as

$$p(\boldsymbol{y}|\boldsymbol{u}) \approx p(\boldsymbol{a}|\boldsymbol{u}) = \prod_i p(a_i|u_i) \tag{2.39}$$

$$p(\boldsymbol{z}|\boldsymbol{u}) \approx p(\boldsymbol{b}|\boldsymbol{u}) = \prod_i p(b_i|u_i) \tag{2.40}$$

and therefore split the decoding problem into two parts where the two decoders compute

$$\begin{aligned} L(U_i|\boldsymbol{y},\boldsymbol{b}) &= L(U_i|b_i) + \log \frac{\sum_{\boldsymbol{u}:u_i=0} p(\boldsymbol{y}|\boldsymbol{u}) e^{\frac{1}{2}\sum_{j\neq i} u_j L(U_j|b_j)}}{\sum_{\boldsymbol{u}:u_i=1} p(\boldsymbol{y}|\boldsymbol{u}) e^{\frac{1}{2}\sum_{j\neq i} u_j L(U_j|b_j)}} \\ &= L(U_i|b_i) + L(U_i|\boldsymbol{y},\boldsymbol{b}_{[i]}) \qquad (2.41) \\ L(U_i|\boldsymbol{z},\boldsymbol{a}) &= L(U_i|a_i) + \log \frac{\sum_{\boldsymbol{u}:u_i=0} p(\boldsymbol{z}|\boldsymbol{u}) e^{\frac{1}{2}\sum_{j\neq i} u_j L(U_j|a_j)}}{\sum_{\boldsymbol{u}:u_i=1} p(\boldsymbol{z}|\boldsymbol{u}) e^{\frac{1}{2}\sum_{j\neq i} u_j L(U_j|a_j)}} \\ &= L(U_i|a_i) + L(U_i|\boldsymbol{z},\boldsymbol{a}_{[i]}), \qquad (2.42) \end{aligned}$$

where the first terms represent the a-priori L-values and the second terms the extrinsic L-values. In order to approximate $p(\boldsymbol{y}|\boldsymbol{u})$ and $p(\boldsymbol{z}|\boldsymbol{u})$ we set the L-values of the marginals of $\boldsymbol{a}$ and $\boldsymbol{b}$ to the extrinsic L-values available at the output of the other decoder

$$L(U_i|a_i) \leftarrow L(U_i|\boldsymbol{y},\boldsymbol{b}_{[i]}) \tag{2.43}$$

$$L(U_i|b_i) \leftarrow L(U_i|\boldsymbol{z},\boldsymbol{a}_{[i]}) \tag{2.44}$$

Finally, we iterate between the two decoders with the goal to improve the approximations during this iterative process and expect to converge to the solution of the joint decoding problem. The iterative decoding process is illustrated in Figure 2.9.

2.3.2 Serial Concatenation

In a serial concatenated coding system as shown in Figure 2.10, the information blocks $\boldsymbol{u}$ are first encoded by an outer encoder to blocks $\boldsymbol{v}$ which are the input

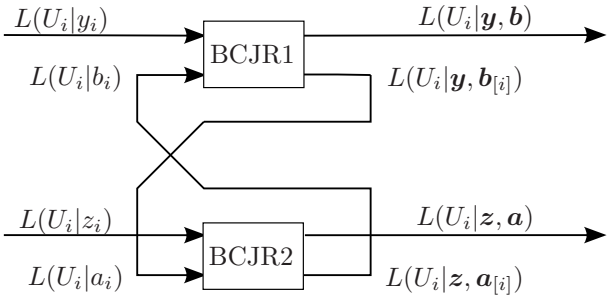


Figure 2.9: Decoding of parallel concatenation.

to the inner encoder that maps these blocks to $\boldsymbol{w}$ which are transmitted over the channel. The maximum a-posteriori decoder has to compute the L-value

$$L(U_i|\boldsymbol{y}) = \log \frac{p(U_i = 0|\boldsymbol{y})}{p(U_i = 1|\boldsymbol{y})}, \tag{2.45}$$

which has a complexity exponential in K unless the two encoders can be represented by one single encoder (which is for example not the case if there exists an interleaver in between them).

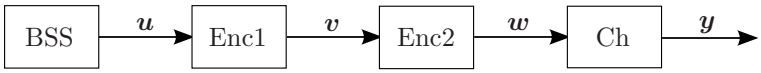


Figure 2.10: Serial concatenation.

Therefore, we split the joint decoding problem into two smaller problems. First, we attempt to decode the inner code leading to estimates of $\boldsymbol{v}$ using the observed vector $\boldsymbol{y}$ and an auxiliary variable $\boldsymbol{a}$ that provides a-priori information about the elements of $\boldsymbol{v}$

$$\begin{aligned}
L(V_i|\boldsymbol{y},\boldsymbol{a}) &= \log \frac{\sum_{\boldsymbol{v}:v_i=0} p(\boldsymbol{y}|\mathcal{C}_2(\boldsymbol{v}))p(\boldsymbol{v})}{\sum_{\boldsymbol{v}:v_i=1} p(\boldsymbol{y}|\mathcal{C}_2(\boldsymbol{v}))p(\boldsymbol{v})} \\
&\approx \log \frac{\sum_{\boldsymbol{v}:v_i=0} p(\boldsymbol{y}|\mathcal{C}_2(\boldsymbol{v})) \prod_j p(a_j)}{\sum_{\boldsymbol{v}:v_i=1} p(\boldsymbol{y}|\mathcal{C}_2(\boldsymbol{v})) \prod_j p(a_j)} \\
&= L(A_i) + \log \frac{\sum_{\boldsymbol{v}:v_i=0} p(\boldsymbol{y}|\mathcal{C}_2(\boldsymbol{v}))e^{\frac{1}{2}\sum_j a_j L(a_j)}}{\sum_{\boldsymbol{v}:v_i=1} p(\boldsymbol{y}|\mathcal{C}_2(\boldsymbol{v}))e^{\frac{1}{2}\sum_j a_j L(a_j)}} \\
&= L(A_i) + L(V_i|\boldsymbol{y},\boldsymbol{a}_{[i]}),
\end{aligned} \tag{2.46}$$

where we approximated $p(\boldsymbol{v})$ with a density $p(\boldsymbol{a})$ that factors into its marginals. This approximation is obviously not correct, since $\boldsymbol{v}$ is an encoded block with dependencies between its elements. We set the marginal L-values of the auxiliary

variable $\boldsymbol{b}$ to be equal to the extrinsic output of the inner decoder as

$$L(V_i|b_i) \leftarrow L(V_i|\boldsymbol{y}, \boldsymbol{a}_{[i]}). \tag{2.47}$$

Using this approximation, the outer decoder computes

$$\begin{aligned} L(V_i|\boldsymbol{b}) &= \log \frac{p(V_i = 0|\boldsymbol{b})}{p(V_i = 1|\boldsymbol{b})} \\ &= L(V_i|b_i) + \log \frac{\sum_{\boldsymbol{v}:v_i=0} [\boldsymbol{v} \in \mathcal{C}_1] e^{\frac{1}{2}\sum_{j\neq i} v_j L(V_j|b_j)}}{\sum_{\boldsymbol{v}:v_i=1} [\boldsymbol{v} \in \mathcal{C}_1] e^{\frac{1}{2}\sum_{j\neq i} v_j L(V_j|b_j)}} \\ &= L(V_i|b_i) + L(V_i|\boldsymbol{b}_{[i]}). \end{aligned} \tag{2.48}$$

Finally, we set

$$L(A_i) \leftarrow L(V_i|\boldsymbol{b}_{[i]}) \tag{2.49}$$

and iterate between the two decoders in order to improve the approximations. The iterative decoding process is illustrated in Figure 2.11, where decoder one has no direct observation from the channel. Furthermore, the outer decoder also computes

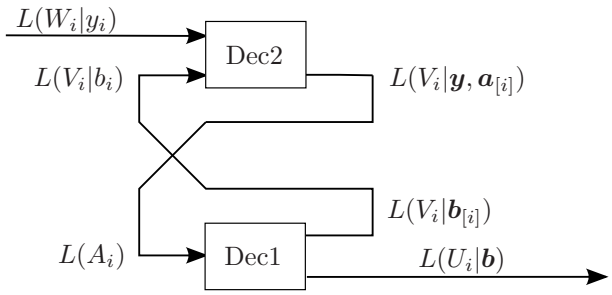


Figure 2.11: Decoding of serial concatenation.

the a-posteriori L-values of the elements of the information block as

$$\begin{aligned} L(U_i|\boldsymbol{b}) &= \log \frac{p(U_i = 0|\boldsymbol{b})}{p(U_i = 1|\boldsymbol{b})} \\ &= \log \frac{\sum_{\boldsymbol{u}:u_i=0} p(\boldsymbol{b}|\mathcal{C}_1(\boldsymbol{u}))}{\sum_{\boldsymbol{u}:u_i=1} p(\boldsymbol{b}|\mathcal{C}_1(\boldsymbol{u}))}, \end{aligned} \tag{2.50}$$

which are used to make a decision on the information symbols.

2.4 Turbo Codes

Turbo codes [BGT93] are parallel concatenated codes as shown in Section 2.3.1 using recursive systematic convolutional codes as component encoders. For the second encoder, the same type of code as for the first encoder with a previous interleaver is used. Every encoder delivers systematic symbols and a sequence of parity symbols. Since both encoders transmit the systematic symbols, the systematic part of the second encoder is punctured resulting in an overall code of rate 1/3 if component codes of rate 1/2 are used [BGT93].

Decoding of turbo codes follows the general description of Section 2.3.1, where the BCJR algorithm is used for decoding of the component codes. The first component code is decoded, and the extrinsic L-values are passed as a-priori information to the second component decoder and vice versa. Decoding is stopped after a predefined number of iterations is exceeded or a stopping criterion is satisfied, e.g. the hard decisions on the information symbols of both component decoders are identical.

2.5 Low-Density Parity-Check Codes

Low-density parity-check (LDPC) [Gal62, Gal63, Mac99] codes are binary, linear block codes that are defined by a parity-check matrix that is sparse, i.e. the number of ones per row and column is fixed, and therefore the density of ones decreases when the block length of the code increases.

LDPC codes are equivalently represented as a factor graph [KFL01] containing variable nodes and check nodes. A variable node represents an element of the codeword, whereas a check node represents a row of the parity-check matrix, i.e. a single parity-check code. An edge in the graph is drawn between every variable node and check node if there is a corresponding one in the parity-check matrix and the *degree* of a node is defined ad the number of connecting edges. The number of ones per column and per row, which correspond to the degree of the variable and check nodes in the factor graph, are denoted by d_v and d_c, respectively. If every type of node has the same degree, the code is called *regular*, otherwise it is called *irregular*. For irregular codes, we have to specify the fraction of edges that are connected to nodes of a certain degree. Let λ_i denote the fraction of edges connected to a variable node of degree i. Especially for the analysis on the binary erasure channel, it is convenient to define a polynomial $\lambda(x)$ whose coefficients are equal to the fraction of edges as

$$\lambda(x) \hat{=} \sum_{i=2}^{d_{v,max}} \lambda_i x^{i-1}. \tag{2.51}$$

For the distribution of the check node degrees, we define a similar polynomial where

the fraction of edges connected to a check node of degree i is denoted by ρ_i

$$\rho(x) \hat{=} \sum_{i=2}^{d_{c,max}} \rho_i x^{i-1}. \tag{2.52}$$

The fraction of variable and check nodes of degree i can be computed as

$$\bar{\lambda}_i \hat{=} \frac{\lambda_i / i}{\sum_{j=2}^{d_{v,max}} \lambda_j / j}, \tag{2.53}$$

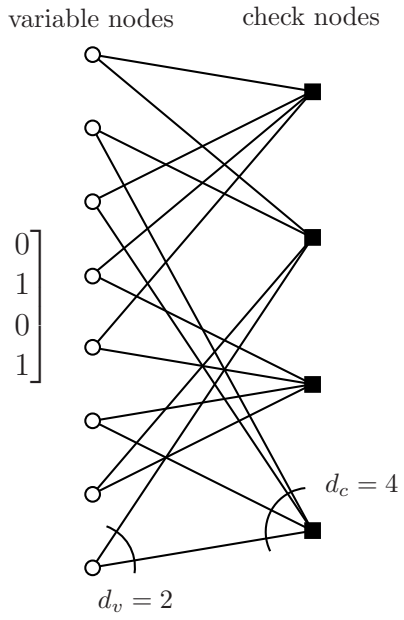
$$\bar{\rho}_i \hat{=} \frac{\rho_i / i}{\sum_{j=2}^{d_{c,max}} \rho_j / j}. \tag{2.54}$$

The definitions (2.51), (2.52) are called *edge perspective* and (2.53), (2.54) *node persepective*.

For a detailed description of LDPC codes and the iterative decoding algorithm we refer to [Mac99, RU01]. An example of a parity-check matrix and the associated factor graph of a regular LDPC code with $d_v = 2$ and $d_c = 4$ is shown in Figure 2.12.

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

(a) Parity-Check Matrix with $d_v = 2$ and $d_c = 4$.



(b) Factor Graph.

Figure 2.12: Regular LDPC code.

Although encoding of LDPC codes cannot be represented as a concatenation of codes, we can represent the decoding process in the same way as for a serial concatenated system consisting of a repetition code (variable nodes as inner code) and a single parity-check code (check nodes as outer code).

3 Analysis of Iterative Systems

The analysis of iterative systems is nontrivial, especially since most systems are nonlinear. In this chapter, we present methods and tools for the analysis of iterative systems. These tools are used to predict the convergence behavior and to determine and optimize the systems performance. First, we start with an example of a non-iterative system to introduce the main principles and methods and to give an overview of the existing analysis tools.

3.1 Density Evolution

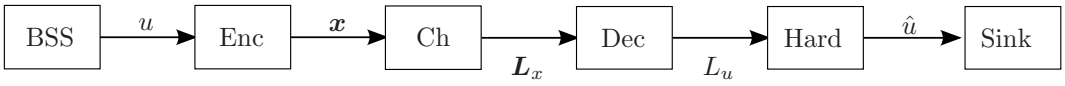


Figure 3.1: Non-iterative system.

Consider a system model as shown in Figure 3.1. A symbol of a binary symmetric source (BSS) with output alphabet $\mathcal{X} = \{0, 1\}$ is encoded using a linear block code and transmitted over a BISOMC[1]. The receiver performs decoding of the forward error correcting code and makes a hard decision of the source symbols. As an example, we will assume a repetition code of length d that is used to transmit data over a BIAWGN channel.

We consider the transmission of a single source symbol corresponding to one codeword of length d. In order to compute the probability of a wrong decision at the receiver, one can *track* the probability density function (or equivalently the probability mass function for discrete quantities) of all random variables involved in the system. Since all components involved in this structure are symmetric, the probability of error p_e is given by

$$\begin{aligned} p_e &= p(\hat{U} \neq U) \\ &= p(\hat{U} = +1|U = -1)p(U = -1) + p(\hat{U} = -1|U = +1)p(U = +1) \\ &= p(\hat{U} = -1|U = +1). \end{aligned} \tag{3.1}$$

[1]For the sake of simplicity, we included the mapping from $\{0, 1\}$ to $\{+1, -1\}$ and the computation of the L-value in the channel.

Therefore, it is sufficient to compute $p(\hat{U} = -1|U = +1)$ to derive the probability of error. Starting at the source, the conditional probability density function of U is given by

$$p(U = u|U = +1) = \delta(u - 1). \tag{3.2}$$

Repeating every symbol d times does not change the probability density function and therefore, the density of the elements of $\boldsymbol{X}$ is

$$p(X_i = x|U = +1) = \delta(x - 1), \tag{3.3}$$

for $i = 1 \dots d$. Using the channel transition probability, we can write the density of the L-values (see Section 2.1.4) as

$$p(L_{x,i} = l|U = +1) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{\left(l - \frac{\sigma_x^2}{2}\right)^2}{2\sigma_x^2}}. \tag{3.4}$$

where σ_x^2 denotes the variance of the distribution of the L-values which is related to the noise variance σ^2 of the BIAWGN channel as

$$\sigma_x^2 = \frac{4}{\sigma^2}. \tag{3.5}$$

The L-values are used to decode the repetition code as described in Section 2.2.3 by summing up all d received L-values corresponding to one data symbol. Since the distribution of the L-values is Gaussian, with variance σ_x^2, mean $\sigma_x^2/2$, and the d observations are independent due to the memoryless channel, we can compute the density of the decoded symbols by adding their means and their variances leading to

$$p(L_{u,i} = l|U = +1) = \frac{1}{\sqrt{2\pi}\sigma_u} e^{-\frac{\left(l - \frac{\sigma_u^2}{2}\right)^2}{2\sigma_u^2}}, \tag{3.6}$$

where

$$\sigma_u^2 = d\sigma_x^2. \tag{3.7}$$

The receiver decides for $\hat{U} = -1$ if $L_u < 0$. Therefore, the probability of error is given by

$$p(\hat{U} = -1|U = +1) = \int_{-\infty}^{0} p(L_u = l|U = +1)dl = Q\left(\frac{\sigma_u}{2}\right), \tag{3.8}$$

where the function $Q(\cdot)$ is defined as

$$Q(\phi) \hat{=} \frac{1}{\sqrt{2\pi}} \int_{\phi}^{\infty} e^{-\frac{\psi^2}{2}} d\psi. \tag{3.9}$$

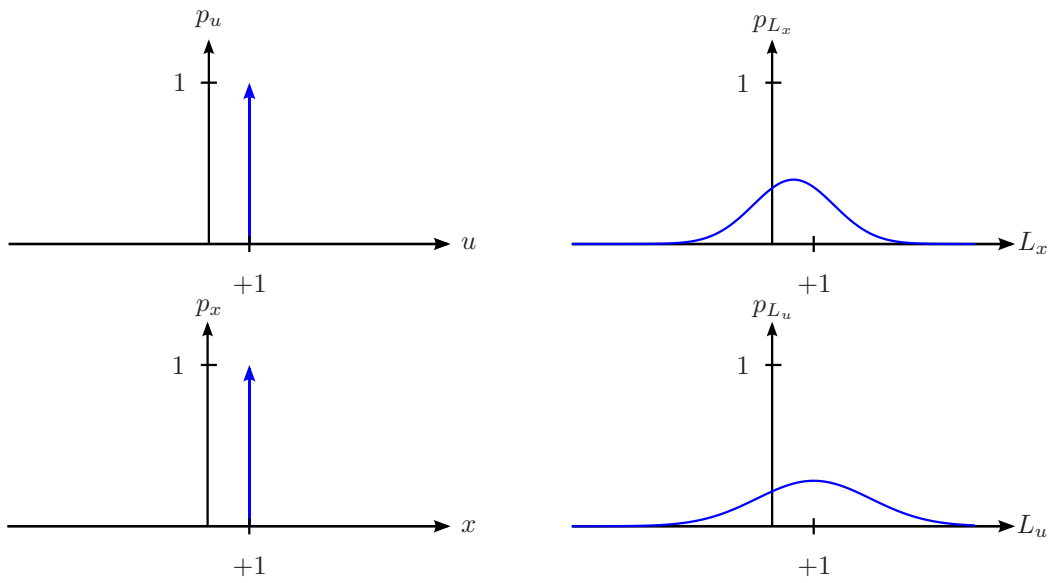


Figure 3.2: Densities of the non-iterative example.

The conditional probability density functions of this example are shown in Figure 3.2 where $d = 2$ and $\sigma^2 = 4$.

In order to be able to perform this analysis, every component has to be described by a *probability density transfer function* that maps a probability density function at one or more inputs to a probability density function at the output. This procedure is know as *density evolution* and was introduced in [RU01]. For a description of how this transfer function can be computed efficiently for blocks that perform decoding of repetition and single parity-check codes we refer to [RU01]. Before we are going to apply this procedure to an iterative system, we present simplifications and approximations in the next sections.

3.1.1 Discretized Density Evolution

To simplify the computation of the involved densities it is common to quantize the probability density function and track the resulting probability mass functions [RSU01]. For the case of a summation of independent random variables (e.g. decoding of a repetition code), the resulting convolution of the associated densities can then be computed using the fast Fourier transformation. This allows an efficient implementation of density evolution. Furthermore, discretized density evolution allows the investigation of the effects of finite precision computations.

3.1.2 Density Modelling

Another way to simplify the analysis is to describe the densities by a model, parametrized with a small set of variables. For many scenarios, the Gaussian distribution is an accurate model for the densities in the system [CRU01, DDP01, SAL06, LM03]. Since a Gaussian distribution is completely characterized by its mean and its variance, the analysis simplifies to tracking the mean/variance tupel instead of density functions. In many cases, the Gaussian distributions also satisfy a symmetry property [RU01], which relates mean and variance. In this case, the Gaussian model is completely described by a single scalar parameter.

The description of the densities using a model and a single scalar parameter is a convenient way to analyze iterative systems. The choice of the parameter is arbitrarily and many different parameters like erasure probability [LMSS01], bit error probability, signal to noise ratio [EGH01], variance of Gaussian distributions, etc. are mentioned in the literature. Some of these parameters have the disadvantage that they are only applicable for a certain message model. In [tB99], it was proposed to track mutual information in order to analyze iterative systems. The resulting transfer functions are called *extrinsic information transfer (EXIT) functions* and the plots of these functions are called *EXIT-charts*.

3.2 Extrinsic Information Transfer Functions

The advantage of tracking mutual information is that it is always possible to compute the mutual information between two random variables regardless of their distribution models. Different distribution models lead to different EXIT functions, but it was shown in [Lan05], that the EXIT function can be upper and lower bounded for specific decoders, and the bounds are tight for certain distributions. It turned out that EXIT functions satisfy some useful properties especially if the distributions and messages involved correspond to a binary erasure channel. We refer to [AKtB04] for a comprehensive description of EXIT charts and their properties.

In order to derive EXIT functions, we use the decoding model shown in Figure 3.3 [AKtB04]. The upper path in this model corresponds to the communication channel and the lower part corresponds to a *virtual* channel, i.e. observations that are not directly observed through the communication channel are modeled as being received over an extrinsic channel. The decoder computes the a-posteriori L-values and the extrinsic L-values (see Section 2.2.2).

EXIT charts are used to track *average mutual information* [AKtB04], where the average is taken over the elements of the corresponding vectors. The average mutual

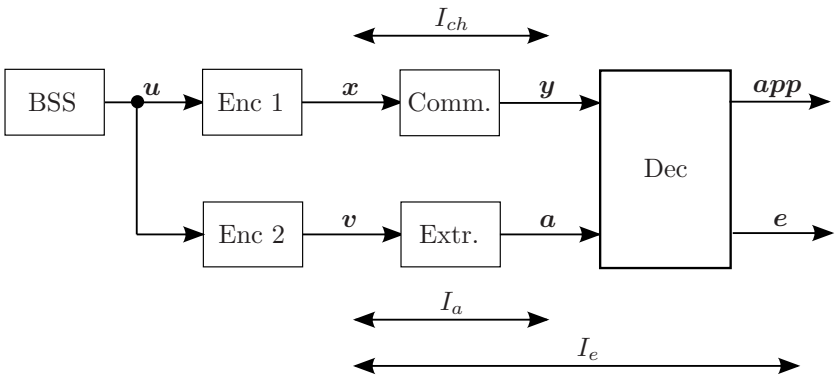


Figure 3.3: Decoding model with communication and extrinsic channel.

information between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as

$$I_{ch} \hat{=} \frac{1}{d} \sum_{i=1}^{d} I(X_i; Y_i), \tag{3.10}$$

where d is the length of the vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$. This average mutual information corresponds to the capacity of the communication channel if the distribution of the elements of $\boldsymbol{X}$ is capacity achieving. The average mutual information between the input and the output of the virtual channel reads

$$I_a \hat{=} \frac{1}{d} \sum_{i=1}^{d} I(V_i; A_i). \tag{3.11}$$

For the output of the decoder, we can define the extrinsic and a-posteriori average mutual information as

$$I_e \hat{=} \frac{1}{d} \sum_{i=1}^{d} I(V_i; E_i), \tag{3.12}$$

$$I_{app} \hat{=} \frac{1}{d} \sum_{i=1}^{d} I(V_i; E_i, A_i). \tag{3.13}$$

In [AKtB04] it is shown that

$$I(V_i; E_i) \leq I(V_i; \boldsymbol{Y} \boldsymbol{A}_{[i]}), \tag{3.14}$$

where the notation $\boldsymbol{A}_{[i]}$ denotes all but the i$^{\text{th}}$ element of the vector $\boldsymbol{A}$. For optimal a-posteriori decoders this inequality is in fact an equality. The consequence of this equality is that an optimal decoder extracts all the information contained in its observations to compute the extrinsic L-value.

An EXIT function plots I_e as a function of I_a parametrized by I_{ch} which is written as

$$I_e = f\left(I_a; I_{ch}\right). \tag{3.15}$$

In order to fully characterize the EXIT function, the assumed distribution model has to be specified, e.g. the EXIT function assuming Gaussian distributions differs from the EXIT function assuming distributions corresponding to a BEC.

In the following sections, we will give an overview of the most important EXIT functions that correspond to elements of iterative decoders and are used in the rest of this work.

3.2.1 Repetition Code

Assume a repetition code of length $d+1$ where one element is transmitted over the communications channel and d elements are observed through the extrinsic channel. This corresponds to a variable node of an LDPC code with degree d. The decoding model is shown in Figure 3.4 and the EXIT functions for $I_{ch} = 0.6$ and $d = 3$ are shown in Figure 3.5. The functions are computed for densities modeled according to a BEC, BSC and BIAWGN channel. As shown in [Lan05], the EXIT function is upper bounded if the distributions are modeled as BEC and lower bounded for a BSC.

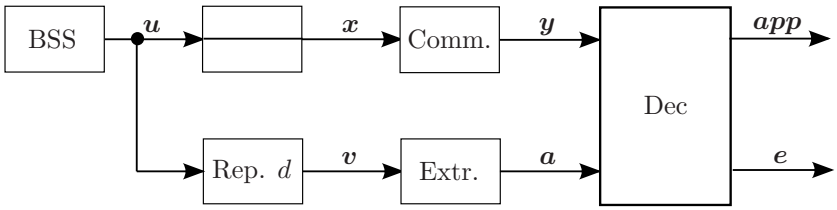


Figure 3.4: Decoding model for repetition code.

3.2.2 Single Parity Check Code

The check nodes of an LDPC code correspond to single parity-check codes, where there is no direct observation from the communications channel available. Therefore, the decoder observes d symbols via the extrinsic channel where d is the length of the single parity-check code. The decoding model and the EXIT functions for this scenario are shown in Figure 3.6 and Figure 3.7, respectively, using $d = 6$. As in the case of a repetition code, the function can be upper and lower bounded [Lan05]. Note that the distributions that achieve the bounds are again distributions modeled

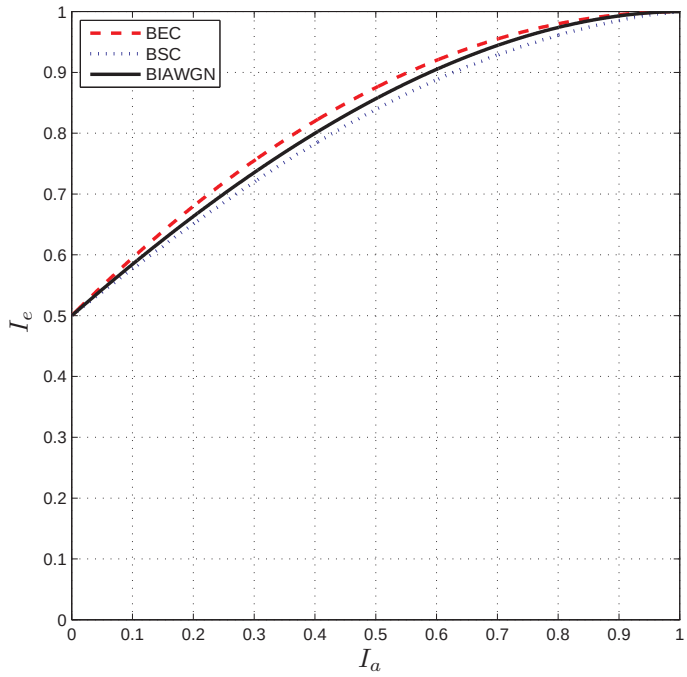


Figure 3.5: EXIT functions of a decoder for a repetition code.

as BEC and BSC, but in the case of a single parity-check decoder, the lower bound is achieved by a BEC and the upper by a BSC.

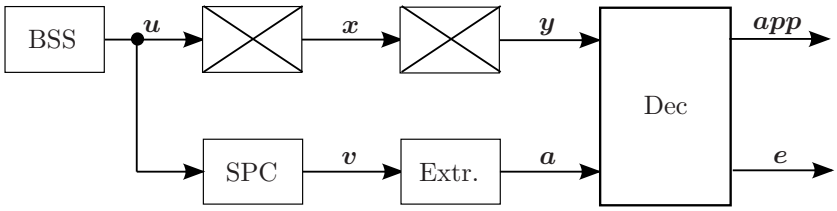


Figure 3.6: Decoding model for a single parity-check code.

3.2.3 Convolutional Code

Convolutional codes as the components of turbo codes are decoded using the BCJR algorithm (see Section 2.2.5). Each decoder accepts L-values of the systematic and parity symbols as well as a-priori L-values of the systematic symbols. The corresponding decoding model is shown in Figure 3.8 and the EXIT function of a recursive systematic convolutional code with generator polynomial $(5, 7)$ is shown in Figure 3.9 for $I_{ch} = 0.4$ using a BIAWGN channel. For convolutional codes, there are no proven bounds as in the case of repetition and single parity-check codes.

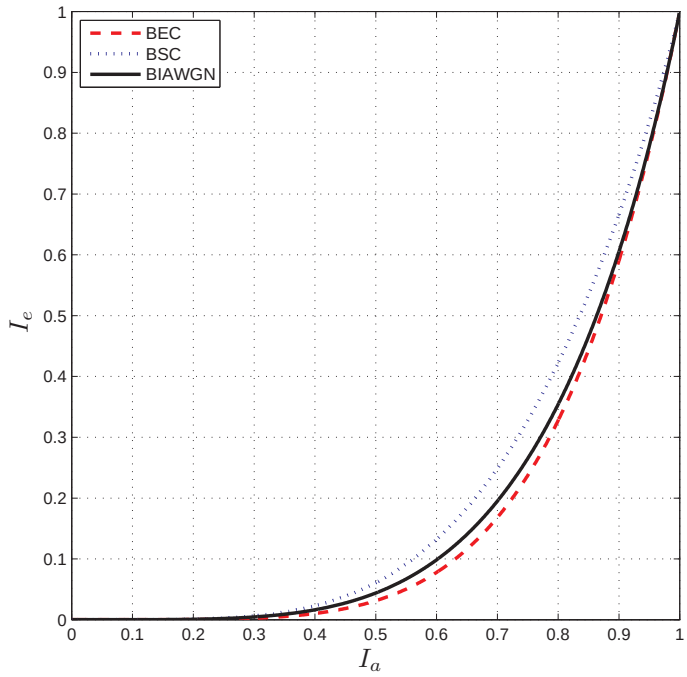


Figure 3.7: EXIT functions of a decoder for a single parity-check code.

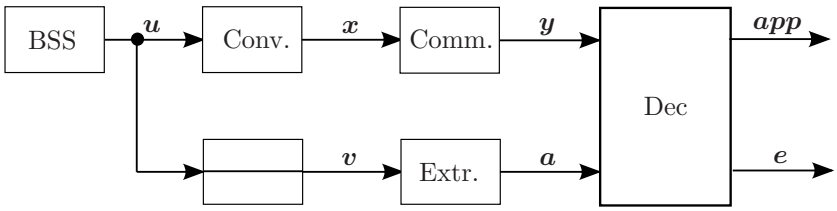


Figure 3.8: Decoding model for convolutional code.

3.3 Iterative Systems

So far, the analysis tools were applied to non-iterative systems. However, the analysis of serial and parallel concatenated decoders as described in Section 2.3 follows the same principle. Density evolution is used to track the probability density functions of the messages that are exchanged between the component decoders. If the probability of error converges to zero when the number of iterations increases, the decoder recovers the transmitted codeword.

EXIT charts can be used as an intuitive way to analyze the iterative decoding process. Since EXIT charts plot extrinsic versus a-priori mutual information, which are the quantities that are exchanged between the component decoders, one can plot the EXIT functions of both component decoders in a single EXIT chart and flip the

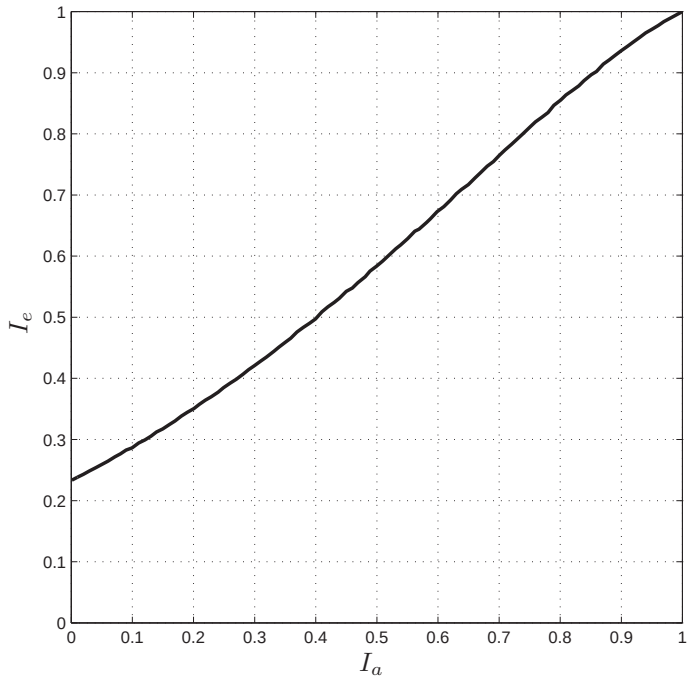


Figure 3.9: EXIT function of a decoder for a convolutional code for the BIAWGN channel.

axis of one of the function corresponding to one decoder. The decoding process is then the *trajectory* between the two EXIT functions. If the two EXIT functions do not intersect, the decoder converges to the top-right corner of the EXIT chart. This corresponds to $I_e = 1$ and therefore, there is no uncertainty about the transmitted codeword leading to error-free decoding.

3.3.1 Turbo Codes

Turbo codes, as described in Section 2.4, are a parallel concatenation of convolution codes, where the component decoders are exchanging information about the common systematic part. Therefore, the EXIT chart of a turbo decoder consists of the EXIT functions of the two convolutional decoders described in Section 3.2.3. The resulting chart is shown in Figure 3.10 for $I_{ch} = 0.4$ and component decoders with generator polynomials $(5, 7)$. The decoding trajectory predicts that the decoder converges to zero error rates after a sufficient number of iterations.

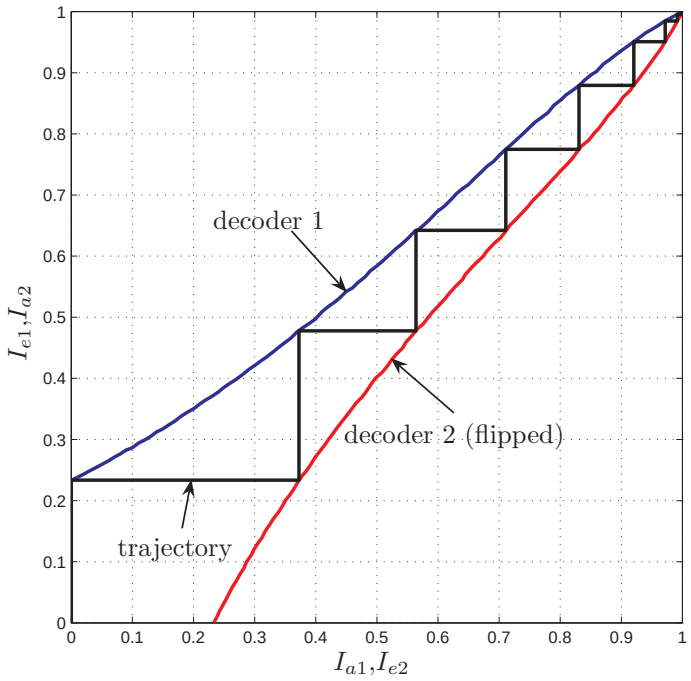


Figure 3.10: EXIT chart of a turbo decoder.

3.3.2 Low-Density Parity-Check Codes

LDPC codes consist of variable nodes, representing repetition codes (Section 3.2.1), and check nodes, representing single parity-check codes (Section 3.2.2). The EXIT chart of an LDPC decoder with variable node degree $d_v = 3$ and check node degree $d_c = 6$ is plotted in Figure 3.11 for $I_{ch} = 0.6$. The decoding trajectory in this case also predicts error-free transmission.

3.4 Conclusions

The analysis of iterative systems is performed by tracking the probability density functions of the involved random variables. Therefore, each component has to be characterized by a probability density transfer function. Modeling the densities and describing them by a single scalar parameter, simplifies the analysis and allows an intuitive representation of the iterative process. However, the presented tools have limitations in practice due to their assumptions:

- *Law of large numbers*
  Tracking probability density functions or statistical quantities like mutual information assumes an infinite block length, i.e. for finite length systems, the

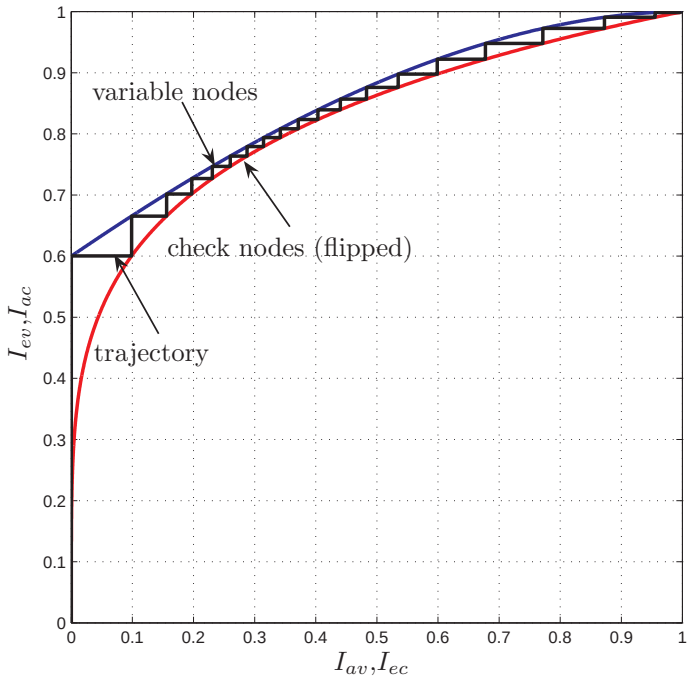


Figure 3.11: EXIT chart of an LDPC decoder.

corresponding empirical quantities scatter around their means.

- *Independence assumption*
  The assumption, that observations from the extrinsic channel are independent from each other is only valid if the system does not introduce correlations, i.e. the associated factor graph [KFL01] is free of cycles. This is true for LDPC codes, if the block length of the code goes to infinity. For finite length systems, observations become correlated which degrades the performance of the decoding algorithm.

Both assumptions state that the analysis is only correct for the asymptotic case of infinite block length. This restriction applies to density evolution as well as to EXIT charts. For EXIT charts, another restriction applies:

- *Accuracy of the distribution model*
  The EXIT chart analysis assumes that all the involved random variables are distributed according to a given model. This model is then described by a single scalar parameter, i.e. mutual information. In most cases (especially in the most important case of an assumed Gaussian distributions), the true probability densities are not exactly Gaussian (due to nonlinearities in the decoder). Therefore, EXIT charts are only exact if the distribution model

describes the true message distributions exactly. Examples of such scenarios are decoding of LDPC codes for a binary erasure channel and binary messages passing decoders which will be discussed in Chapter 5.

4 Simplifying Component Decoders

The component decoders of concatenated coding schemes operate on simple codes like repetition codes, single parity-check codes and convolutional codes. Although decoding of these component codes is relatively simple, there are still challenging tasks when implementing those decoders. In this chapter, we will concentrate on the simplification of the component decoders of LDPC codes, i.e. repetition and single parity-check codes.

LDPC codes are decoded using the sum-product algorithm (SPA) [KFL01, Mac99] and the messages passed between the nodes are L-values since they offer a higher numerical stability than probabilities. Decoding repetition and single parity-check codes in the L-value domain was described in Section 2.2.3 and Section 2.2.4, respectively. The principle of decoding LDPC codes is shown in Figure 4.1.

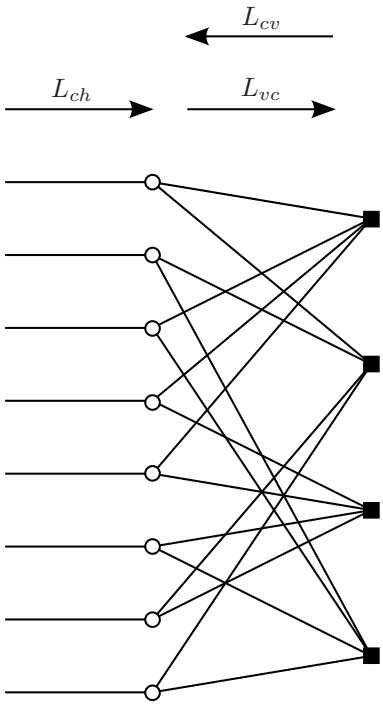


Figure 4.1: Decoding LDPC codes.

The L-values arriving from the channel are denoted as L_{ch}, the message from

variable to check nodes as L_{vc} and the messages from check to variable nodes as L_{cv}. Let N denote the number of variable nodes and M denote the number of check nodes corresponding to the number of columns and rows in the parity-check matrix. The set of edges connected to variable node n are denoted by $\mathcal{E}_v(n)$ and the set of edges connected to check node m by $\mathcal{E}_c(m)$. A message sent out on an edge $j = 1 \dots d_v$ of variable node n of degree d_v is the sum of all L-values at the other incoming edges and the channel L-value of variable node n

$$L_{vc,j} = L_{ch,n} + \sum_{i \in \mathcal{E}_v(n); i \neq j} L_{cv,i}. \tag{4.1}$$

A message sent out on edge $j = 1 \dots d_c$ of check node m of degree d_c is computed as

$$L_{cv,j} = 2 \tanh^{-1} \left(\prod_{i \in \mathcal{E}_c(m); i \neq j} \tanh \frac{L_{vc,i}}{2} \right). \tag{4.2}$$

The operations at the variable node decoder are easy to implement. However, the implementation of the check node decoder is less trivial because of the following reasons:

- *Nonlinear functions*
  The implementation of the hyperbolic tangent and its inverse is usually realized with a lookup table. In order to be able to compute many check nodes in parallel (which is required for high-throughput architectures), many lookup tables have to be implemented which increases the amount of required resources significantly.

- *Multiplication*
  The algorithm consists of real-valued multiplications which require many resources on dedicated hardware.

- *Numerical stability*
  When the magnitude of the argument of the inverse hyperbolic tangent is close to one, the inverse operation becomes numerically unstable.

All those disadvantages can be avoided by an approximation of the decoding algorithm for single parity-check codes as described in the next section.

4.1 Approximation of SPC Decoding

Decoding of single parity-check codes can be approximated by using the min-sum algorithm (MSA) [KFL01, CF02b, CF02a] which approximates (4.2) by

$$L_{cv,j} \approx Z_{cv,j} = \min_{i \in \mathcal{E}_c(m); i \neq j} |L_{vc,i}| \cdot \prod_{i \in \mathcal{E}_c(m); i \neq j} \operatorname{sgn}(L_{vc,i}). \tag{4.3}$$

Using this approximation, the algorithm becomes numerically stable, there are no nonlinear functions involved that require a lookup table and a real-valued multiplication is not required anymore (the multiplication in (4.3) can be realized using exclusive-or operations).

While the min-sum algorithm not only lowers the implementation complexity, it also leads to a decoding algorithm where a multiplicative factor at the input, i.e. scaling all channel L-values by a constant, results only in a scaling of all messages passed in the graph by this factor. A consequence of this property is that, for the BIAWGN channel, it is not necessary to estimate the noise variance of the channel, and the channel observations can be used directly for the decoding process, since the computation of the L-values is just a scaling as described in Section 2.1.4.

However, the advantage of reduced complexity and elimination of the noise estimation, has to be paid by a degradation of the decoding threshold of approximately 0.5 to 1.0dB for regular codes. For irregular codes, the loss in performance is even higher. It has been recognized by several authors that this performance degradation can be partly compensated by transforming the check node output of the min-sum algorithm. We call those methods *post-processing* and the transformations that have been applied heuristically are either scaling (normalized belief propagation) or subtracting an offset (offset belief propagation) [CF02a]. The correction terms that result in the lowest decoding threshold have been found using density evolution and it has been shown that the thresholds of the resulting algorithms are close to that of the sum-product algorithm. However, when applying these methods to irregular LDPC codes, it has been observed [LS06a] that these codes exhibit an error floor which is not caused by the properties of the code, but by the decoding algorithm.

The aim of this chapter is to put these heuristic post-processing methods on a solid scientific basis by providing a theoretical framework for post-processing the check node decoder output in general. This theoretical framework allows us not only to explain normalized and offset belief propagation, but also to find the correction terms for both algorithms analytically. Furthermore, the insight gained by this analysis allows us to extend post-processing to irregular LDPC codes, achieving a good decoding threshold without a noticeable error floor.

Before, we derive the post-processing function for single parity-check codes, we use EXIT charts to motivate post-processing.

4.1.1 EXIT Charts and Density Evolution

It is interesting to note the complementary insights obtained by analyzing the min-sum algorithm with density evolution and with EXIT charts. Density evolution gives an exact measure of the performance achieved by the algorithm but gives no indication that this performance can be improved by post-processing the messages.

Since the EXIT chart analysis treats both component decoders separately, it is not influenced by a wrong interpretation of the messages when connecting these component decoders. Therefore, EXIT charts can be used to obtain an upper bound on the performance that is achievable by proper means of post-processing[1].

In Figure 4.2, we show an EXIT chart of a regular LDPC code with variable node degree $d_v = 3$ and check node degree $d_c = 6$ where the check node EXIT function was computed using the min-sum algorithm. The chart is plotted for a signal to noise ratio that is just above the decoding threshold of the min-sum algorithm, which was evaluated using density evolution. In addition to the EXIT functions, the decoding trajectory (also obtained using density evolution), is shown in this figure. Comparing this analysis with Figure 3.11, it can be recognized that in the case of the min-sum algorithm, the decoding trajectory is not in agreement with the EXIT chart analysis. The trajectory follows the check node curve but does not fit to the variable node curve. This leads to the suggestion that the output of the check node decoder is not correctly processed by the variable nodes.

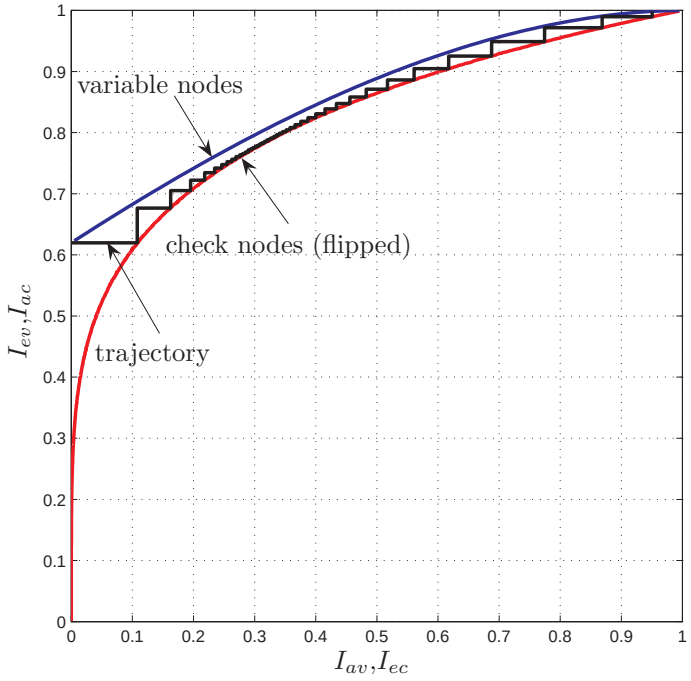


Figure 4.2: EXIT chart and decoding trajectory for the min-sum algorithm and $E_b/N_0 = 1.72$dB ($d_v = 3$, $d_c = 6$).

In Table 4.1, we compare the decoding thresholds (E_b/N_0 in dB) of the sum-

[1]Note that this bound is not exact, since the EXIT chart analysis relies on approximations as discussed in Section 3.4.

	$d_v = 3,\ d_c = 6$		$d_v = 4,\ d_c = 8$		$d_v = 5,\ d_c = 10$	
	SPA	MSA	SPA	MSA	SPA	MSA
DDE	1.10	1.70	1.54	2.50	2.01	3.09
EXIT	1.13	1.19	1.57	1.65	2.02	2.11

Table 4.1: Thresholds ($\frac{E_b}{N_0}$ in dB) obtained using discretized density evolution and EXIT charts.

product algorithm and the min-sum algorithm that where obtained by discretized density evolution (DDE) and EXIT chart analysis.

These results show that for the sum-product algorithm, the thresholds obtained with the two methods are close to each other (the difference is due to the Gaussian assumption in the EXIT chart analysis). However, for the min-sum algorithm, the prediction from the EXIT chart analysis is significantly better than the threshold obtained by DDE. This motivates us to post-process the messages generated by the min-sum algorithm in order to improve its performance.

4.2 Post-processing

First, we consider the case of computing an outgoing message of a check node using the sum-product algorithm, where the term *message* denotes a log likelihood ratio. Given that d_c binary variables $X_1, \ldots, X_{d_c}$ satisfy a parity-check equation, we aim to compute the L-value of one binary variable given the L-values of the $d_c - 1$ other variables, where the received L-values are the outputs of binary input, symmetric, memoryless channels with capacity $I_{ac,i}$[2]. Without loss of generality we will compute L_{d_c} as a function of $L_1, \ldots, L_{d_c-1}$. It is well known that the solution for this problem is given by (see Section 2.2.4)

$$L\left(L_1, \ldots, L_{d_c-1} | X_{d_c}\right) = 2 \tanh^{-1} \prod_{i=1}^{d_c-1} \tanh \frac{L_i}{2}. \tag{4.4}$$

For the min-sum approximation, we can state a similar problem depicted in Figure 4.3. After the min-sum algorithm, we have to compute the L-value of the desired variable given the output of the check node approximation denoted as

$$Z\left(L_1, \ldots, L_{d_c-1}\right) = \min_{i=1}^{d_c-1} |L_i| \cdot \prod_{i=1}^{d_c-1} \operatorname{sgn}\left(L_i\right). \tag{4.5}$$

[2]Strictly speaking we are interested in the symbol-wise mutual information between the input and the output of the channel, but since we will assume equally likely input symbols which maximize the mutual information, this is equivalent to the capacity of the channel.

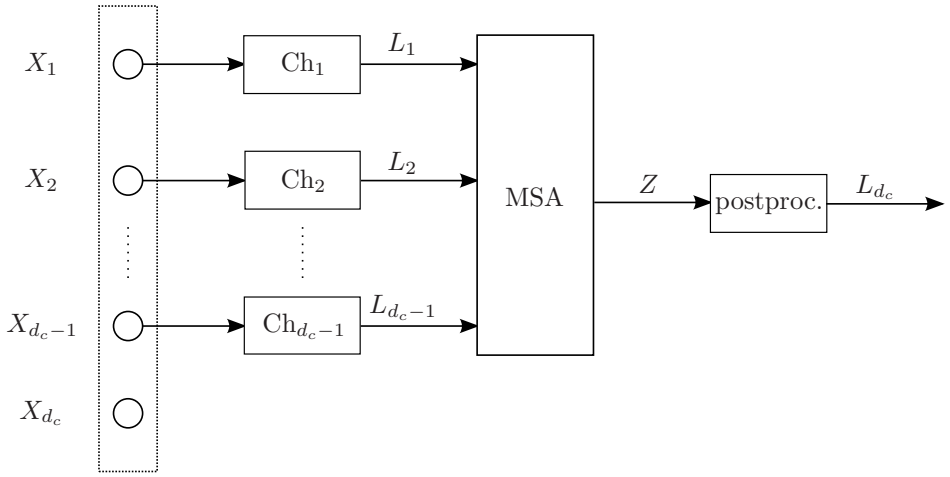


Figure 4.3: Model for check node operation using the min-sum algorithm.

We are still able to analytically compute the L-value

$$L(z|X_{d_c}) = \log \frac{p(Z = z|X_{d_c} = +1)}{p(Z = z|X_{d_c} = -1)} \tag{4.6}$$

as a function of z, using the conditional probability density function

$$\begin{aligned} p(Z = z|X_{d_c} = x_{d_c}) &= \frac{\partial}{\partial z} Pr(Z < z|X_{d_c} = x_{d_c}) \\ &= \frac{1}{2} \sum_{j=1}^{d_c-1} \Bigg\{ (q_j(z) + q_j(-z)) \prod_{i=1;i\neq j}^{d_c-1} \gamma_{i+}(z) \\ &\qquad + x_{d_c}(q_j(z) - q_j(-z)) \prod_{i=1;i\neq j}^{d_c-1} \gamma_{i-}(z) \Bigg\}. \end{aligned} \tag{4.7}$$

The derivation of this function and the definition of q_j, γ_{i-} and γ_{i+} can be found in Appendix A.

Although the L-value of the desired message can be computed analytically, the computation requires the knowledge of the distributions of all incoming messages. In the next section we will show how these requirements can be relaxed in order to obtain practical post-processing functions.

4.2.1 Gaussian Approximation

For regular LDPC codes, the probability density functions of all messages at the input of a check node are identical. For irregular codes this is no longer true, since

the densities depend on the degrees of the variable nodes they originated from. However, motivated by the accuracy of EXIT chart analysis of irregular LDPC codes, where all the densities are modeled as being Gaussian, we will apply the Gaussian assumption for regular and irregular LDPC codes.

The consequence is that we model the probability density functions at the input of the check nodes as identical Gaussian distributions with mean and variance depending only on the a-priori mutual information [RU01] as

$$\sigma_L = J^{-1}(I_{ac}), \tag{4.8}$$

$$\mu_L = \frac{\sigma_L^2}{2}. \tag{4.9}$$

Applying this approximation, we obtain a post-processing function that is parameterized by the two scalar quantities d_c and I_{ac}.

$$L(z|X_{d_c}) = f(z; d_c, I_{ac}). \tag{4.10}$$

Due to the assumption that all incoming messages have the same distribution, we can simplify the post-processing function to

$$\begin{aligned} L(z|X_{d_c}) &= \log \frac{[q(z)+q(-z)]\gamma_+(z)^{d_c-2} + [q(z)-q(-z)]\gamma_-(z)^{d_c-2}}{[q(z)+q(-z)]\gamma_+(z)^{d_c-2} - [q(z)-q(-z)]\gamma_-(z)^{d_c-2}} \\ &= \log \frac{\left(\frac{\gamma_+(z)}{\gamma_-(z)}\right)^{d_c-2} + \frac{q(z)-q(-z)}{q(z)+q(-z)}}{\left(\frac{\gamma_+(z)}{\gamma_-(z)}\right)^{d_c-2} - \frac{q(z)-q(-z)}{q(z)+q(-z)}}. \end{aligned} \tag{4.11}$$

Applying the Gaussian assumption

$$q(z) = \frac{1}{\sqrt{2\pi}\sigma_L} e^{-\frac{(z-\mu_L)^2}{2\sigma_L^2}}, \tag{4.12}$$

allows us to rewrite

$$\frac{q(z)-q(-z)}{q(z)+q(-z)} = \tanh\frac{z}{2}, \tag{4.13}$$

leading to

$$L(z|X_{d_c}) = \log \frac{\left(\frac{\gamma_+(z)}{\gamma_-(z)}\right)^{d_c-2} + \tanh\frac{z}{2}}{\left(\frac{\gamma_+(z)}{\gamma_-(z)}\right)^{d_c-2} - \tanh\frac{z}{2}}. \tag{4.14}$$

This post-processing function is shown in Figure 4.4 for $d_c = 6$ and I_{ac} ranging from 0.0 to 1.0. It can be observed that the post-processing function becomes an identity function if I_{ac} approaches one. For small values of I_{ac} the post-processing

function becomes more nonlinear. In Figure 4.5 we show the dependency on the check node degree d_c for fixed a-priori information $I_{ac} = 0.5$. For higher check node degrees, the post-processing function becomes more nonlinear, while it becomes the identity function for $d_c = 2$. This is obvious, since a check node of degree two just passes the incoming message on one edge to the output at the other edge. From these results one can conclude that post-processing is necessary for small values of I_{ac} and for high check node degree.

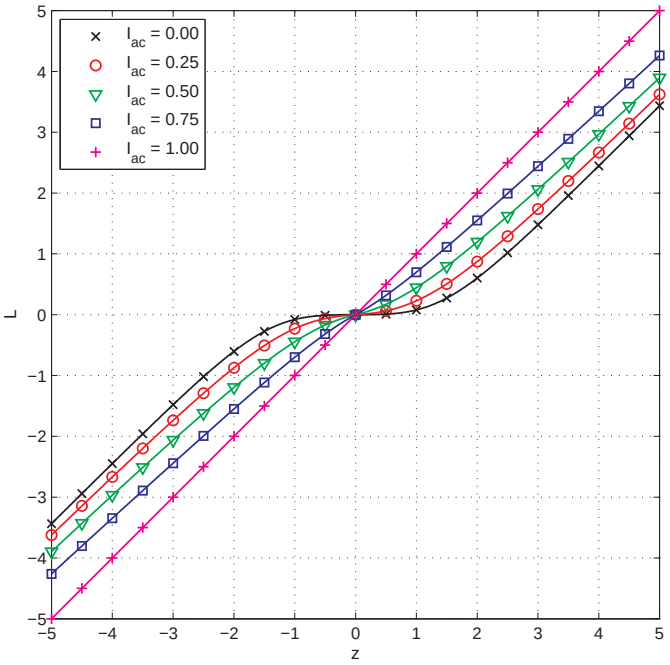


Figure 4.4: Dependency of the post-processing function on the a-priori information for $d_c = 6$.

4.2.2 Linear Approximation

The resulting post-processing function is nonlinear and the dependency on the mutual information requires that it has to be recomputed for every iteration. In [LS04, LS06a, CF02b, CF02a], this function is approximated by a linear function, where the approximation takes the probability density function of the minimum Z into account. We define $\alpha(d_c, I_{ac})$ as the scaling factor which minimizes the expected squared error as

$$\alpha(d_c, I_{ac}) \hat{=} \underset{\tilde{\alpha}}{\operatorname{argmin}} \int_{-\infty}^{\infty} \Big[f(z; d_c, I_{ac}) - \tilde{\alpha} z \Big]^2 \cdot p(Z = z; d_c, I_{ac}) dz, \tag{4.15}$$

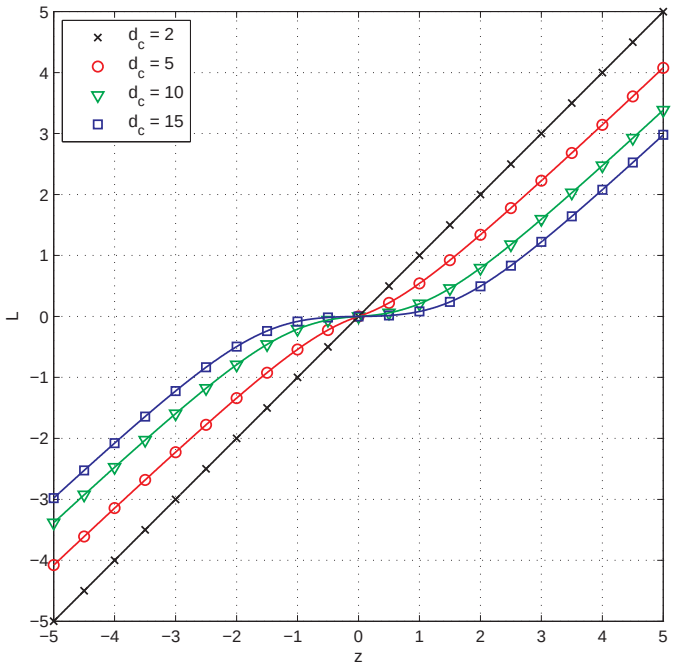


Figure 4.5: Dependency of the post-processing function on the check node degree for $I_{ac} = 0.5$.

where $p(Z = z; d_c, I_{ac})$ denotes the probability density function of the outgoing messages of the check node approximation under the assumption of Gaussian distributed incoming messages. Using this linear approximation, (4.10) becomes

$$L(z|X_{d_c}) = \alpha(d_c, I_{ac}) \cdot z. \tag{4.16}$$

4.2.3 Offset Approximation

Another way of achieving a low complexity post-processing function is to approximate the nonlinear function with an offset function [CF02b, CF02a, LS06a] as

$$L(z|X_{d_c}) = \operatorname{sgn}(z) \cdot \max\left(|z| - \beta(d_c, I_{ac}), 0\right). \tag{4.17}$$

The offset term β is defined as

$$\beta(d_c, I_{ac}) \hat{=} \underset{\tilde{\beta}}{\operatorname{argmin}} \int_{-\infty}^{\infty} \left[f(z; d_c, I_{ac}) - \operatorname{sgn}(z) \cdot \max\left(|z| - \tilde{\beta}, 0\right) \right]^2 \cdot p(Z = z; d_c, I_{ac}) dz. \tag{4.18}$$

This approximation is often preferred in hardware, since subtractions are easier to implement than multiplications and furthermore, when using quantized messages,

subtraction of an offset does not cause additional quantization effects in contrast to a multiplicative scaling. Note that in the case of an offset approximation, the value of the offset term has to be scaled if all the messages passed in the graph are scaled, i.e. if noise estimation is not applied and direct channel observations are used for decoding.

The offset approximation can be combined with the linear approximation of the previous section. However, the resulting gain in performance is negligible and therefore we will not consider this case.

4.3 Regular LDPC Codes

In this section, we investigate how the post-processing function derived in the previous section can be simplified for regular LDPC codes.

4.3.1 Constant Linear or Offset Post-Processing

After applying the linear or offset approximation, the correction terms α and β still depend on I_{ac} which requires a new value for every iteration in the decoding process. We can further simplify the post-processing function by choosing α or β for I_{ac} where it is most crucial that the operations at the variable nodes are optimal, i.e. where the tunnel between the EXIT curves is most narrow. For regular LDPC codes, where the curves touch each other only at a single point in general, this results in a scaling factor or an offset term that can be kept constant over the iterations [LS04].

For the example of a regular LDPC code with $d_v = 3$ and $d_c = 6$, we find that the EXIT functions have their narrowest gap at $I_{ac} = 0.76$ as shown in Figure 4.6. In the context of our computation, we obtain $\alpha = 0.81$ which corresponds to the results in [CF02a][3]. For the constant offset, we computed a value of $\beta = 0.41$.

4.3.2 Simulation Results

The bit error rate simulation of a regular LDPC code with $d_v = 3$ and $d_c = 6$ is shown in Figure 4.7. The code used for this example has a block length of $N = 10^5$ and a rate $R = 0.5$ and was decoded using the sum-product, the min-sum algorithm, constant linear post-processing and constant offset post-processing. For the constant post-processing, we used the values obtained in the previous section. At a bit error rate of 10^{-4}, the min-sum decoder has a gap to the sum-product decoder of approximately 0.6dB. Using constant linear post-processing or constant offset post-processing, the gap can be lowered to approximately 0.1dB. Since this

[3] Note that the scaling factor α defined in [CF02a] is the inverse of our definition.

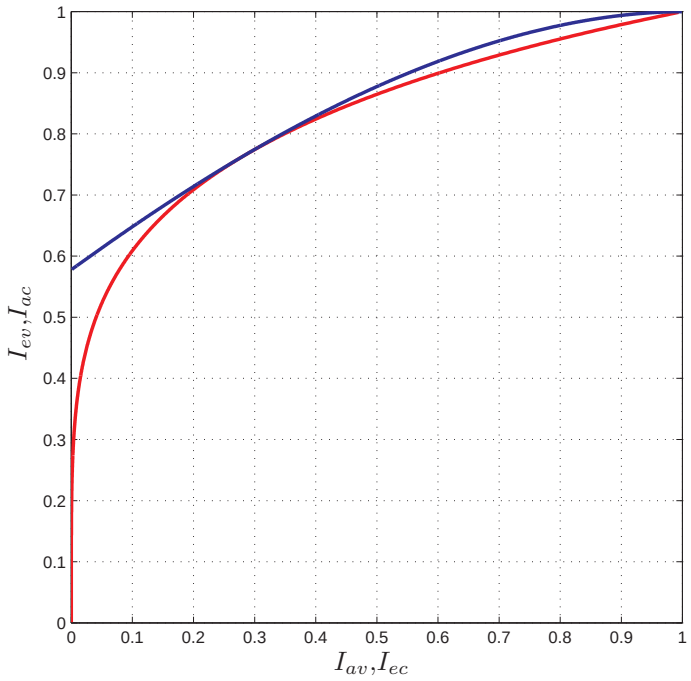


Figure 4.6: Constant post-processing for critical point.

performance is close to the optimal decoder, it is not worth to spend more complexity for post-processing (i.e. using nonlinear, or non-constant post-processing functions).

4.4 Irregular LDPC Codes

As mentioned in Section 4.2.1, we apply a Gaussian approximation also to the case of irregular LDPC codes. Since I_{ac} is not the same for every channel, it now denotes the average mutual information with respect to the variable node degree distribution (edge perspective). Therefore, also the linear and offset approximation of Section 4.2.2 and 4.2.3 extend naturally to irregular LDPC codes. However, using a constant post-processing function as described in Section 4.3.1, results in a tradeoff between good decoding threshold and low error-floor as shown in [LS06a]. If the code is not check-regular, there exists an individual post-processing function (or one of its approximations) for every check node degree.

In the following sections, we will show how post-processing has to be adapted for irregular LDPC codes. As an example, we use an irregular LDPC code that was taken from [Amr]. The code is check regular with $d_c = 9$ and the nonzero coefficients of the variable node distribution (see Section 2.5) are shown in Table 4.2. The resulting code rate is $R = 0.5$. For the code construction we set the block length to

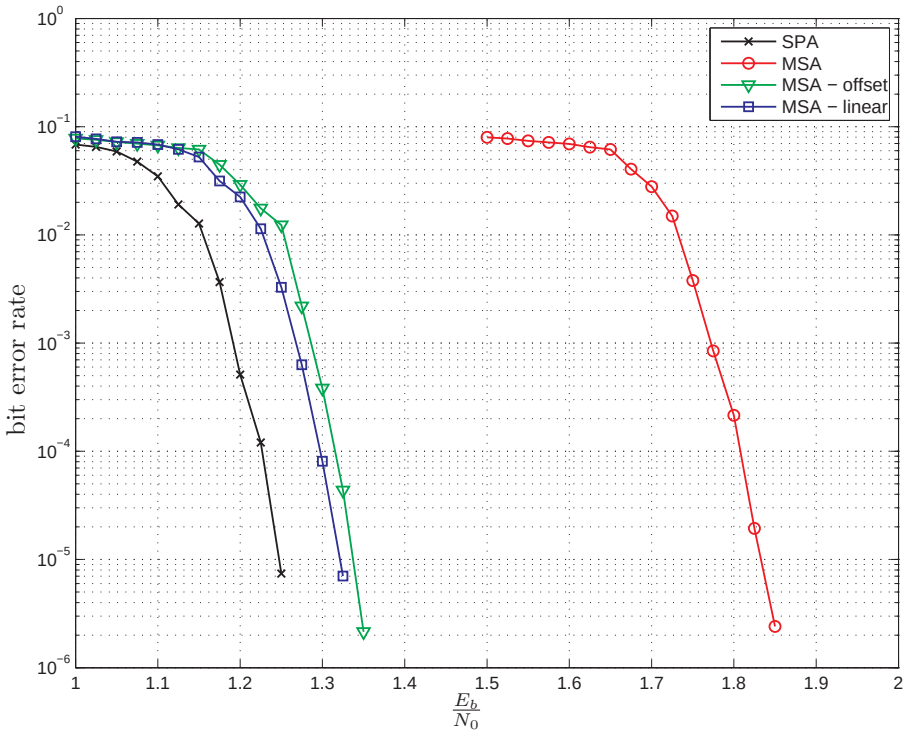


Figure 4.7: Sum-product and min-sum algorithm for regular LDPC code of rate 0.5.

$N = 10^5$ in order to minimize the effects that are caused by cycles in the graph.

4.4.1 Sequence of Correction Terms

Since the EXIT functions of variable and check node decoder touch each other almost everywhere if the code is designed in order to achieve capacity, it is not possible to apply a constant post-processing function for every iteration. One way to overcome this, is to vary the post-processing function over the number of iterations resulting in a sequence of scaling factors or a sequence of offset values. These sequences can be found by predicting the trajectory in the EXIT chart just above the threshold and computing a correction term for every iteration. The correction terms can then be obtained using density evolution as in [CF02a], or by using the analytical derivation ((4.15) and (4.18)). An example of a sequence of scaling factors for 100 iterations is shown in Figure 4.8. It can be observed, that this sequence is increasing and it converges to 1.0, i.e. no post-processing, which is due to the fact that the decoder is converging to $I_{ac} = 1.0$ where post-processing becomes an identity function (see Section 4.2.1).

i	λ_i
2	0.1111081
3	0.4125882
10	0.0003862
12	0.2647241
29	0.0000303
30	0.2111631

Table 4.2: Degree distribution of the irregular example code.

4.4.2 Constant Nonlinear Post-Processing

When implementing an LDPC decoder in hardware, a scaling factor can easily be applied by using a lookup table and thus avoiding the need for a multiplication. However, the application of a sequence of scaling factors would require either a new lookup table for every iteration, or a multiplication unit. Both options are not convenient for small and fast architectures. Therefore, we aim to apply a post-processing function that can be kept constant over the iterations without loss in performance, i.e. degradation of the threshold.

We propose to apply a *universal*, nonlinear post-processing function that is derived as a combination of post-processing functions weighted with the probability density function of the output of the min-sum approximation. We use a heuristic method to derive this universal post-processing function, which is computed as the weighted average between the minimum and the maximum I_{ac} at the check node input as

$$f_u(z; dc) \hat{=} \int_{I_{ac,min}}^{I_{ac,max}} f(z; d_c, I_{ac}) \cdot p(Z = z; d_c, I_{ac}) dI_{ac}. \tag{4.19}$$

In Figure 4.9, we show the universal post-processing function obtained using (4.19), where we set $I_{ac,min} = 0.517$ (which is the threshold of the example code) and $I_{ac,max} = 1.0$. For comparison, we also show the post-processing functions for I_{ac} at 0.517 and 1.0. It can be observed that for small magnitudes of Z, the universal function is close to the function for $I_{ac} = 0.517$ and for large magnitudes the universal function gets close to the post-processing function for $I_{ac} = 1.0$ (which is the identity function).

4.4.3 Simulation Results

Figure 4.10 shows bit error rate simulations for the irregular example code, using a maximum of 100 iterations. We used post-processing with varying scaling factors

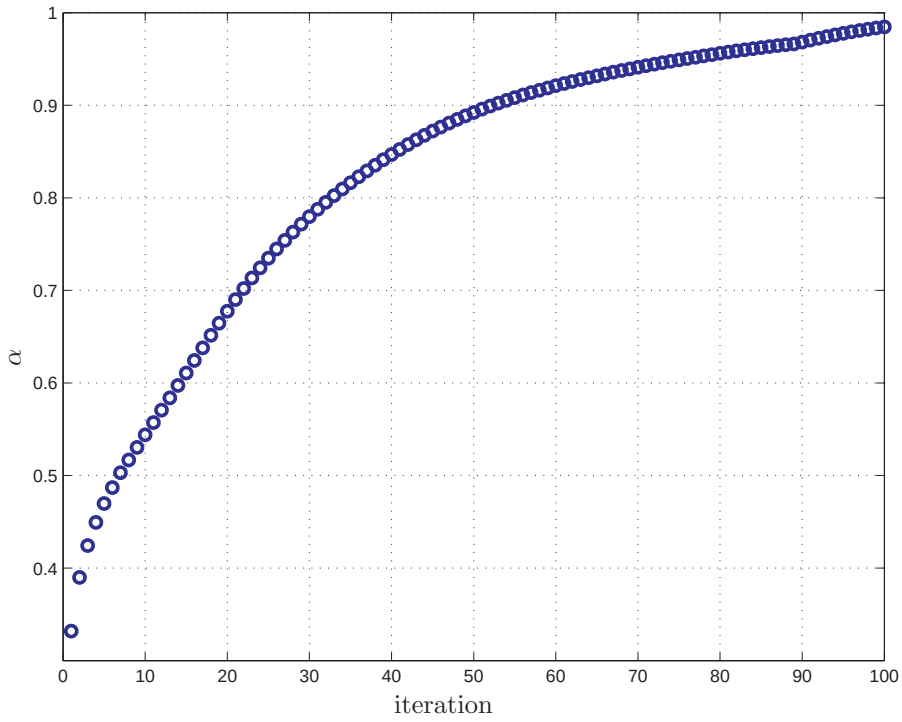


Figure 4.8: Sequence of scaling factors for the irregular example code.

shown in Figure 4.8, that where obtained as described in Section 4.4.1 and constant nonlinear post-processing as described in Section 4.4.2. For comparison, we included the curves for the sum-product algorithm, the min-sum algorithm without post-processing and the min-sum algorithm with constant scaling factors. It can be observed that the universal post-processing function has the smallest gap to the optimal sum-product algorithm and that the performance is very close to that of varying scaling factors. Compared to the min-sum algorithm without post-processing, we notice an improvement of approximately 1.0dB. For the case of constant scaling factors (as in the case of regular LDPC codes), we observe a trade-off between error floor and decoding threshold. Small scaling factors lead to a good decoding threshold but suffer from a high error floor. This is because small scaling factors cause the decoder to get stuck after a few iterations and therefore, the trajectory does not reach the top right corner of the EXIT chart. On the other hand, using large scaling factors causes the decoder to perform sub-optimal in the first iterations, leading to a higher decoding threshold.

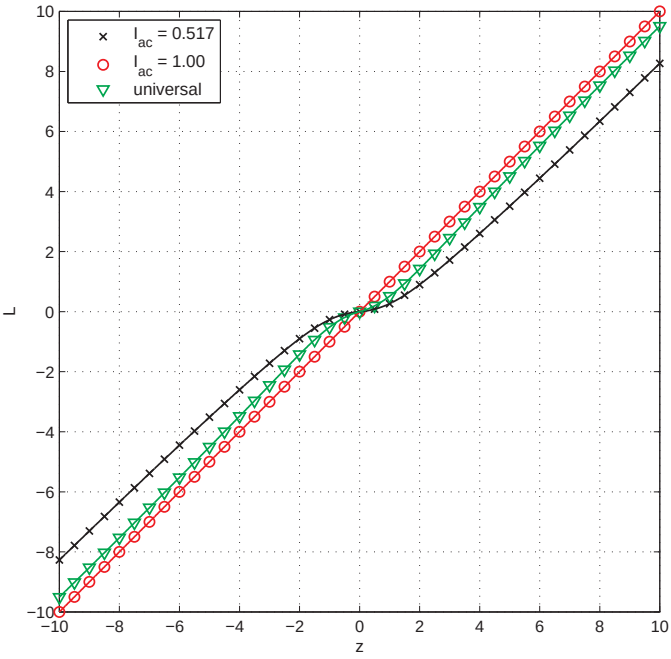


Figure 4.9: Universal post-processing function for $d_c = 9$.

4.5 Conclusions

We showed how the min-sum algorithm for decoding LDPC codes can be improved by means of post-processing, by re-interpreting the minimum as an observation and computing the a-posteriori L-value of the code digits given this observation.

The resulting post-processing function is nonlinear and has to be changed for every iteration in general. However, it is sufficient to approximate this post-processing function by either a linear or an offset function. Furthermore, we showed that for regular codes it is sufficient to optimize the post-processing function for a single point in the EXIT chart, allowing us to keep it constant for every iteration.

For irregular LDPC codes, we showed that using a constant approximation of the post-processing function results in a trade-off between good decoding threshold and low error floor. To achieve decoding with good threshold and low error floor one has to use either a sequence of linear post-processing functions or nonlinear post-processing. Using these techniques decreases the gap between optimal sum-product decoding and sub-optimal min-sum decoding significantly.

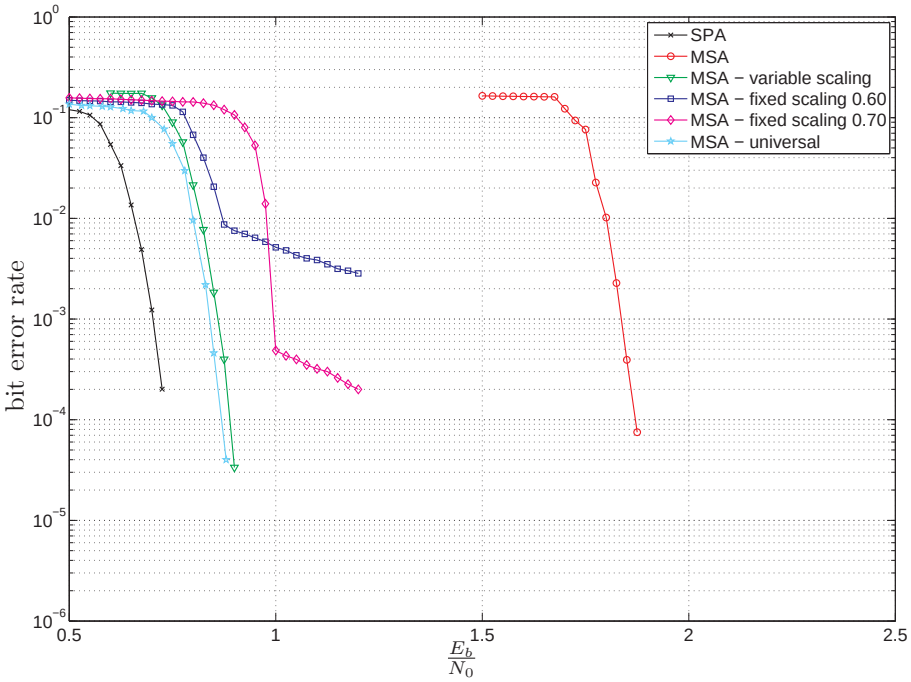


Figure 4.10: Bit error rates for irregular code with and without post-processing.

5 Binary Message-Passing Decoders

When Gallager introduced low-density parity-check codes [Gal62, Gal63], he also presented binary message-passing decoding algorithms, that exchange only binary messages between the variable and check node decoder—called Gallager A and Gallager B algorithm. The advantages of these algorithms are the reduced memory requirements and the low complexity implementation, especially of the check node decoder, making them promising candidates for high-speed applications, e.g. optical transmission systems [ITU04]. However, these advantages come with the cost of a significant loss in performance.

In this chapter, we apply extrinsic information transfer (EXIT) charts [AKtB04] in the analysis of binary message-passing algorithms. For binary messages, the mutual information describes the probability densities of the messages uniquely. In this case, the EXIT functions are exact. In contrast, this is not the case if the messages are approximated by Gaussian distributions. Furthermore, the EXIT functions for binary message-passing algorithms can be derived analytically, avoiding the need of Monte-Carlo simulations.

Binary message-passing algorithms were studied in [AK05] where the authors proved that optimal algorithms have to satisfy certain symmetry and isotropy conditions. In contrast to majority based decision rules, we assume that the variable node decoder converts all incoming messages to L-values, performs decoding in the L-value domain and applies a hard decision on the result. In Section 5.5 we will show how majority decision rules can be derived from these algorithms. This general approach assures that the symmetry and isotropy conditions are satisfied and we are able to extend the algorithms for systems where the channel provides more information than hard decisions, while the variable and check node decoder still exchange binary messages only. This reduces the gap between optimum decoding and binary message-passing decoding, while still keeping the complexity low.

5.1 Preliminaries

For binary message-passing decoders, the extrinsic channel [AKtB04] of the variable and check node decoder is represented as a binary symmetric channel (BSC) with crossover probability ϵ which we assume to be smaller than or equal to 0.5. Since

there is a one-to-one mapping between mutual information and crossover probability for the BSC, we can equivalently describe those channels using their capacities

$$C_{\mathrm{BSC}} = 1 - h_b(\epsilon), \tag{5.1}$$

where $h_b(\cdot)$ denotes the binary entropy function.

The *reliability* D associated with a BSC is defined as

$$D \hat{=} \log \frac{1-\epsilon}{\epsilon} \geq 0, \tag{5.2}$$

which allows to write the computation of the L-value as

$$L = y \cdot D, \tag{5.3}$$

where y denotes the output of a BSC which takes on values from $\{+1, -1\}$.

5.2 EXIT Functions of Component Decoders

A check node of degree d_c of a binary message-passing algorithm computes the output as the modulo-2 sum of the other $d_c - 1$ inputs. Let $\epsilon_{ac} = h_b^{-1}(1 - I_{ac})$ denote the a-priori crossover probability at the input of the check node. Then the crossover probability at the output reads [Gal63, Lemma 4.1]

$$\epsilon_{ec} = f_c(\epsilon_{ac}; d_c) = \frac{1 - (1 - 2\epsilon_{ac})^{d_c - 1}}{2}, \tag{5.4}$$

where f_c is the EXIT function of a check node parametrized by d_c. Using (5.4) and (5.1) leads to $I_{ec} = 1 - h_b(\epsilon_{ec})$. The inverse of the EXIT function in (5.4) takes the form

$$\epsilon_{ac} = f_c^{-1}(\epsilon_{ec}; d_c) = \frac{1 - (1 - 2\epsilon_{ec})^{\frac{1}{d_c - 1}}}{2}. \tag{5.5}$$

An example of an EXIT function of a check node decoder with $d_c = 6$ is shown in Figure 5.1.

For a variable node of degree d_v, the computation of the outgoing messages consists of the summation of all other $d_v - 1$ incoming messages and the channel message in the L-value domain [RU01]. In order to be able to perform this summation, all messages are converted to L-values using (5.2) and (5.3), assuming that the variable node decoder knows the parameters of the communication and the extrinsic channel. We show in Section 5.4 how the decoder can be implemented without this knowledge. In the following, we assume a binary input additive white Gaussian

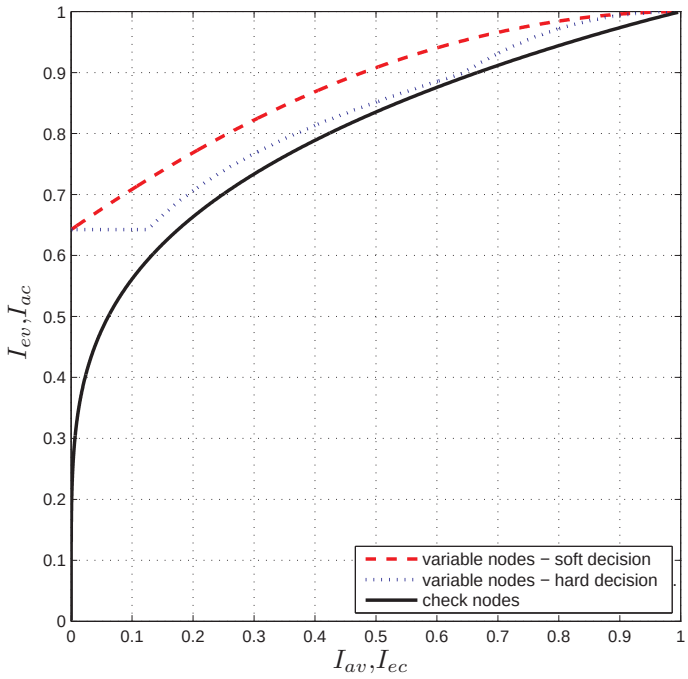


Figure 5.1: EXIT function of check nodes with $d_c = 6$ and variable nodes with $d_v = 4$ and $\sigma = 0.67$ for hard and soft decision output.

noise (BIAWGN) communication channel with noise variance σ^2 where the received values y are converted to L-values (see Section 2.1.4) as

$$L_{ch} = \frac{2}{\sigma^2} y, \tag{5.6}$$

before being quantized. The unquantized L-values are Gaussian distributed with variance σ_{ch}^2 and mean $\mu_{ch} = \sigma_{ch}^2/2$ [RSU01], where σ_{ch}^2 is related to the variance of the additive Gaussian noise as

$$\sigma_{ch}^2 = \frac{4}{\sigma^2}. \tag{5.7}$$

In the following sections, we derive the EXIT function of the variable node decoder for various quantization schemes.

5.2.1 Hard Decision Channel

Consider the case where the receiver performs hard decisions. Then the communication channel can be modeled as a BSC with crossover probability ϵ_{ch}. Let D_{ch} and D_{av} denote the reliabilities of the communication and extrinsic channel, respectively. In order to compute an outgoing message, the variable node decoder converts all

incoming messages to L-values and computes the sum of the channel L-value and all other $(d_v - 1)$ incoming L-values as

$$L_{ev,j} = L_{ch} + \sum_{i=1;i\neq j}^{d_v} L_{av,i}. \tag{5.8}$$

The outgoing message transmitted to the check node decoder is the hard decision of L_{ev}. The probability that this message is in error is

$$\begin{aligned} \epsilon_{ev} &= f_v(\epsilon_{av}; d_v, \epsilon_{ch}) \\ &= 1 - \epsilon_{ch} B\left(\left\lfloor \frac{D_{av}(d_v - 1) - D_{ch}}{2D_{av}} \right\rfloor ; d_v - 1, \epsilon_{av}\right) \\ &\quad - (1 - \epsilon_{ch}) B\left(\left\lfloor \frac{D_{av}(d_v - 1) + D_{ch}}{2D_{av}} \right\rfloor ; d_v - 1, \epsilon_{av}\right), \end{aligned} \tag{5.9}$$

where

$$B(k; n, p) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i} \tag{5.10}$$

denotes the binomial cumulative distribution. The first term in (5.9) represents the probability that the channel message is in error and the messages from the check nodes are not able to correct it, and the second term represents the probability that the channel message is correct but too many check node messages are in error.

An example of this EXIT function is shown in Figure 5.1. It can be observed that the decoder changes its behavior depending on I_{av}. This corresponds to the Gallager B algorithm [Gal62,Gal63] where the majority decision rule is changed depending on the crossover probability. Compared with channels using a larger output alphabet, this EXIT function serves as a lower bound. Using the L-value representation we are able to generalize this algorithm to channels with larger output alphabets.

5.2.2 Soft Decision Channel

In the limit of no quantization of the output of a BIAWGN channel, the crossover probability at the output of the variable node decoder can be derived as

$$\epsilon_{ev} = 1 - \sum_{z=0}^{d_v - 1} b(z; d_v - 1, \epsilon_{av}) Q\left(\frac{D_{av}(d_v - 1 - 2z) - \mu_{ch}}{\sigma_{ch}}\right), \tag{5.11}$$

where $Q(\cdot)$ is defined as

$$Q(\phi) \hat{=} \frac{1}{\sqrt{2\pi}} \int_{\phi}^{\infty} e^{-\frac{\psi^2}{2}} d\psi, \tag{5.12}$$

and

$$b(k;n,p) = \binom{n}{k} p^k (1-p)^{n-k}, \tag{5.13}$$

denotes the binomial probability mass function.

The EXIT function for this type of channel is shown in Figure 5.1. Since every quantized channel can be derived from the soft output channel, this EXIT function serves as an upper bound.

5.2.3 Larger Output Alphabets

We consider the case where the channel messages stem from a binary input additive noise channel with a quantizer. The quantizer provides the sign of the received values and the magnitude where the quantization scheme is described by the vector $\boldsymbol{\zeta} = [\zeta_0, \ldots, \zeta_K]$ where $\zeta_0 < \zeta_1 < \cdots < \zeta_K$. Let k be the sub-channel indicator defined as

$$k \hat{=} \underset{k'}{\operatorname{argmin}} \, |L_{ch}| < \zeta_{k'}, \tag{5.14}$$

and let $L_{ch,K}$ be the quantized channel message

$$L_{ch,K} = \operatorname{sgn}(L_{ch}) \cdot k. \tag{5.15}$$

Following [Lan05], this channel quantization scheme can be decomposed as $(K+1)$ BSCs. Sub-channels $k = 0, 1, \ldots, K$ are used with probabilities p_k and let $\epsilon_{ch,k}$ denote the crossover probability of sub-channel k. We define sub-channel zero as a BSC with crossover probability 0.5 [Lan05]. The parameters for sub-channel zero are

$$p_0 = \int_{-\zeta_0}^{\zeta_0} g(y)dy, \quad \text{and} \quad \epsilon_{ch,0} = \frac{1}{2}, \tag{5.16}$$

where $g(y) = p(Y = y|X = +1)$ is the conditional transition probability of the channel. The probability that a sub-channel $k > 0$ is used is

$$\begin{aligned} p_k &= \int_{\zeta_{k-1}}^{\zeta_k} g(y)dy + \int_{-\zeta_k}^{-\zeta_{k-1}} g(y)dy \\ &= \int_{-\zeta_k}^{\zeta_k} g(y)dy - p_{k-1}, \end{aligned} \tag{5.17}$$

and the crossover probability $\epsilon_{ch,k}$ of this sub-channel is given by

$$\epsilon_{ch,k} = \frac{1}{p_k} \int_{-\zeta_k}^{-\zeta_{k-1}} g(y)dy. \tag{5.18}$$

The EXIT function of the overall channel is the average EXIT function of the sub-channels

$$I_{ev} = \sum_{k=0}^{K} p_k I_{ev,k}. \tag{5.19}$$

We will present two examples of channels with higher output alphabet—the binary symmetric erasure channel (BSEC) and the binary symmetric quaternary output channel (BSQC).

Example 1. *The output of the binary symmetric erasure channel (BSEC) takes on values from* $\{+1, 0, -1\}$*. This quantization can be represented using*

$$\boldsymbol{\zeta} = \left[\zeta_0, \quad \infty\right]. \tag{5.20}$$

The EXIT function of the variable node decoder for a BSEC with $\zeta_0 = 1.69$ *is shown in Figure 5.2. The value of* ζ_0 *was chosen using the criterion that the area below the EXIT function is maximized. For code design, this parameter has to be optimized jointly with the degree distributions of the code.*

Example 2. *The output of a binary symmetric quaternary output channel (BSQC) takes on values from* $\{-2, -1, +1, +2\}$ *which can be represented by a quantization using*

$$\boldsymbol{\zeta} = \left[0, \quad \zeta_1, \quad \infty\right]. \tag{5.21}$$

The EXIT function of the variable node decoder for this channel using $\zeta_1 = 1.90$ *(which maximizes the area below the EXIT function) is shown in Figure 5.2.*

5.3 Code Optimization

In this section we describe how to maximize the code rate for a given channel by optimizing the variable node degree distribution [RSU01] of the code (we consider only check regular codes). In the case of binary message-passing decoders, the EXIT function of the mixture of codes cannot be computed as the average of the EXIT functions of the component codes as presented in [AKtB04] because of the following reason. In Section 2.1.1, we showed that the computation of the capacity is a linear operation on the density of the L-values. Therefore, a linear combination of densities (i.e. a mixture of codes) and the computation of the capacity can be interchanged. In the case of binary message passing decoders, the density of the computed L-values of the i^{th} component code consists of two nonzero values at $+R_{ev,i}$ and $-R_{ev,i}$. The mixture of these densities has more than two nonzero values, but will be quantized to $\{+1, -1\}$ before being transmitted to the check nodes. This nonlinear operation prohibits the exchange of averaging and the computation of the mutual information.

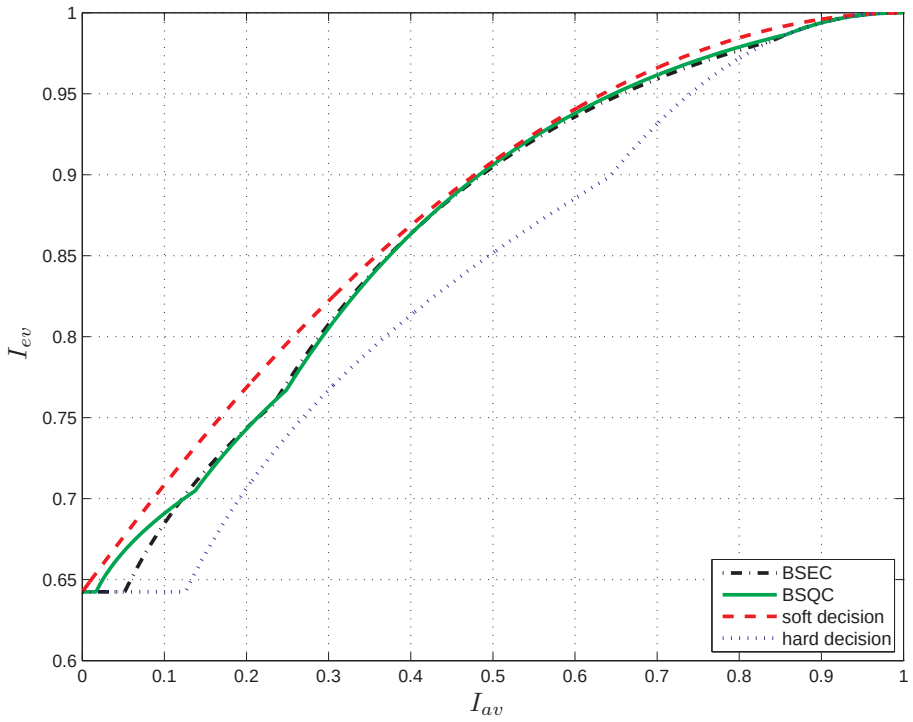


Figure 5.2: EXIT functions of variable nodes with $d_v = 4$ and $\sigma = 0.67$ for BSEC and BSQC channels.

Computing the resulting EXIT function of the variable node decoder can still be written in a linear manner by averaging over the crossover probabilities instead of the individual EXIT functions [AK02] as

$$\epsilon_{ev} = \sum_{i=1}^{d_{v,max}} \lambda_i \epsilon_{ev,i}, \tag{5.22}$$

and also formulating the constraint in terms of crossover probabilities

$$f_v(\epsilon) > f_c^{-1}(\epsilon), \tag{5.23}$$

for every ϵ in $(0, 0.5)$. Since the rate of the code is a linear function in λ and also the constraints are linear, we can apply linear programming to solve this optimization problem.

Using this procedure, we optimized codes and compared them with the capacity of the BIAWGN and BSC. For the optimization we set the maximum variable node degree to $d_{v,max} = 100$ and performed the optimization for check node degrees in the range between 2 and 100. The thresholds of these codes are shown in Figure 5.3.

It can be observed, that the gap to capacity decreases with increasing rate. This makes binary message-passing decoders attractive for applications which require a

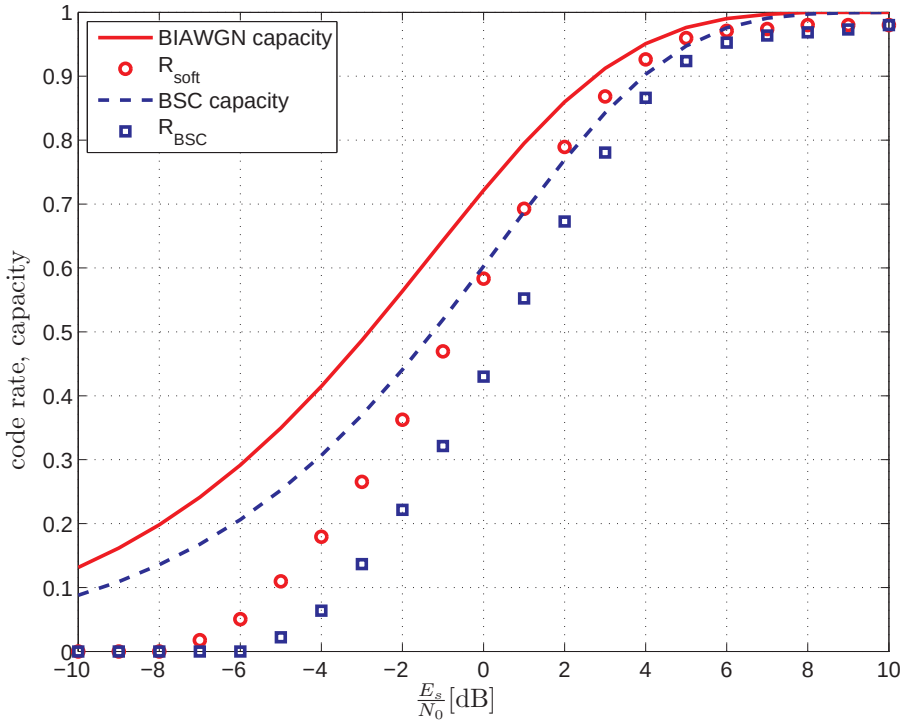


Figure 5.3: Thresholds of optimized codes for soft channel information and hard decision channel (BSC).

high code rate, e.g. [ITU04]. For a rate of 0.9, the best code using soft channel information is as close as approximately 0.5 dB to the capacity of the BIAWGN channel. Note that the soft channel and the hard decision channel serve as upper and lower bounds respectively for all quantization schemes.

5.4 Estimation of the A-Priori Channel

In Section 5.2, we assumed that the variable node decoder has knowledge of the parameters of the extrinsic channel, i.e. it knows the crossover probability of the messages going from check nodes to variable nodes. Since this assumption cannot be satisfied in practice, we present a method to implement the decoder without this knowledge.

By using the EXIT functions, we can predict the trajectory of the decoder. We assume a channel parameter where convergence in a predefined maximum number of iterations is guaranteed (i.e. just above the threshold) and use the decoding trajectory to compute a sequence of crossover probabilities. This sequence is used instead of the genie-aided knowledge by the variable node decoder. An example of a pre-

dicted decoding trajectory and the corresponding sequence of crossover probabilities is shown in Figure 5.4 and 5.5, respectively.

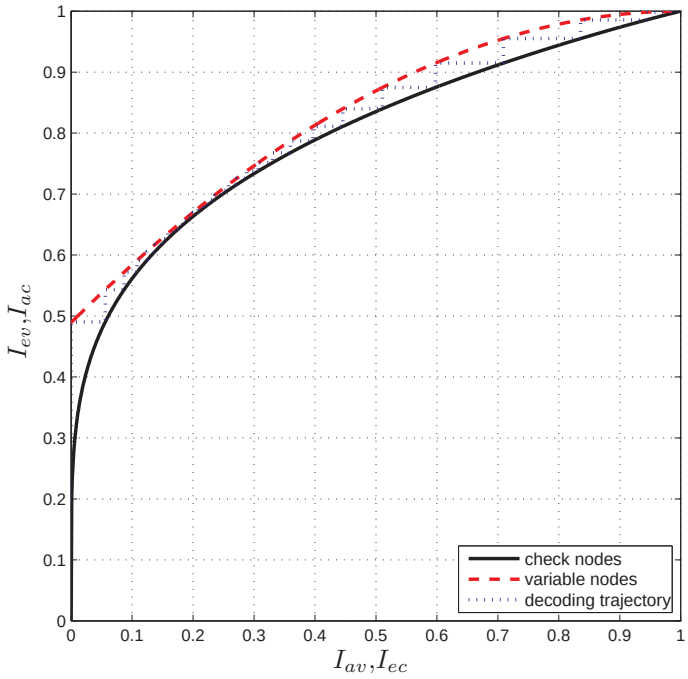


Figure 5.4: Prediction of the decoding trajectory for $d_v = 4$, $d_c = 6$ and $\sigma = 0.83$ for soft decision channel.

5.5 Majority Decision Rules

The concept of converting all incoming messages to L-values, perform decoding in the L-value domain and send the hard decision from variable to check nodes is a theoretical framework in order to derive analytical expressions for the EXIT function of the variable node decoder. In practice, these operations are replaced by majority decision rules. Consider a variable node of degree d_v. Depending on I_{av}, a minimum number b of messages from the check nodes have to disagree with the channel messages, in order to change the outgoing message of that variable node. This quantity b is given by

$$b = \left\lfloor \frac{D_{av}(d_v - 1) - D_{ch}}{2D_{av}} \right\rfloor . \tag{5.24}$$

This allows us to derive a majority decision rule, that is parametrized by I_{av}. For channels with higher output alphabet, a majority decision rule for every sub-channel has to be defined.

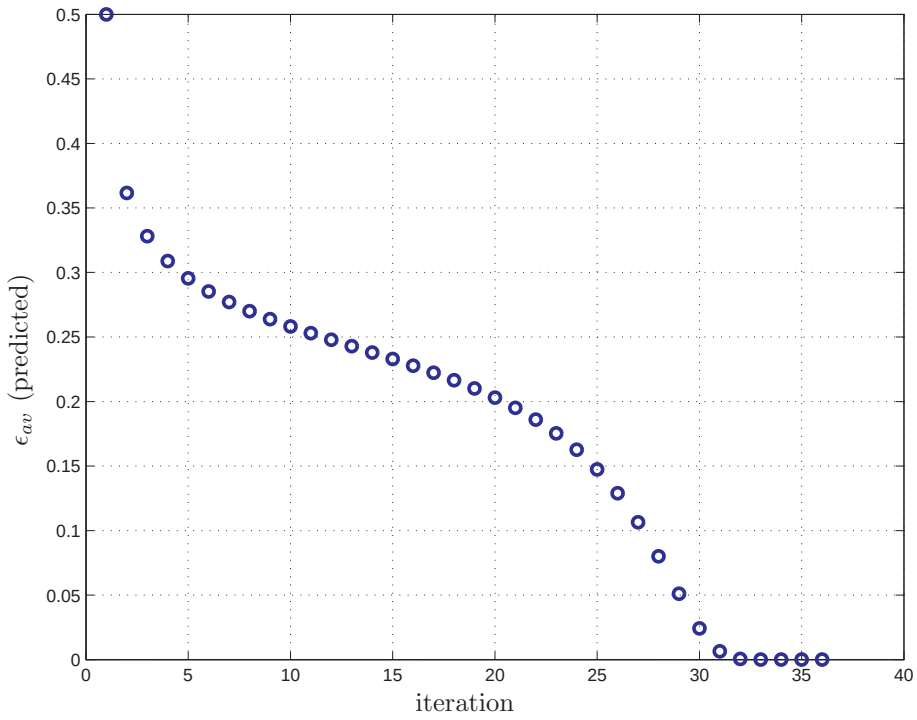


Figure 5.5: Sequence of predicted crossover probabilities at the input of the variable node decoder.

	BSC	BSEC	BSQC	Soft
E_b/N_0[dB]	3.62	2.95	2.62	2.28

Table 5.1: Thresholds for code of rate 0.5.

5.6 Simulation Results

As an example, we optimized codes of rate 0.5 for the BSC, BSEC, BSQC and the soft information channel. The thresholds of these codes using the associated quantization schemes are shown in Table 5.1 and the bit error rate simulation results are shown in Figure 5.6 using codes of length $N = 10^4$ constructed with the progressive edge-growth (PEG) algorithm [HEA05, Hu].

The system with hard channel decisions (BSC) corresponds to the algorithm Gallager B. It can be seen, that by adding one more bit for the channel messages and quantize them according to a BSQC, the performance of this algorithm can be improved by more than 1.0 dB with only a small increase in complexity. A finer quantization of the channel messages will not result in a significant gain, since the gap to the unquantized system is only approximately 0.25 dB.

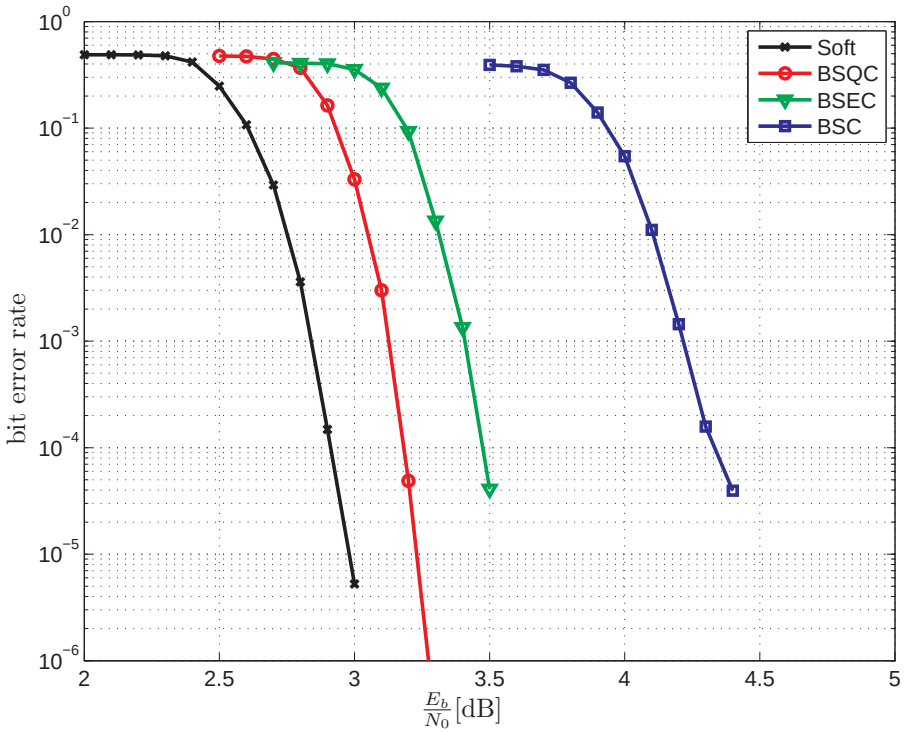


Figure 5.6: Bit error rate simulations for code of rate 0.5.

From Figure 5.3, one can see that the gap to the capacity of the BIAWGN channel decreases with increasing code rate. Therefore, binary message passing decoders are suited for high rate systems. Such a high rate code is used in [ITU04, Section I.6]. This code of rate $R = 0.9375$ is a regular LDPC code with $d_v = 7$, $d_c = 112$ and is used for an optical transmission system for high data rates up to 40Gbps. In [ITU04] it is assumed that the decoder observes hard decisions from the channel. Using our analysis, the thresholds for channels with higher output alphabet are shown in Table 5.2 assuming that the unquantized channel can be modeled as a BIAWGN channel. The corresponding bit error rate simulations are shown in Figure 5.7. From these results, it can be seen that a quantization to three values instead of two leads to an improvement of 0.96dB for the BSEC and a quantization to four values to an improvement of 0.97dB for the BSQC. Both quantization schemes need two bits to represent the channel observation instead of one bit required for a hard decision. Therefore, adding an additional bit leads to significant improvement of the decoding capability.

	BSC	BSEC	BSQC	Soft
E_b/N_0[dB]	6.08	5.12	5.11	5.02

Table 5.2: Thresholds for code used in [ITU04].

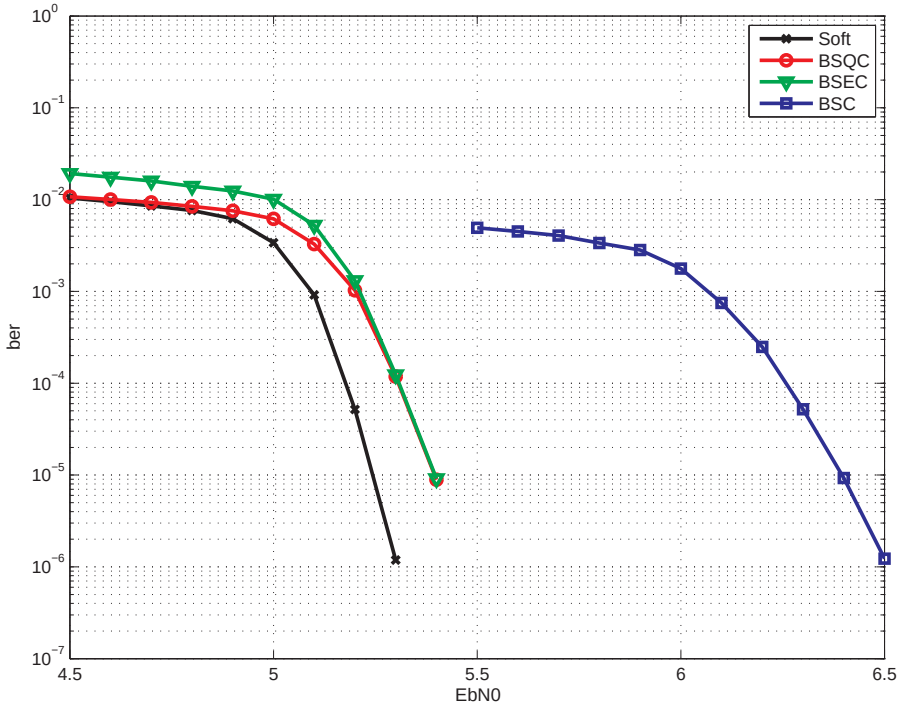


Figure 5.7: Bit error rate simulations for code of rate 0.9375.

5.7 Conclusions

We analyzed binary message-passing decoders using EXIT charts. For channels which deliver hard decisions, this analysis lead to an algorithm that is equivalent to Gallagers decoding algorithm B. The analysis of this algorithm was extended to channels with higher output alphabets including channels that deliver soft information. A small increase of the output alphabet of the channel results in a significant gain in performance. Finally, we showed that the mixing property of EXIT functions does not apply directly to binary message-passing algorithms, and presented a modified mixing method in order to optimize codes.

6 Joint Receiver Tasks

A receiver has to perform several tasks like equalization, demapping, multi-user detection, decoding, etc. In order to achieve best performance, these problems should be solved jointly. However, due to the complexity of the joint solution, it is more attractive to solve them separately and iterate between those components. If we aim to iterate over an entire receiver *front-end* (e.g. a soft demapper for a specific modulation format or an equalizer for a channel with intersymbol-interference) and an error correcting code, we have to optimize the overall EXIT function of the code to match the EXIT function of such receiver front-end[1]. The overall extrinsic information transfer function of a code optimized for the Gaussian memoryless channel is a step function [PSS]. This is consistent with the curve fitting approach of EXIT charts, since the extrinsic information transfer function of a memoryless Gaussian channel is constant and therefore, the best fit is achieved using a step function. On the other hand, for most other channels, e.g., channels with memory or non-binary channels, the EXIT function of the channel is not constant. Therefore, the front-end of the receiver can make use of a-priori information.

When designing LDPC codes, it is common practice to fix the degree of the check nodes (i.e. use check-regular codes) and optimize the variable node distribution. One approach for matching the LDPC code to the front-end is to combine the variable nodes with the front-end as shown in [tBKA04]. This requires an analytical expression for the EXIT function of the front-end, which is provided in [tBKA04] by taking a polynomial approximation of the measured EXIT function. We follow a different approach, by matching the overall EXIT function of the LDPC decoder to the front-end. This only requires the measured EXIT function of the front-end and eliminates the need for an analytical approximation.

Designing the degree distributions of an LDPC code to match a given front-end is a nontrivial task, since the optimization problem is nonlinear, and therefore, search algorithms have to be employed to find good codes, as shown in [SPS05]. In this chapter, we optimize both the variable and the check node distributions. Ideally, one would like to optimize both distributions jointly, but this is usually not feasible. We will show in Section 6.3 that optimization of the check node distribution for

[1] This is motivated by the area property of the EXIT chart [AKtB04] which is only correct for the binary erasure channel (BEC) but results in good codes also for other channels.

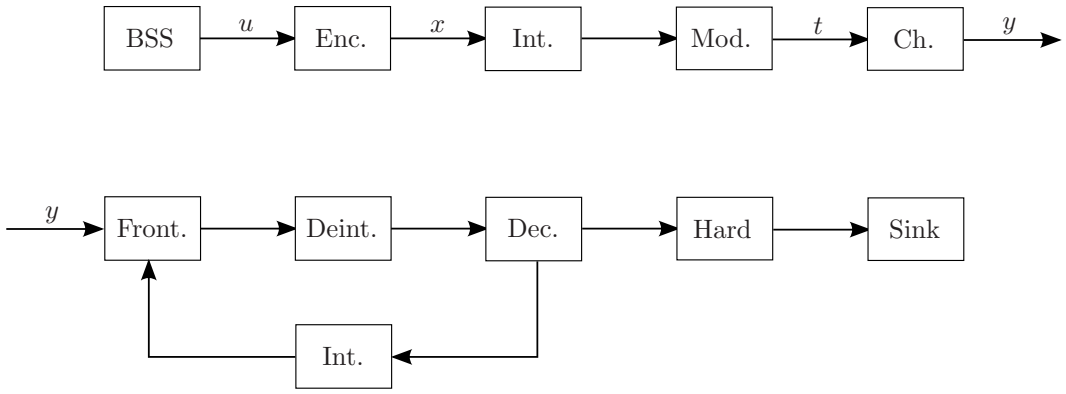


Figure 6.1: System model.

a fixed variable node distribution is a linear problem that can be solved efficiently using linear programming. By optimizing the check node degree distribution for a fixed variable node degree distribution, the search for a good variable node degree distribution is far less complex, since the EXIT function of the LDPC code is less sensitive to the choice of the variable node distribution as we will see later.

6.1 System Model

We consider a system as shown in Figure 6.1. The transmitter encodes a binary vector $\boldsymbol{u}$ to a binary vector $\boldsymbol{x}$. This block is interleaved, mapped to a vector $\boldsymbol{t}$ of complex transmit symbols, and transmitted over the channel. For the moment, we do not make any assumptions about the channel. The receiver processes the received vector $\boldsymbol{y}$ in a front-end and iterates between the (iterative) channel decoder and this front-end for a given number of iterations, before making a hard decision on the information vector. In this double iterative system, we perform iterations inside the LDPC decoder until it reaches a steady-state (i.e., until the decoder converges to a codeword, or gets stuck at a point where the EXIT functions of variable and check nodes intersect) before returning to the outer iteration between the LDPC decoder and the front-end.

In order to optimize iterative processing between the front-end and the decoder, we aim to match the overall EXIT function of the code to the EXIT function of the front-end. Since, in our general model, we allow the front-end to perform several possible tasks, we do not make any assumptions about the transfer function of this component and regard it as given (obtained either analytically or via simulation) denoted by

$$I_{e,frontend} = T(I_a). \tag{6.1}$$

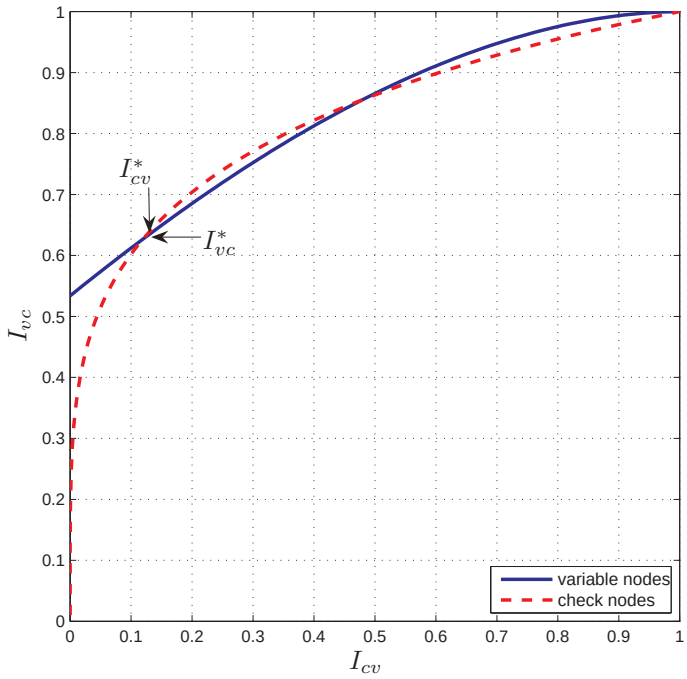


Figure 6.2: EXIT chart of LDPC code.

6.2 EXIT Function of LDPC Codes

In Chapter 3, we derived the EXIT functions of the component codes that form an LDPC code. In order to design LDPC codes that match a given EXIT function of a front-end, we will derive the EXIT function of the whole LDPC decoder in this section. Since we are iterating between the front-end of the receiver and the channel decoder, which is an iterative system in itself, we have to assume some scheduling. Our analysis relies on asymptotic long blocks resulting in a graph for which the probability of cycles tends to zero and therefore, we are free to choose an arbitrary scheduling. Our assumption is that the decoder continues iterating until it reaches a steady-state, before performing an outer iteration between the decoder and the front-end.

An EXIT chart of the LDPC decoder is shown in Figure 6.2. Quantities from variable to check nodes are denoted with the subscript vc and quantities from check to variable nodes with cv. When we allow the decoder to perform as many iterations as needed to converge to a steady-state, the decoder will always get stuck in the smallest intersection between the variable and check node transfer curves denoted with I^*_{vc} and I^*_{cv}.

In order to obtain simple expressions for I^*_{vc} and I^*_{cv}, we use the duality property [AKtB04], which is an approximation for channels other than the BEC. Using this,

we can write the variable and check node transfer functions as [AKtB04]

$$I_{vc} = \sum_{i=1}^{d_{v,max}} \lambda_i \cdot J\left(\sqrt{(i-1)J^{-1}(I_{cv})^2 + J^{-1}(I_{a,LDPC})^2}\right), \tag{6.2}$$

$$I_{cv} = 1 - \sum_{j=1}^{d_{c,max}} \rho_j \cdot J\left(\sqrt{j-1}J^{-1}(1-I_{vc})\right), \tag{6.3}$$

where the function $J(\cdot)$ is defined in Section 2.1.4.

The intersection point can thus be found by solving

$$\begin{aligned} I_{cv}^* = 1 - \sum_{j=1}^{d_{c,max}} \rho_j \cdot J\Bigg(\sqrt{j-1}J^{-1}\Bigg(1 \\ - \sum_{i=1}^{d_{v,max}} \lambda_i \cdot J\left(\sqrt{(i-1)J^{-1}(I_{cv}^*)^2 + J^{-1}(I_{a,LDPC})^2}\right)\Bigg)\Bigg), \end{aligned} \tag{6.4}$$

and this quantity is used to compute the extrinsic mutual information of the LDPC decoder as

$$I_{e,LDPC} = \sum_{i=1}^{d_{v,max}} \bar{\lambda}_i \cdot J\left(\sqrt{i}J^{-1}(I_{cv}^*)\right), \tag{6.5}$$

where $\bar{\lambda}_i$ denotes the fraction of variable nodes of degree i (see Section 2.5).

6.3 LDPC Code Design

Using (6.4) and (6.5), we are now able to state our design problem. We aim to maximize the design rate R_d of the code defined as

$$R_d = 1 - \frac{\sum_{j=1}^{d_{c,max}} \frac{\rho_j}{j}}{\sum_{i=1}^{d_{v,max}} \frac{\lambda_i}{i}} \tag{6.6}$$

under the constraint that the transfer function of the LDPC decoder (6.1) is larger than a given target function at every point

$$I_{e,LDPC}(I_a) \geq I_{e,target}(I_a). \tag{6.7}$$

This target function is given by the inverse transfer function of the front-end

$$I_{e,target}(I_a) = T^{-1}(I_a). \tag{6.8}$$

Looking at (6.4), (6.5), and (6.6), one can observe that the maximization is non-linear in λ given ρ, but linear in ρ given λ. Therefore, the optimization of ρ is much

simpler than the optimization of λ. We will show in an example in Section 6.4 that the transfer function of the LDPC decoder is less sensitive to the choice of λ when ρ is optimized for each λ, than when an optimal λ is sought for a fixed ρ. Therefore, we can reduce the search space for λ significantly, which allows to perform an exhaustive search over the variable node degree distribution.

Stability Condition

To guarantee the convergence of the decoder, the degree distributions have to satisfy the stability condition [RSU01]. For the Gaussian channel with variance σ^2, the stability condition can be written as

$$\sum_{j=1}^{d_{c,max}} \rho_j \cdot (i-1) < \frac{e^{\frac{1}{2\sigma^2}}}{\lambda_2}. \tag{6.9}$$

Note that the stability condition is a requirement for successful decoding of the LDPC code, that can be derived from properties of its EXIT chart in the top right corner. Therefore, in our double iterative setup, the stability condition is only relevant in the last overall iteration, when the LDPC decoder is expected to decode successfully, and its component EXIT curves do not intersect. If we assume that the messages between the front-end and the LDPC decoder are Gaussian distributed, the condition (6.9) has thus to be satisfied for

$$\sigma \leq \frac{2}{J^{-1}(I_{a,LDPC,max})}, \tag{6.10}$$

where $I_{a,LDPC,max}$ is the maximum of the EXIT function of the receiver front-end. This additional constraint is also linear in ρ and can therefore directly be included in the linear optimization problem.

6.4 Turbo Demapping

As an example, we will apply the method described in the previous sections to iterative demapping and decoding (*turbo-demapping* [SB04]). Consider a system where the transmitter maps the encoded and interleaved data to a 16 QAM signal constellation. Depending on the applied mapping, the soft demapper in the receiver front-end has a certain information transfer function. Two examples for a 16 QAM signal constellation are shown in Figure 6.4. For Gray mapping, this transfer function is approximately constant, meaning that turbo-demapping does not lead to a significant performance improvement. However, for other mappings, like set-partitioning

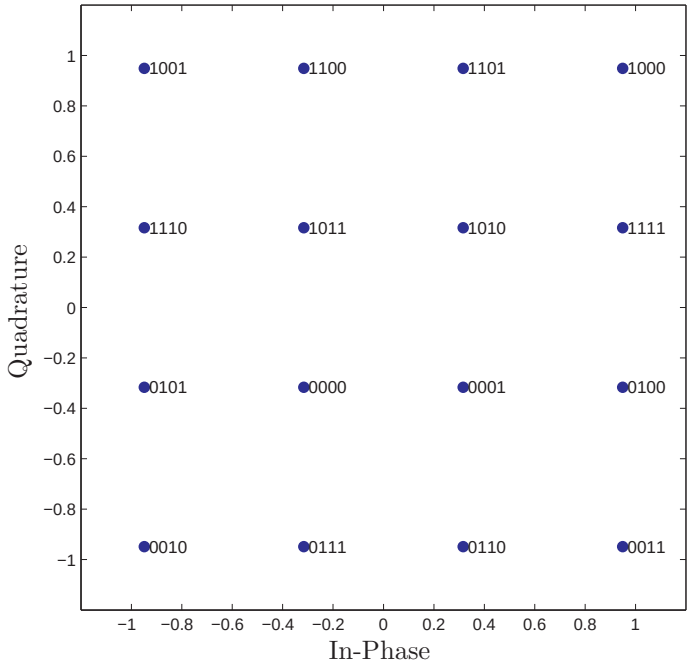


Figure 6.3: 16 QAM constellation and set-partitioning mapping.

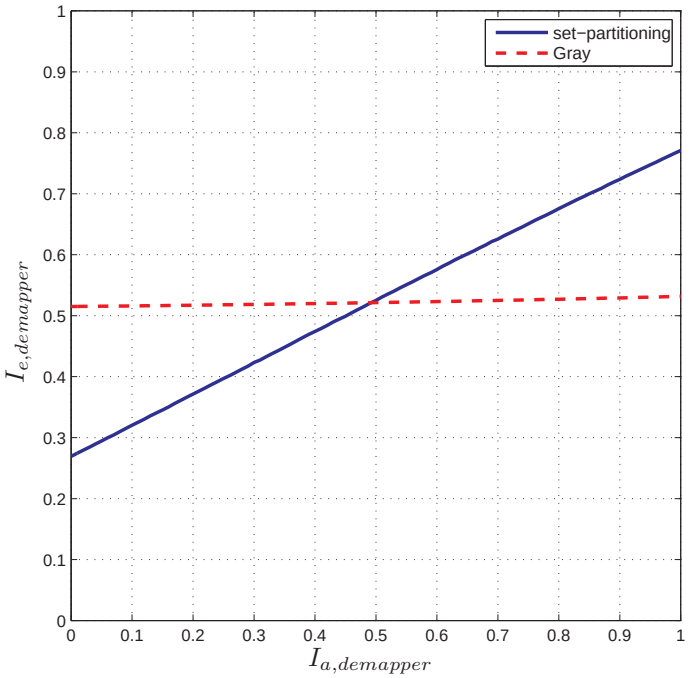


Figure 6.4: EXIT function of the demapper.

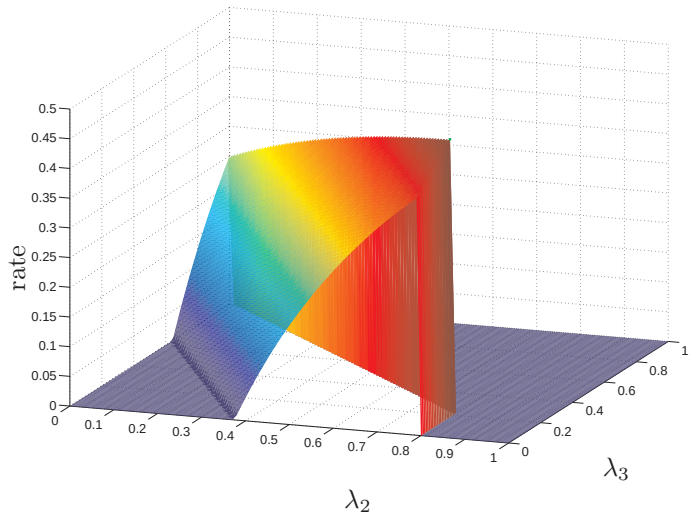


Figure 6.5: Code search for fixed ρ.

or non-unique symbol mappings [SH05], the transfer function of the demapper is not constant, which makes them a good candidate for turbo-demapping.

We will consider a 16 QAM signal constellation and set-partitioning mapping, transmitted over a memoryless additive white Gaussian noise (AWGN) channel with variance σ^2. The constellation and the mapping are shown in Figure 6.3. We perform no constellation shaping, i.e. every constellation point is transmitted with the same probability. The transfer function of the soft demapper at $\sigma^2 = 0.28$ is shown in Figure 6.4. Obviously, an LDPC code that was optimized for an AWGN channel (and therefore has a step-like transfer function [PSS]) cannot achieve capacity for this system.

To demonstrate the advantage of optimizing both the variable and check node degree distribution, we compare two strategies. For both approaches, we restrict λ to have only three nonzero elements, namely for variable node degrees of two, three and ten. This restriction leads to only two degrees of freedom for the variable node degree distribution, making it easy to apply an exhaustive search over the complete search space.

First, the check node degree distribution is kept constant and regular with a check node degree of 4, which was found to deliver the highest rates, following the approach in [SPS05]. The result of the exhaustive search is shown in Figure 6.5. It can be observed that the rate achieved is sensitive to the choice of λ. Furthermore, it can be seen that the fixed check degree distribution limits the maximum value of λ_2, due to the stability condition. The maximum rate found using this method is $R_d = 0.46$.

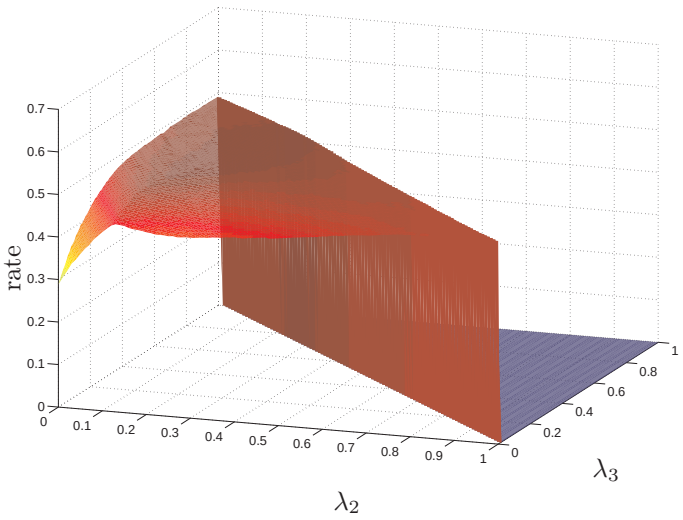


Figure 6.6: Code search for optimized ρ.

The second approach is to search for the best λ over the same search space, but to optimize ρ for every λ. The result of this search for a maximum check node degree of 30 is shown in Figure 6.6. Using the optimization of ρ, a maximum rate of $R_d = 0.51$ was found, and the rate is less sensitive to the choice of λ. Even when setting all the variable nodes to degree two or three, the rates found were $R_d = 0.48$ and $R_d = 0.49$, respectively, still outperforming the best rate obtained using the first approach with a non-optimized check node degree distribution.

The EXIT function of the code with the optimized check node distribution matching the demapper function is shown in Figure 6.7 and the results are verified by bit error rate simulations with a codeword length of $N = 10^5$. For comparison, we also simulated the system with a rate $R_d = 0.5$ LDPC code optimized for an AWGN channel with BPSK mapping (having approximately a step function as transfer characteristic). From the results in Figure 6.8 it can be seen that the gain due to the code optimization is approximately 3.0dB.

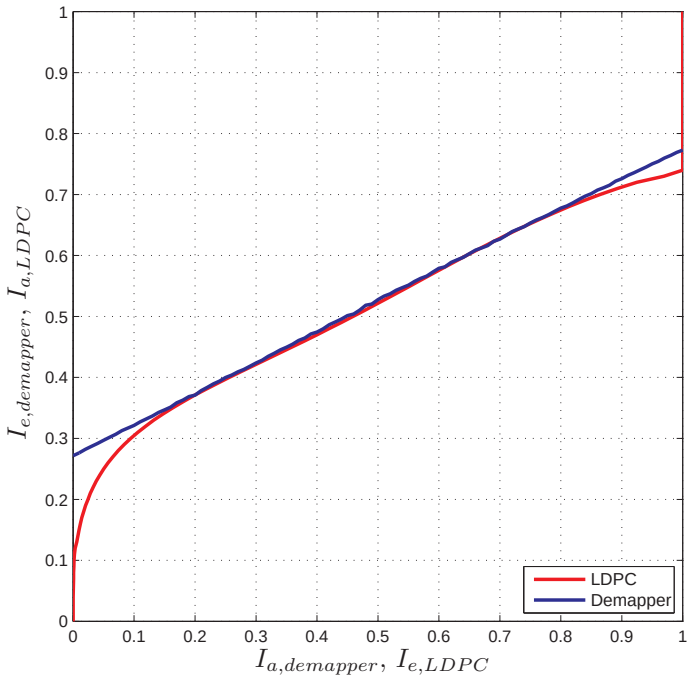


Figure 6.7: EXIT chart of optimized LDPC code and demapper.

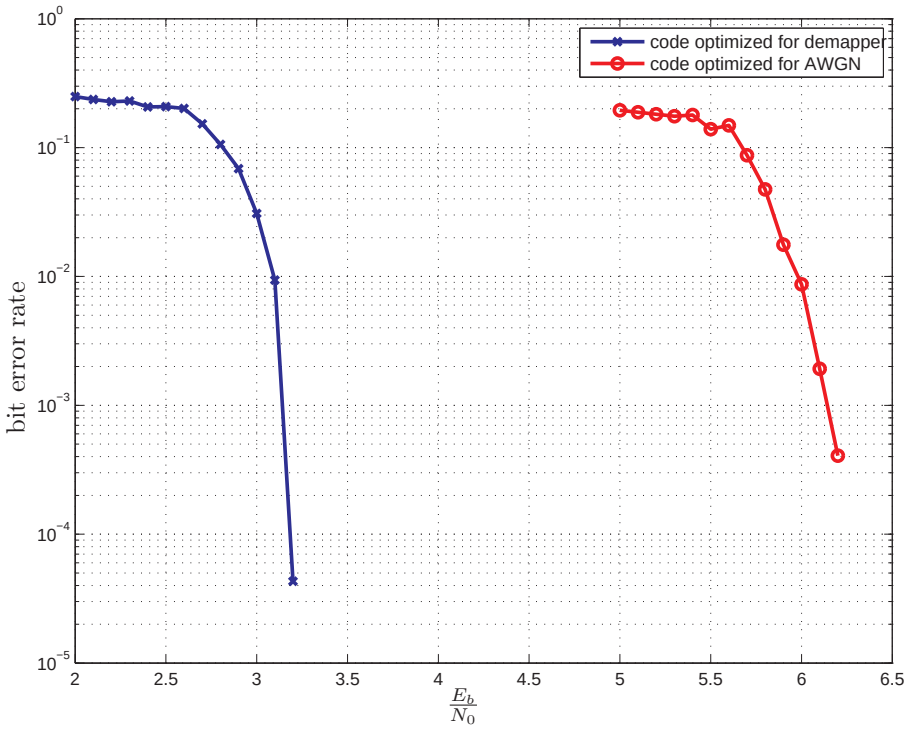


Figure 6.8: Bit error rate simulations.

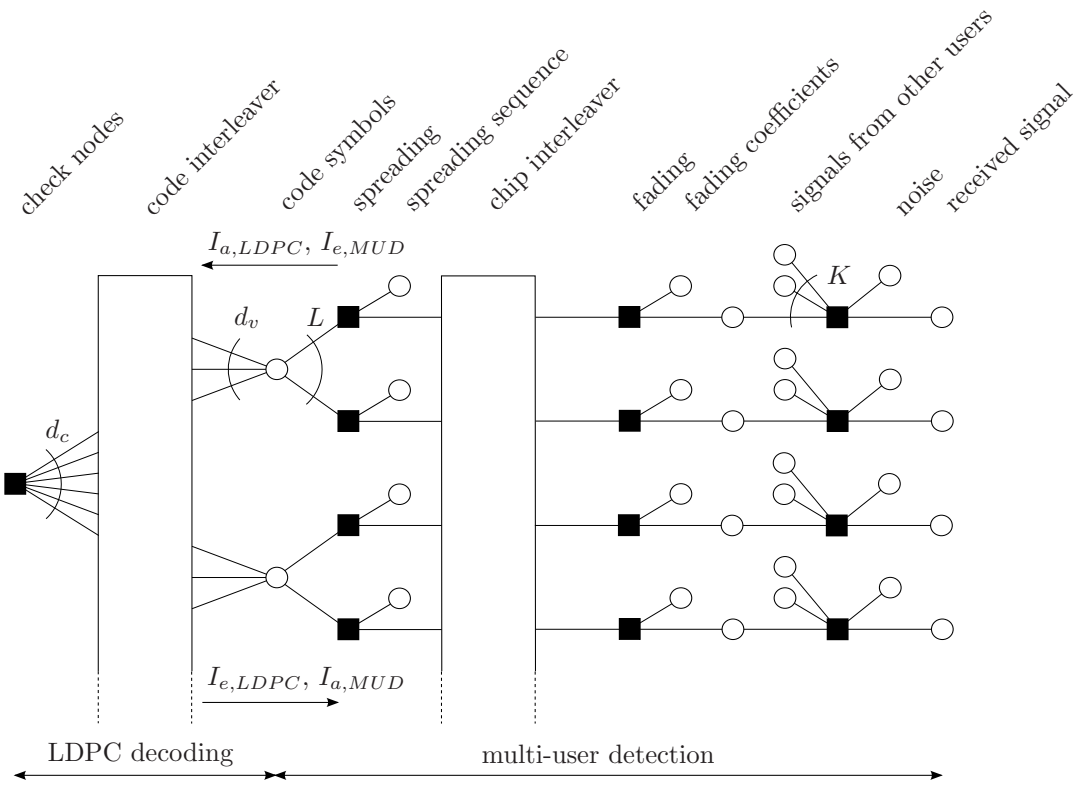


Figure 6.9: Factor graph for iterative multi-user detection and decoding.

6.5 Joint Multi-User Detection and Decoding

In this section, we consider a receiver that performs iterative multi-user detection and decoding. The whole system can be represented by augmenting the factor graph [KFL01] of the LDPC code with the elements of the multi-user detector as shown in Figure 6.9. The left part shows the LDPC decoder with check nodes of degree d_c and variable nodes of degree d_v. The right part corresponds to the multi-user detector, where every code symbol is repeated L times, which corresponds to the spreading length. Spreading is performed by multiplying a spreading sequence and the spreaded chips are interleaved over the complete block. This has the advantage that chips corresponding to one code symbol are spread in time which increases the diversity in the case of a slow fading channel. We will consider random spreading sequences which are not necessarily orthogonal but allow to operate the system overloaded. The chips are multiplied by fading coefficients which can be independent for every chip and we assume perfect knowledge of the spreading coefficients. The receiver observes the sum of the transmitted signals from K users plus white Gaussian noise.

6.5.1 Multi-User Detector

We focus on systems with low implementation complexity that are scalable in the number of users and in the spreading length. The optimal multi-user detector has a complexity that is exponential in the number of users. The multi-user detector presented in this section performs interference cancellation at chip level which can be implemented using a message passing algorithm avoiding the need of matrix multiplications and matrix inversions. The computational complexity of this system is proportional to the number of edges in the graph. Therefore, the system has a complexity of the order $\mathcal{O}(K \cdot L)$.

Let $\boldsymbol{c}_k$ denote the codeword of length N of user k with elements from $\{+1, -1\}$. The transmitter performs spreading followed by a chip-level interleaver. The whole process with spreading length L can equivalently be described by a repetition code with encoder matrix $\boldsymbol{G}_R$ of dimension $LN \times N$, the multiplication with a spreading sequence $\boldsymbol{s}_k$ of length LN followed by a permutation that represents the chip-level interleaver as

$$\boldsymbol{x}_k = \boldsymbol{P}_k \cdot \operatorname{diag}(\boldsymbol{s}_k) \cdot \boldsymbol{G}_R \cdot \boldsymbol{c}_k, \tag{6.11}$$

where $\boldsymbol{x}_k$ denotes the signal transmitted by user k, $\boldsymbol{P}_k$ is the permutation matrix of dimension $LN \times LN$ and the operator $\operatorname{diag}(\cdot)$ produces a diagonal matrix with the elements of its argument on the main diagonal.

The received signal $\boldsymbol{y}$ is composed of the sum of all user signals and additive white Gaussian noise as

$$\boldsymbol{y} = \sum_{k=1}^{K} (\operatorname{diag}(\boldsymbol{h}_k) \cdot \boldsymbol{x}_k) + \boldsymbol{n}, \tag{6.12}$$

where $\boldsymbol{h}_k$ contains the LN channel realizations of user k.

The multi-user detector performs interference cancellation on chip level for every user. Let $\boldsymbol{z}_k$ denote the signal after the interference cancellation given by

$$\boldsymbol{z}_k = \boldsymbol{y} - \sum_{j=1;j\neq k}^{K} \left(\operatorname{diag}(\boldsymbol{h}_j) \cdot \boldsymbol{P}_j \cdot \operatorname{diag}(\boldsymbol{s}_j) \cdot \boldsymbol{G}_R \cdot \tanh \frac{\boldsymbol{L}_{a,j}}{2} \right), \tag{6.13}$$

where $\boldsymbol{L}_{a,j}$ denotes the a-priori L-values of the code symbols, that are provided by the LDPC decoder. The vector $\boldsymbol{z}_k$ is well approximated by a Gaussian distribution [WP99] with mean $\boldsymbol{h}_k$ and variance

$$\boldsymbol{\sigma}_{z,k}^2 = \sigma_n^2 + \sum_{j=1;j\neq k}^{K} \operatorname{diag}(\boldsymbol{h}_j)^2 \left(1 - \left(\tanh \frac{\boldsymbol{L}_{a,j}}{2} \right)^2 \right). \tag{6.14}$$

This allows to express the L-values $\boldsymbol{L}_k$ of $\boldsymbol{z}_k$ as

$$\boldsymbol{L}_k = 2 \cdot \boldsymbol{z}_k \operatorname{diag}(\boldsymbol{h}_k) \cdot \operatorname{diag}\left(\boldsymbol{\sigma}_{z,k}^2\right)^{-1}. \tag{6.15}$$

Finally, decoding of the repetition code $\boldsymbol{G}_R$ is performed by summing up the corresponding L-values after despreading and deinterleaving

$$\boldsymbol{L}_{e,k} = \boldsymbol{G}_R^T \cdot \mathrm{diag}\left(\boldsymbol{s}_k\right) \cdot \boldsymbol{P}_k^T \cdot \boldsymbol{L}_k, \tag{6.16}$$

resulting in a vector $\boldsymbol{L}_{e,k}$ that is passed to the LDPC decoder to complete the iterative detection and decoding process. In addition to performing only one iteration at the multi-user detector, the system can be modified by allowing the multi-user detector to perform more than one iteration between the code symbols and the received signal, which corresponds to multi-stage interference cancellation.

In the following, we will assume a flat fading channel where the fading coefficients are constant during one symbol, i.e. during L chips, and are independently taken from a Rayleigh distribution. Furthermore, we assume that the receiver has perfect knowledge of the channel realizations.

6.5.2 EXIT Functions

We use extrinsic information transfer (EXIT) [AKtB04] functions to analyze the behavior of the multi-user detector and design an LDPC code for the system. For this purpose, we investigate how the EXIT function of the multi-user detector depends on the parameters of the system.

In Figure 6.10 we show the dependency of the EXIT function on the system load $\beta = K/L$ which were obtained using Monte Carlo simulations. The functions are plotted for the system loads of 1, 2, 4 and 8 at a signal to noise ratio $E_s/N_0 = 0.0$dB, where E_s denotes the energy per transmitted code symbol, i.e. the energy per L chips. The spreading length was $L = 4$ and the multi-user detector was allowed to perform five iterations. It can be observed that for small values of $I_{a,MUD}$ the EXIT curve is dominated by the multi-user interference. In contrast, for $I_{a,MUD}$ approaching 1.0, the EXIT curve becomes independent of the system load since the other users are perfectly known and can be canceled.

To analyze the dependency on the signal to noise ratio, we fixed the system load at $\beta = 4$ and simulated the EXIT function of the multi-user detector for increasing signal to noise ratio shown in Figure 6.11. For small values of $I_{a,MUD}$ the influence of the signal to noise ratio is quite small, whereas for large values of $I_{a,MUD}$, the value of the EXIT function is dominated by the signal to noise ratio.

For low signal to noise ratio and/or low system load, the EXIT function of the multi-user detector approaches a horizontal line, i.e. it becomes independent of the a-priori information provided by the LDPC decoder. In this scenario, the multi-user channel can be treated as a single user Gaussian channel (i.e. the noise interference is much stronger than the interference of the other users) and error correcting codes optimized for Gaussian channels can be used. In all other cases, codes have to be

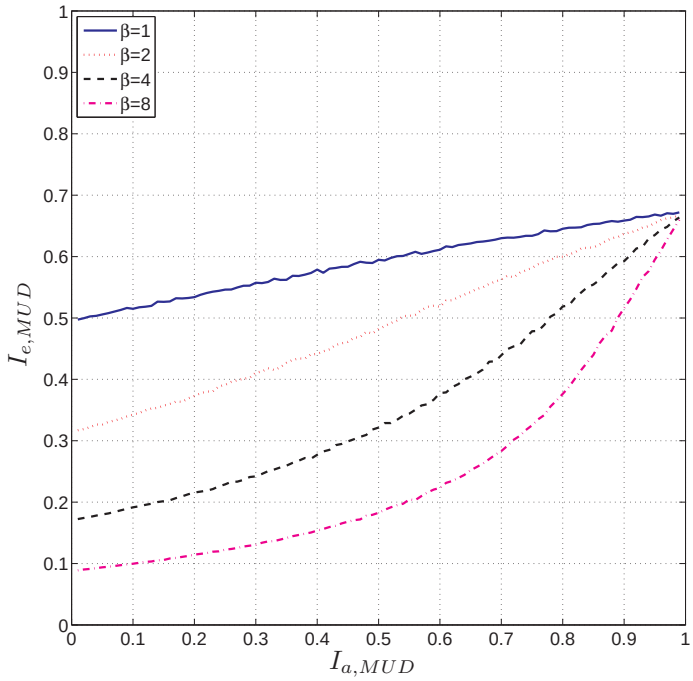


Figure 6.10: EXIT functions of the multi-user detector as a parameter of the system load β for $L = 4$ and $E_s/N_0 = 0.0$dB.

designed for a specific system load and signal to noise ratio in order to approach the capacity of the multi-user channel.

6.5.3 Coding versus Spreading

To compare systems with different parameters we compute the spectral efficiency [VS99]

$$C = \frac{K}{L} R = \beta R, \tag{6.17}$$

where R denotes the rate of the error correcting code. The maximum spectral efficiency for a given load is achieved when the rate of the error correcting code is equal to the capacity C_u for every user. The overall redundancy in the system is split up into spreading (i.e. a repetition code of length L) and into the redundancy added by the LDPC code. To answer the question, how this redundancy should be divided into coding and spreading, we evaluate the spectral efficiency of the system parametrized by the system load. For this purpose we use the area property of EXIT functions [AKtB04], which states that the area below the EXIT function is equivalent to the capacity if the a-priori information is modeled as being transmitted over a binary erasure channel. This area corresponds to the capacity C_u for every

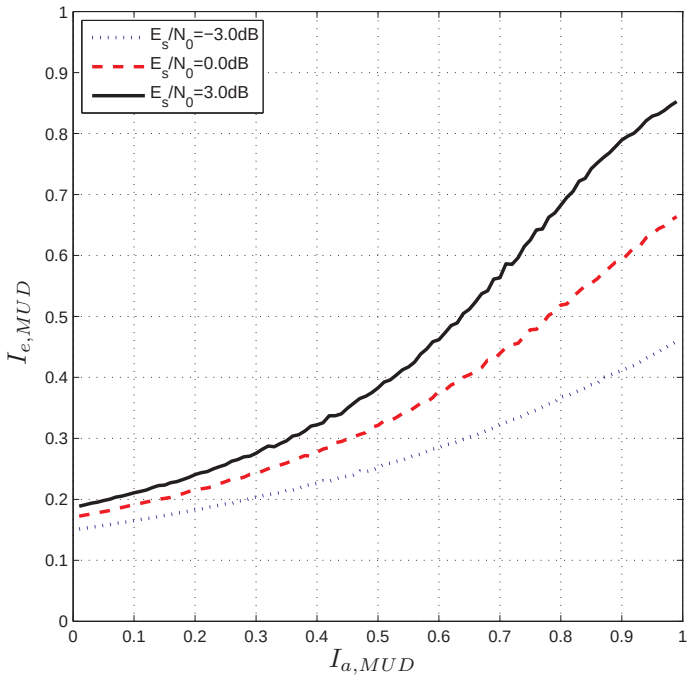


Figure 6.11: EXIT functions of multi-user detector as a parameter of the signal to noise ratio for $L = 4$ and $\beta = 4$.

user.

The maximum achievable spectral efficiency is shown in Figure 6.12 for system loads of 1, 2, 4 and 8 as a function of E_b/N_0. It can be observed, that the spectral efficiency increases with increasing system load. Therefore, from a theoretical point of view, all the redundancy should be spent for coding and no spreading should be performed.

However, not using spreading results in an EXIT function of the multi-user detector that depends heavily on the number of users in the system which is equivalent to the system load in the case of no spreading. From a practical point of view, spreading can be used to keep the system load approximately constant and design an LDPC code for this load.

6.5.4 LDPC Code Design

To optimize the code in a multi-user scenario (i.e. maximizing the code rate for a given signal to noise ratio), we use the methods presented in Section 6.3. In order to obtain codes with practical parameters, we restricted the variable node degree distribution to three non-zero parameters (2, 3 and 10) and set the maximum check node degree to 30. The optimization of the variable node degree distribution was

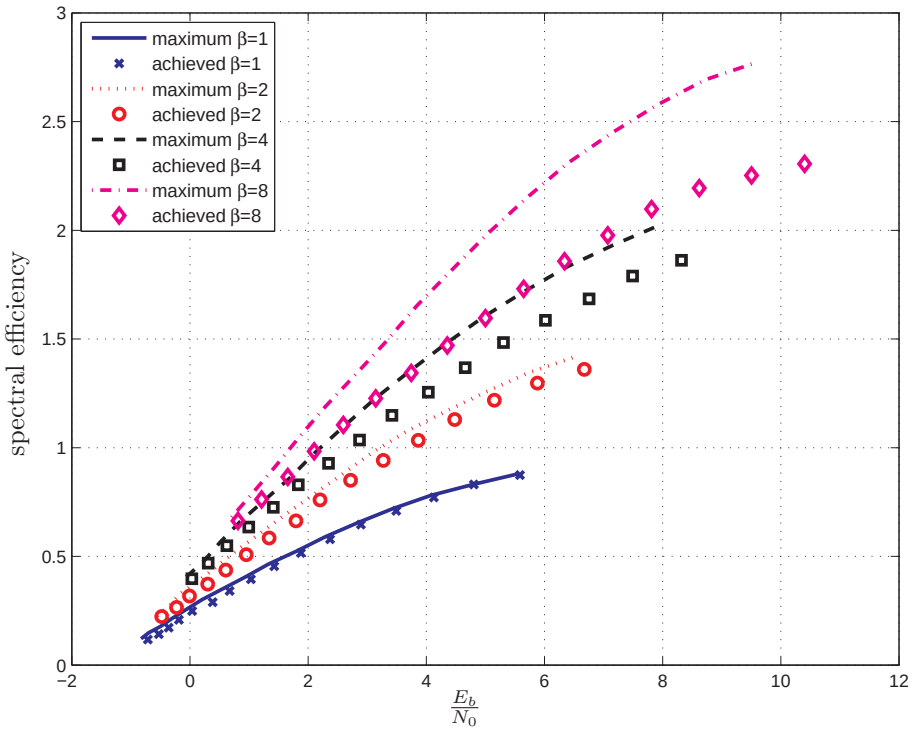


Figure 6.12: Spectral efficiency and results of code optimization.

performed by an exhaustive search while for every distribution, the check node degree distribution was optimized using linear programming. The resulting achieved spectral efficiency of the designed systems is shown in Figure 6.12. Systems with higher load always outperform those with smaller load. For small loads the achieved spectral efficiency of the designed systems is close to the maximum achievable spectral efficiency. For increasing load and increasing signal to noise ratio, the gap increases. The reason for that is, that the shape of the EXIT function of the multiuser detector cannot be matched by the LDPC code with the given parameters (see for example, the EXIT function at $E_s/N_0 = 3.0$dB in Figure 6.11).

6.5.5 Simulation Results

To verify our derivations, we designed a system for a load of $\beta = 4$ and a spectral efficiency of 1.0, leading to a code rate $R = 0.25$. The EXIT functions of the multiuser detector and the optimized LDPC code for this system are shown in Figure 6.13. From the EXIT chart analysis, the threshold of this system is at $E_b/N_0 = 2.75$dB. We constructed a random LDPC code with the given degree distributions and a block length of $N = 10^4$. The bit error rate simulation shown in Figure 6.14 shows that the simulation agrees with the EXIT chart prediction. In Figure 6.13 we also

plotted the EXIT function of an LDPC code that was optimized for an AWGN channel [Amr]. This system cannot converge for the given load since the EXIT functions always intersect (even for the case of very large signal to noise ratio).

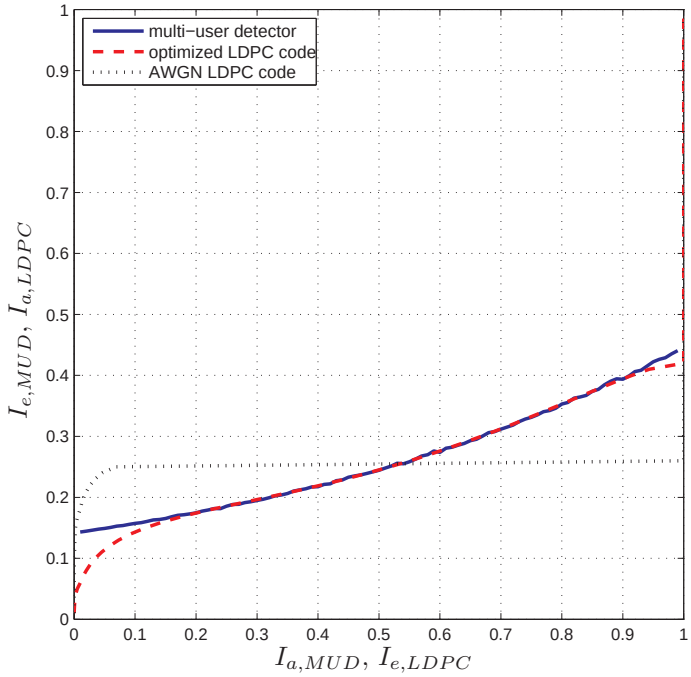


Figure 6.13: EXIT function of multi-user detector and LDPC decoder for $\beta = 4$ and $E_b/N_0 = 2.75$dB.

6.6 Conclusions

We showed how the overall EXIT function of an LDPC decoder can be derived from its own EXIT chart using a Gaussian approximation. Optimizing this function, i.e. maximizing the rate of the code with constraints on its EXIT function, turned out to be a nonlinear problem for the optimization of the variable degree distribution, but is linear in the check node degree distribution. In contrast to conventional code design, where only the variable node distribution is optimized, we proposed to also optimize the check node distribution, resulting in a less complex code search and offering more degrees of freedom for the choice of the variable node distribution.

This code optimization technique was applied to turbo-demapping as an example, and the results were verified by bit error rate simulations, demonstrating the importance of code optimization in iterative receivers.

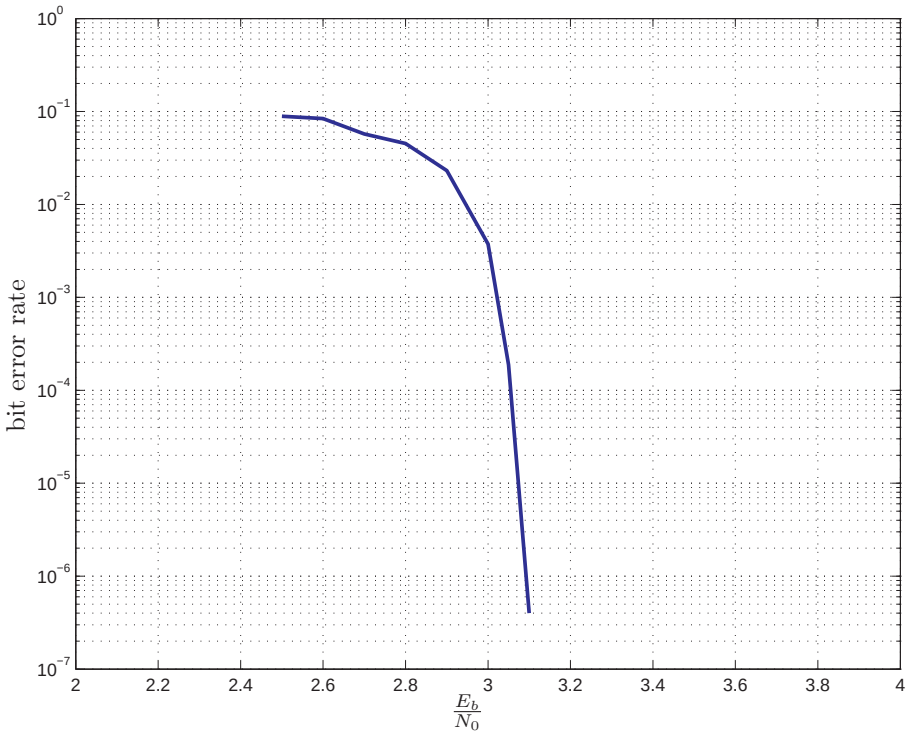


Figure 6.14: Bit error rate simulation for design example with $\beta = 4$ and threshold at 2.75dB.

Furthermore, we applied this code optimization technique to joint multi-user detection and decoding. We showed how the EXIT function of a multi-user detector depends on the load and the signal to noise ratio. By using the area property of EXIT charts, we compared the achievable spectral efficiency of systems of different loads. For the case where the multi-user interference is stronger than the interference caused by Gaussian noise, an LDPC code optimized for this scenario can significantly outperform a standard LDPC code that is optimized for an AWGN channel.

7 Conclusions

In this thesis, we investigated methods to reduce the implementation complexity of iterative decoders for LDPC codes. We presented a detailed analysis of the min-sum algorithm and its improvement by means of post-processing. The post-processing function was derived by re-interpreting the minimum as an observation and computing the a-posteriori L-value of the code digits given this observation. With this analysis, we put some heuristic methods of post-processing—namely linear and offset post-processing—on a solid scientific basis.

For regular codes, we presented efficient methods and provided analytical functions to derive the optimal correction terms. For irregular codes, we showed that the post-processing function has either to be changed with the number of iterations or a nonlinear post-processing function has to be applied in order to achieve a good decoding threshold and to avoid an error-floor. In both cases we reduced the gap between the optimal sum-product and the sub-optimal min-sum algorithm to approximately 0.1dB.

For high-throughput and low-complexity implementations of LDPC decoders, we analyzed binary message-passing algorithms, that use only hard decisions for all internal messages. The EXIT chart analysis of these algorithms led to the well known Gallager A and B algorithms. With our analysis we were able to extend these algorithms to the case where the decoder can make use of a finer quantized channel observation. We showed that a quantization scheme using four values, i.e. two bits of quantization, already leads to a significant improvement of the performance of the decoder of approximately 1.0dB. The gap to an unquantized channel observation is only approximately 0.25dB which indicates that a larger quantization alphabet will not lead to significant improvements.

Finally, we considered joint receiver tasks, where the LDPC decoder was not considered for its own, but in the context of a whole receiver structure which performs additional tasks like demapping or multi-user detection. We showed how the degree distributions of an irregular LDPC code can be optimized to match the EXIT function of a receiver front-end. In contrast to conventional code design, where only the variable node distribution is optimized, we proposed to also optimize the check node distribution, resulting in a less complex code search and offering more degrees of freedom for the choice of the variable node distribution. Since, the optimization of the check node distribution turned out to be a linear problem, we were able to

use efficient optimization techniques in order to perform the overall optimization.

We applied this code optimization to turbo-demapping and joint multi-user detection and decoding. In both cases, the performance of the overall system was improved compared to a standard LDPC code that was optimized for an AWGN channel. Since both—the demapper and the multi-user detector—can be included in the framework of the LDPC decoder, this leads to a reduction of the implementation complexity of the whole receiver. Furthermore, the improved performance allows trade-offs between complexity and performance.

A Derivation of the Post-Processing Function

In this chapter, we present the derivation of the post-processing function (4.7) of Chapter 4.

Consider d_c L-values L_i $(i = 1, \ldots, d_c)$ that are associated to binary random variables $X_i \in \{+1, -1\}$ that satisfy a parity-check equation $\prod_{i=1}^{d_c} X_i = 1$. The conditional probability density function $p(L_i = \zeta | X_i = x_i)$ satisfies

$$p(L_i = \zeta | X_i = +1) = p(L_i = -\zeta | X_i = -1) \hat{=} q_i(\zeta). \tag{A.1}$$

We define the integrals

$$\begin{aligned} & \int_{|a|}^{\infty} \left[p(L_i = \zeta | X_i = x_i) + p(L_i = -\zeta | X_i = x_i) \right] d\zeta \\ = & \int_{|a|}^{\infty} \left[p(L_i = -\zeta | X_i = -x_i) + p(L_i = \zeta | X_i = -x_i) \right] d\zeta \\ = & \int_{|a|}^{\infty} \left[q_i(\zeta) + q_i(-\zeta) \right] d\zeta \hat{=} \gamma_{i+}(a), \end{aligned} \tag{A.2}$$

$$\begin{aligned} & \int_{|a|}^{\infty} \left[p(L_i = \zeta | X_i = x_i) - p(L_i = -\zeta | X_i = x_i) \right] d\zeta \\ = & \int_{|a|}^{\infty} \left[p(L_i = -\zeta | X_i = -x_i) - p(L_i = \zeta | X_i = -x_i) \right] d\zeta \\ = & \; x_i \int_{|a|}^{\infty} \left[q_i(\zeta) - q_i(-\zeta) \right] d\zeta \hat{=} x_i \gamma_{i-}(a) \end{aligned} \tag{A.3}$$

and their derivatives

$$\frac{\partial}{\partial a} \gamma_{i+}(a) = -\operatorname{sgn}(a) \left[q_i(a) + q_i(-a) \right], \tag{A.4}$$

$$\frac{\partial}{\partial a} \gamma_{i-}(a) = -\left[q_i(a) - q_i(-a) \right]. \tag{A.5}$$

Let Z be defined as

$$Z = \min_{i=1}^{d_c - 1} |L_i| \cdot \prod_{i=1}^{d_c - 1} \operatorname{sgn}(L_i). \tag{A.6}$$

The joint probability that $|Z| > |z|$ and $S = \text{sgn}(Z) = s$, conditioned on X_{d_c} can be written as

$$\begin{aligned} Pr(|Z| > |z|, S = s | X_{d_c} = x_{d_c}) &= \frac{1}{2}\left\{ \prod_{i=1}^{d_c-1} \gamma_{i+}(z) + s \prod_{i=1}^{d_c-1} x_i \gamma_{i-}(z) \right\} \\ &= \frac{1}{2}\left\{ \prod_{i=1}^{d_c-1} \gamma_{i+}(z) + s x_{d_c} \prod_{i=1}^{d_c-1} \gamma_{i-}(z) \right\}, \end{aligned} \tag{A.7}$$

where we used $\prod_{i=1}^{d_c-1} x_i = x_{d_c}$ due to the parity-check constraint. The derivative of (A.7) is

$$\begin{aligned} &\frac{\partial}{\partial z} \; Pr(|Z| > |z|, S = s | X_{d_c} = x_{d_c}) \\ &= \frac{1}{2}\left\{ \sum_{j=1}^{d_c-1} \frac{\partial \gamma_{j+}(z)}{\partial z} \prod_{i=1; i\neq j}^{d_c-1} \gamma_{i+}(z) + s x_{d_c} \sum_{j=1}^{d_c-1} \frac{\partial \gamma_{j-}(z)}{\partial z} \prod_{i=1; i\neq j}^{d_c-1} \gamma_{i-}(z) \right\} \\ &= \frac{1}{2} \sum_{j=1}^{d_c-1} \Bigg\{ -\text{sgn}(z) \left(q_j(z) + q_j(-z)\right) \prod_{i=1; i\neq j}^{d_c-1} \gamma_{i+}(z) \\ &\qquad - s x_{d_c} \left(q_j(z) - q_j(-z)\right) \prod_{i=1; i\neq j}^{d_c-1} \gamma_{i-}(z) \Bigg\}. \end{aligned} \tag{A.8}$$

The cumulative distribution function $Pr(Z < z | X_{d_c} = x_{d_c})$ for positive values of z can be derived as

$$\begin{aligned} Pr(Z < z | X_{d_c} = x_{d_c}) &= Pr(Z < z, S = +1 | X_{d_c} = x_{d_c}) \\ &\quad + Pr(Z < z, S = -1 | X_{d_c} = x_{d_c}) \\ &= Pr(|Z| < |z|, S = +1 | X_{d_c} = x_{d_c}) \\ &\quad + Pr(S = -1 | X_{d_c} = x_{d_c}) \\ &= Pr(S = +1 | X_{d_c} = x_{d_c}) \\ &\quad - Pr(|Z| > |z|, S = +1 | X_{d_c} = x_{d_c}) \\ &\quad + Pr(S = -1 | X_{d_c} = x_{d_c}) \\ &= 1 - Pr(|Z| > |z|, S = +1 | X_{d_c} = x_{d_c}), \end{aligned} \tag{A.9}$$

and for negative values of z

$$\begin{aligned} Pr(Z < z | X_{d_c} = x_{d_c}) &= Pr(Z < z, S = -1 | X_{d_c} = x_{d_c}) \\ &\quad + Pr(Z < z, S = +1 | X_{d_c} = x_{d_c}) \\ &= Pr(|Z| > |z|, S = -1 | X_{d_c} = x_{d_c}). \end{aligned} \tag{A.10}$$

The derivative of (A.9) and (A.10) leads to

$$\begin{aligned} p(Z = z|X_{d_c} = x_{d_c}) &= \frac{\partial}{\partial z} Pr(Z < z|X_{d_c} = x_{d_c}) \\ &= \frac{1}{2} \sum_{j=1}^{d_c-1} \Bigg\{ (q_j(z) + q_j(-z)) \prod_{i=1; i\neq j}^{d_c-1} \gamma_{i+}(z) \\ &\quad + x_{d_c}(q_j(z) - q_j(-z)) \prod_{i=1; i\neq j}^{d_c-1} \gamma_{i-}(z) \Bigg\}. \end{aligned} \tag{A.11}$$

Finally, the L-value is obtained using

$$L(z|X_{d_c}) = \log \frac{p(Z = z|X_{d_c} = +1)}{p(Z = z|X_{d_c} = -1)}. \tag{A.12}$$

Bibliography

[AK02] M. Ardakani and F. Kschischang, "Designing irregular LPDC codes using EXIT charts based on message error rate," in *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, 2002, p. 454.

[AK05] ——, "Properties of optimum binary message-passing decoders," *Information Theory, IEEE Transactions on*, vol. 51, no. 10, pp. 3658–3665, Oct. 2005.

[AKtB04] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: model and erasure channel properties," *Information Theory, IEEE Transactions on*, vol. 50, no. 11, pp. 2657–2673, Nov. 2004.

[Amr] A. Amraoui, "LdpcOpt - a fast and accurate degree distribution optimizer for LDPC code ensembles." [Online]. Available: http://lthcwww.epfl.ch/research/ldpcopt/

[BCJR74] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *Information Theory, IEEE Transactions on*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2, 23-26 May 1993, pp. 1064–1070.

[CF02a] J. Chen and M. P. C. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Communications Letters*, vol. 6, no. 5, pp. 208–210, May 2002.

[CF02b] ——, "Near optimum universal belief propagation based decoding of low-density parity check codes," *Communications, IEEE Transactions on*, vol. 50, no. 3, pp. 406–414, Mar. 2002.

[CRU01] S.-Y. Chung, T. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approxima-

tion," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[CT91] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 1991.

[DDP01] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on density evolution," *Selected Areas in Communications, IEEE Journal on*, vol. 19, no. 5, pp. 891–907, May 2001.

[EGH01] H. El Gamal and J. Hammons, A.R., "Analyzing the turbo decoder using the Gaussian approximation," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 671–686, Feb. 2001.

[Gal62] R. Gallager, "Low density parity check codes," *Information Theory, IEEE Transactions on*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[Gal63] ——, *Low Density Parity Check Codes*, ser. Research monograph series. Cambridge, Mass.: MIT Press, 1963, no. 21.

[HEA05] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *Information Theory, IEEE Transactions on*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[HOP96] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *Information Theory, IEEE Transactions on*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[Hu] X.-Y. Hu, "Software for PEG code construction." [Online]. Available: http://www.inference.phy.cam.ac.uk

[ITU04] ITU-T G.975.1, "Forward error correction for high bit-rate DWDM submarine systems," 2004.

[KFL01] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[Lan05] I. Land, "Reliability information in channel decoding – practical aspects and information theoretical bounds," Ph.D. dissertation, University of Kiel, Germany, 2005. [Online]. Available: http://e-diss.uni-kiel.de/diss_1414/

[LB07] G. Lechner and A. Burg, "Design of spatially multiplexed LDPC codes for multi-user detection," in *Proc. 15th European Signal Processing Conference EUSIPCO 2007, Poznan, Poland*, 2007.

[LBSR04] G. Lechner, A. Bolzer, J. Sayir, and M. Rupp, "Implementation of an LDPC decoder on a vector signal processor," in *Proc. Asilomar Conference on Signals, Systems, and Computers, Monterey, CA, USA*, 2004.

[Lec04a] G. Lechner, "Improved sum-min decoding of LDPC codes," in *Proc. 3. ITG-workshop "Angewandte Informationstheory", Ulm, Germany*, 2004.

[Lec04b] ——, "Postprocessing of suboptimal decoding," in *Proc. Kommunikation im Kleinwalsertal, Kleinwalsertal, Austria*, 2004.

[Lec05] ——, "Suboptimal components: How to cook well with cheap ingredients," in *Proc. Winterschool on Coding and Information Theory, Bratislava, Slovakia*, 2005.

[Lec06] ——, "Optimization of LDPC codes for receiver frontends," in *Proc. 15th Joint Conference on Communications and Coding (JCCC), Sölden, Austria*, 2006.

[LM03] F. Lehmann and G. Maggio, "Analysis of the iterative decoding of LDPC and product codes using the Gaussian approximation," *Information Theory, IEEE Transactions on*, vol. 49, no. 11, pp. 2993–3000, Nov. 2003.

[LMSS01] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 569–584, Feb. 2001.

[LPK07] G. Lechner, T. Pedersen, and G. Kramer, "EXIT chart analysis of binary message-passing decoders," in *Proc. ISIT 2007, International Symposium on Information Theory, Nice, France*, 2007.

[LS04] G. Lechner and J. Sayir, "Improved sum-min decoding of LDPC codes," in *Proc. ISITA 2004, International Symposium on Information Theory and its Applications, Parma, Italy*, 2004.

[LS06a] ——, "Improved sum-min decoding for irregular LDPC codes," in *Proc. 4th International Symposium on Turbo Codes and Related Topics, Munich, Germany*, 2006.

[LS06b] ——, "Improved sum-min decoding of irregular LDPC codes using nonlinear post-processing," in *Proc. NEWCOM-ACoRN joint workshop, Vienna, Austria*, 2006.

[LSL06] G. Lechner, J. Sayir, and I. Land, "Optimization of LDPC codes for receiver frontends," in *Proc. ISIT 2006, International Symposium on Information Theory, Seattle, USA*, 2006.

[LSR04] G. Lechner, J. Sayir, and M. Rupp, "Efficient DSP implementation of an LDPC decoder," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004), Montreal, Canada*, May 2004, pp. IV–665–IV–668.

[Mac99] D. MacKay, "Good error-correcting codes based on very sparse matrices," *Information Theory, IEEE Transactions on*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[PSS] M. Peleg, A. Sanderovich, and S. Shamai, "On extrinsic information of good codes operating over discrete memoryless channels," available at http://arxiv.org/pdf/cs.IT/0504028.

[Reg05] P. Regalia, "Iterative decoding of concatenated codes: A tutorial," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 6, pp. 762–774, 2005.

[RSU01] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[RU01] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[SAL06] E. Sharon, A. Ashikhmin, and S. Litsyn, "Analysis of low-density parity-check codes based on EXIT functions," *Communications, IEEE Transactions on*, vol. 54, no. 8, pp. 1407–1414, Aug. 2006.

[SB04] F. Schreckenbach and G. Bauch, "EXIT charts for iteratively decoded multilevel modulation," in *Proc. 12th European Signal Processing Conference (EUSIPCO), Vienna, Austria, September 2004*, 2004.

[SH05] F. Schreckenbach and P. Henkel, "Signal shaping using non-unique symbol mappings," in *Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing, Monticello, USA*, September 2005.

[Sha48] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, July and October 1948.

[SPS05] A. Sanderovich, M. Peleg, and S. Shamai, "LDPC coded MIMO multiple access with iterative joint decoding," *Information Theory, IEEE Transactions on*, vol. 51, no. 4, pp. 1437–1450, Apr. 2005.

[tB99] S. ten Brink, "Convergence of iterative decoding," in *Electronics Letters*, vol. 35, no. 13, Jun. 1999, pp. 1117–1119.

[tBKA04] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *Communications, IEEE Transactions on*, vol. 52, no. 4, pp. 670–678, April 2004.

[VS99] S. Verdu and S. Shamai, "Spectral efficiency of CDMA with random spreading," *Information Theory, IEEE Transactions on*, vol. 45, no. 2, pp. 622–640, Mar. 1999.

[WH00] J. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: an overview," *Vehicular Technology, IEEE Transactions on*, vol. 49, no. 6, pp. 2208–2233, Nov. 2000.

[WP99] X. Wang and H. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *Communications, IEEE Transactions on*, vol. 47, no. 7, pp. 1046–1061, Jul. 1999.