Patrick Grosa

On the Understanding of Protograph-based Low-Density Parity-Check Convolutional Code Design

Beiträge aus der Informationstechnik

Mobile Nachrichtenübertragung
Nr. 71

Patrick Grosa

On the Understanding of Protograph-based Low-Density Parity-Check Convolutional Code Design

A Study on Applicability in Iterative Receiver Systems

VOGT

Dresden 2014

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Bibliographic Information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.

Zugl.: Dresden, Techn. Univ., Diss., 2014

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „On the Understanding of Protograph-based Low-Density Parity-Check Convolutional Code Design“ von Patrick Grosa überein.

Gesetzt vom Autor

ISBN 978-3-938860-82-3

Jörg Vogt Verlag
Niederwaldstr. 36
01277 Dresden
Germany

Phone: +49-(0)351-31403921
Telefax: +49-(0)351-31403918
e-mail: info@vogtverlag.de
Internet : www.vogtverlag.de

TECHNISCHE UNIVERSITÄT DRESDEN

On the Understanding of Protograph-based Low-Density Parity-Check Convolutional Code Design

- A Study on Applicability in Iterative Receiver Systems -

Patrick Grosa

der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieurs

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Univ.-Prof. Dr. Techn. Klaus Janschek
Gutachter:	Prof. Dr.-Ing. Dr. h.c. Gerhard P. Fettweis
	Prof. Dr. Michael Lentmaier

Tag der Einreichung:	7. Juli 2014
Tag der Verteidigung:	14. Oktober 2014

Contents

List of Figures

List of Acronyms

ACE	Approximated Cycle EMD
AWGN	Additive White Gaussian Noise
AR	Accumulate-Repeat
ARA	Accumulate-Repeat-Accumulate
ARJA	Accumulate-Repeat-Jagged-Accumulate
BC	Block Code
BEC	Binary Erasure Channel
BER	Bit-Error Rate
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
CC	Convolutional Codes
CIR	Channel Impulse Response
CPG	Convolutional Protograph
DE	Density Evolution
EMD	Extrinsic Message Degree
EXIT	Extrinsic Information Transfer
FEC	Forward Error Correction
FFT	Fast Fourier Transform

IFFT	Inverse Fast Fourier Transform
i.i.d.	independent and identically distributed
ISI	Intersymbol Interference
LDPC	Low-Density Parity-Check
LDPCC	Low-Density Parity-Check Convolutional
LDPCCC	Low-Density Parity-Check Convolutional Code
LLR	Log Likelihood Ratio
LOS	line-of-sight
LP	Linear Programming
MAP	Maximum a Posteriori
MI	Mutual Information
ML	Maximum Likelihood
MS	Min-Sum
NLOS	Non-line-of-sight
PCM	Parity-Check Matrix
PDF	Probability Density Function
PDP	Power Delay Profile
PEG	Progressive Edge Growth
PG	Protograph
PG-LDPCC	Protograph-based Low-Density Parity-Check Convolutional
PMF	Probability Mass Function
QPSK	Quadrature Phase Shift Keying
QAM	Quadrature Amplitude Modulation

RCPC	Rate-Compatible Punctured Convolutional
SISO	Soft-Input Soft-Output
SNR	Signal-to-Noise Ratio
TC	Turbo Code
TP	Turbo Principle
TG	Tanner Graph
TCPG	Terminated Convolutional Protograph
UWB	Ultra-Wideband

List of Notations

L_p	a posteriori log-likelihood ratio
I_p	a posteriori mutual information
L_a	a-priori log-likelihood ratio
I_a	a-priori mutual information
L_e	extrinsic log-likelihood ratio
I_e	extrinsic mutual information
P	permutation size
m_s	syndrome former memory
L	termination length

Abstract

The optimization process of communication systems having an iterative structure somewhere within the transmission, in the particular case studied, it is in the receiver, is a delicate process. The main intention is to match the behavioral peculiarities of individual components in order to achieve a performance gain through their interaction. However, the component-wise optimization with regard to each other mostly pays off when specific prerequisites, which often lead to trade-offs (if the system has to work for various conditions), are met. Hence, system designers are interested in methods that can enable a straight-forward match while covering a broader set of environments.

With the (recent) rise of the so-called Low-Density Parity-Check codes (LDPCC), a very powerful kind of forward error correction codes has entered almost every recent wireless communication standard. Though these codes have widely known benefits, the researcher's desire for improvement (e. g., reducing gap to capacity, reducing complexity) is without limits. This eventually leads to the application of paradigms, which were previously established for other codes, to LDPCC and the introduction of LDPC Convolutional Codes (LDPCCC).

This thesis tries to tackle the issue of straight-forward matching of a decoder to an equalizer, in a so-called turbo receiver, by the utilization of a particular type of LDPCCC. This kind of code is derived from well-studied structural templates of block code variants, named protographs, through a multi-step derivation process. The steps involved are studied with respect to parameters affecting the code and its performance. As a first step, the basic protograph is unwrapped to a composition graph that reflects the structural distinctiveness of convolutional codes of band parity-check matrix. Here, the syndrome former memory and the termination length are the tunable variables that have behaviors that partially counteracts. Then, an adaptation of the code to the equalizer is feasible, but only within certain limits. The extrinsic information gained for small to medium amounts of a-priori information can be raised by increasing the syndrome former memory, due to the

introduction of lower degree nodes that can provide higher reliability. In turn, the enlargement of the composition graph lowers this by installing higher degree nodes between the lower degree nodes at the ends of the graph. Technically speaking, the termination length of the composition graph, which is referred to as the Terminated Convolutional Protograph, increases. However, they help reduce the rate loss induced by the addition of low degree nodes. A recommendation of the parameter set preferred depends on the operating point suggested.
In the next step, the TCPG is *lifted* to the final code (that are named Protograph-based Low-Density Parity-Check Convolutional **(PG-LDPCC)** codes), where the nodes are duplicated and the edges are permuted. Here, the size of these permutations is important since it, along with the other two variables mentioned before as well, is related to the final block size. Eventually, the system designer has a set of conditions to be satisfied along with and a set of tools for creating codes spanning a design space and has to choose from it. For such a code design challenge, ensemble maps are introduced in order to provide the overview tool at hand.
Furthermore, the opportunity to adapt to certain prerequisites is extended to the study of puncturing methods for the special case of PG-LDPCCCs. Since the derivation process is multi-tiered, puncturing can be applied to different code representations at each stage. In light of this multi-tiered challenge, the error probability distribution over the graph is examined, in particular its fixed point behavior. A modification of extrinsic transfer curve is also feasible by puncturing, but the influence of the puncturing scheme on the transfer curve is also very limited.

Zusammenfassung

Die Optimierung von Kommunikationssystemen mit rückgekoppelten Funktionseinheiten ist eine komplexe Herausforderung. Die Hauptaufgabe dabei ist es, die besonderen Eigenheiten der Komponenten so aufeinander abzustimmen, dass sie bei ihrer Interaktion einen Leistungsgewinn des Gesamtsystems erzielen. Aber, in den meisten Fällen ist eine derartige Optimierung davon geprägt, dass diese nur für bestimmte Rahmenbedingungen optimal ist, was letztendlich zu Abwägungen führt, falls die Komponenten auch unter verschiedenen Bedingungen arbeiten sollen. Aus diesem Grund sind Systemdesigner sehr interessiert an Möglichkeiten, um diese Abstimmung auf einfache Weise durchzuführen und damit eine größere Bandbreite an Randbedingungen abzudecken.
Obwohl Low-Density Parity Check codes (LDPCC) sehr leistungsfähige Vorwärtsfehlerkorrekturverfahren sind und mittlerweile eine so große Verbreitung gefunden haben, dass sie Teil fast jeden neuen Kommunikationsstandards sind, wurde und wird nach weiteren Verbesserungsmöglichkeiten geforscht. Eine Option der Fortentwicklung nutzt die Anwendung bereits bekannter Paradigma auf diese Art der Fehlerkorrektur was zu den sogenannten LDPC Convolutional Codes (LDPCCC) führte.
In der vorliegenden Arbeit wird das Problem eines iterativen Empfängers angegangen, der einen Dekoder unter Benutzung von Low-Density Parity-Check Convolutional Code **(LDPCCC)** auf einfache Art und Weise an einen Entzerrer anpasst. Da beide Komponenten auf Empfängerseite iterativ rückgekoppelt zu finden sind, wird dies auch als Turbo Empfänger bezeichnet. Dabei werden die verwendeten Codes von gut untersuchten strukturellen Vorlagen, genannt Protographen, in einem mehrstufigen Verfahren abgeleitet und die einzelen Schritte werden hinsichtlich der zur Verfügung stehenden Parameter und der daraus resultierenden Leistungsfähigkeit untersucht. Im ersten Schritt, wird der Ausgangsgraph vervielfacht und neu neusortiert zu einer Anordnung, welche die Faltungsstruktur der Paritätsmatrix verdeutlicht. In diesem Schritt sind die syndrombeeinflussende Gedächtnislänge

und die Terminierungslänge von großer Bedeutung und zeigen teilweise entgegenarbeitende Verhaltensweisen. An dieser Stelle ist eine Anpassung zum Entzerrer möglich, jedoch nur begrenzt. Die extrinsische Information, die mittels kleiner bis mittlerer a-priori Information gewonnen werden kann, vergrößert sich mit steigender syndrombeeinflussender Gedächtnislänge durch die Einführung von niedrigen Knotengraden, die eine höhere Zuverlässigkeit bereitstellen können. Im Gegensatz dazu wird dieser Gewinn durch Vergrößerung des Graphen, der Terminierter Faltungsprotograph (TFPG) genannt wird, durch eine steigende Terminierungslänge, mit der höhergradige Knoten zwischen den Enden eingeführt wird, reduziert. Allerdings wird damit auch der Ratenverlust veringert, der durch die niedrigen Knotengrade induziert wird. Eine generelle Empfehlung ist daher nicht möglich und hängt vom Arbeitspunkt ab.
In einem nächsten Schritt wird der TFPG zum finalen Kode erhoben, wobei wiederum Knoten dupliziert und Kanten permutiert werden. Hierbei spielt die Permutationsgröße eine entscheidende Rolle, da diese, genauso wie die bereits erwähnten Parameter, die letztendliche Blockgröße beeinflussen. Schlussendlich muss der System Designer die gegebenen Rahmenbedingungen auf den Designraum der Codes abbilden und aus diesem wählen, oder durch einen Algorithmus für unterschiedliche Bedingungen wählen lassen. Zu diesem Zweck werden dem Designer sogenannte *Code Ensemble Maps* an die Hand gegeben.
Zudem wird die Möglichkeit der Punktierung von PG-LDPCCC untersucht um diese für spezifische Randbedinungen anzupassen. Diese Punkturierung kann dabei auf unterschiedlichen Ebenen des Herleitungsprozesses angewendet werden. Zu diesem Zweck werden Fehlerverteilungen über den Graphen hinweg, im Speziellen sein sogenanntes Fixpunktverhalten, untersucht. Auch hier ist eine Anpassung der extrinsischen Transferkurve möglich, aber auch diese besitzt nur eine eingeschränkter Wirkung.

Acknowledgement

The last four years, since I joined the Vodafone Chair, have passed by so quickly and I still cannot believe that I am handing in this piece of work as a closure of this chapter of my life. So many things have happened for the first time, the first "Klassenfahrt" to Lichtenwalde, the first paper acceptance at ICUWB, the first conference in Nanjing, China, the first "Ski-Seminar" organized by Vinay and me, the first large IEEE VTC2013-Spring conference as organizer, and so on.
I got my first impressions of the chair back in 2006, when PhD students were still sitting in room E63 and I was an undergraduate student assistant working on a first Verilog implementation of NOCs. Apart from attending all the lectures available at that time, I also wrote my student thesis and my Diploma thesis here. However, it is a completely different story when you are part of this incredible team. Some of them I knew from my very first semester at TUD, e. g., Fabian, Richard, and Jan. Some of them I got to know, when I was writing my Diploma thesis, e. g., André, Albrecht ,and Michael, and some of them I just got to know, when I was already here, e. g. Vinay, Walter, Rohit, Nikola, Ines, Esther to name a few. All of them are kind of nerdy in their own way (just to be clear, I see myself as being nerdy as well), but at the same time very friendly, open-minded, and eager for what is ahead of them. This is exactly what it makes so special to be among them. I am very glad that I could experience this.
I would need a lot more space to recall all that has happened during the time at the chair. And although there was a lot of work involved, it never felt like it. When I look back, I see lots of interesting opportunities offered; I see very exciting and funny moments; I see the things that I learned; I see the way that I have changed since then; I see all the aspects in which I have not change. However, all of this wouldn't be possible without Gerhard. I appreciate it a lot and thank him for it!
Of course, I would like to thank Wolfgang, my group leader, for the support over the years, the good discussions and long walks.
A special thanks goes to Michael Lentmaier, who did not hesitate to agree to be

my second reviewer, when he was asked. We always had good discussions about our field of LDPCC codes in channel coding, but it was not limited to that. These discussions always lasted longer than expected, but it was very productive and we also found the time for things apart from research.
Additionally, I want to thank my family for all the years of support financially and emotionally; who pushed me at the right moments and were always there for me. This support is priceless even though I always did it "my way".

1 Introduction

1.1 Motivation

Even though digital communication systems have changed dramatically over the last 30 years, Forward Error Correction **(FEC)** coding is an integral part of these systems since the beginning. Driven by Shannon's coding theorem stating that there exists a code having a rate below the capacity and allowing error-free transmission over a noisy channel, researcher have been looking for these codes ever since. Notwithstanding that Gallager proposed a kind of low complexity codes having very good error correction capabilities, algebraic and Convolutional Codes **(CC)** have been the preferred choice for communications systems for many years due to their more straight-forward implementations. Only when Turbo Codes **(TCs)**, cf. [BG96], were invented and Gallager's codes were rediscovered[1] was the limit proposed by Shannon, i. e., the channel capacity, within the grasp of researchers. Since then these codes have been studied extensively, they are very well understood, and hence, are included in every modern communication standard proposed.

Understanding that both codes, TCs and codes known as Low-Density Parity-Check **(LDPC)** codes nowadays, rely on the same basic principle led to a different perspective on the interplay of components of a communication system. The basic idea is that an iterative exchange of extrinsic information by two (or more) components improves the performance of a certain task. Based on this shifted point of view, the so-called Turbo Principle **(TP)** is applied to various systems, e. g., turbo receiver, turbo equalizer, or blind estimators, to name just a few. A lot of work was done in the field of information theory in order to understand how this exchange of reliability information works, and finally, there were tools proposed to measure and quantify this information, and even visualize and study the exchange

[1] There were papers on Gallager's codes in the meantime, but the time was not ripe, before the computational power available reached a certain level.

itself (e. g., Extrinsic Information Transfer **(EXIT)** charts, cf. §2.5.1). Having these tools at hand, they were used to improve the interplay between the components by adapting them with respect to each other and/or for certain surrounding situations, e. g., generator polynomials of CCs in TC, degree distributions of variable and check node components of LDPC decoder, mapping rules for interplay of mapper and decoder under certain channels, etc. However, with the rising success of LDPC codes, some inevitable drawbacks also garnered attention and the idea of applying the main concepts of other well known codes to LDPC codes have been studied, and so-called Low-Density Parity-Check Convolutional **(LDPCC)** codes were introduced in [JZ98]. In recent years, a lot of research has been carried out related to threshold analysis [LTZC10], distance properties ([STL+07], [MPZC08]), decoder adaptations ([uHPL+12], [HPL+13]), and the basic understanding of the information theoretical relation between the Maximum a Posteriori **(MAP)** and the Belief Propagation **(BP)** thresholds of block and convolutional LDPC variants ([LSCZ10], [KRU11], [STS12]).

1.2 Thesis Outline and Contributions

This thesis deals with a particular variant of LDPCC codes, and to be more precise, with LDPCC code ensembles based on so-called Protographs **(PGs)**. The focus of this study lies on the possibility of adapting these kinds of codes in order for them to work well in turbo receiver systems, and in turbo equalization systems[2], in particular. For this reason, a close look at the parameters involved in the derivation of LDPCC codes is necessary. In particular, the adjustable variables that influence the derivation from a specific well-known PG. The choice taken here is to choose the Accumulate-Repeat-Jagged-Accumulate **(ARJA)**-PG ([DJDT05], [DDJA09]) for the reason of its widespread acceptance. Additionally, this PG has been studied for some time (first by extensive search, later through correspondences to known modules) by the authors of the papers mentioned before. While at first the parameters on the PG level from the block code variant to the convolutional variant are examined, the derivation process from the ensemble description to a real code is considered at a later stage.

[2] In general, the other component, the decoder interplays with, is not important for the study itself, since the main focus is on the adaptability of these codes, but it is important for the results evaluated.

Analysis in the previous step can be undertaken by means of tools already in use for the original PG on a higher level, omitting time consuming Monte-Carlo-Simulations, and can be used to predict the error correction performance expected. The latter step is necessary in order to provide recommendations for codes to be chosen based on certain circumstances. Thus, a trade-off , that helps a decision maker choose, is worked out.

At the end, it is examined whether it is possible to improve the performance of codes with the help of puncturing. For this purpose, the distribution of error probabilities throughout the PG are studied, and certain modifications are also attempted.

Although a specific system model with particular channels and a particular turbo equalizer is assumed, the results from this study can be considered as generally valid and are applicable to a wide spectrum of applications, where the only requirement is a turbo receiver structure. This work is the first study on the adaptation of LDPCC codes for iterative receiver structures by means of variation of convolutional protograph and LDPCC derivation parameters. Therefore, the steps of the derivation process are identified - starting from the well-known protographs, which are used mainly for the derivation of LDPC block codes, to the final LDPCC code. In each step the main parameters involved are studied. Furthermore, puncturing can be introduced on several occasions. The overall contributions to the scientific discourse are summarized in Fig. 1.1, where blue boxes represent the starting point, green boxes illustrate the parameters studied w. r. t. their influence on curve shaping and performance, and the orange box reflects a benchmark necessary for the evaluation of the findings, which is open for further research.

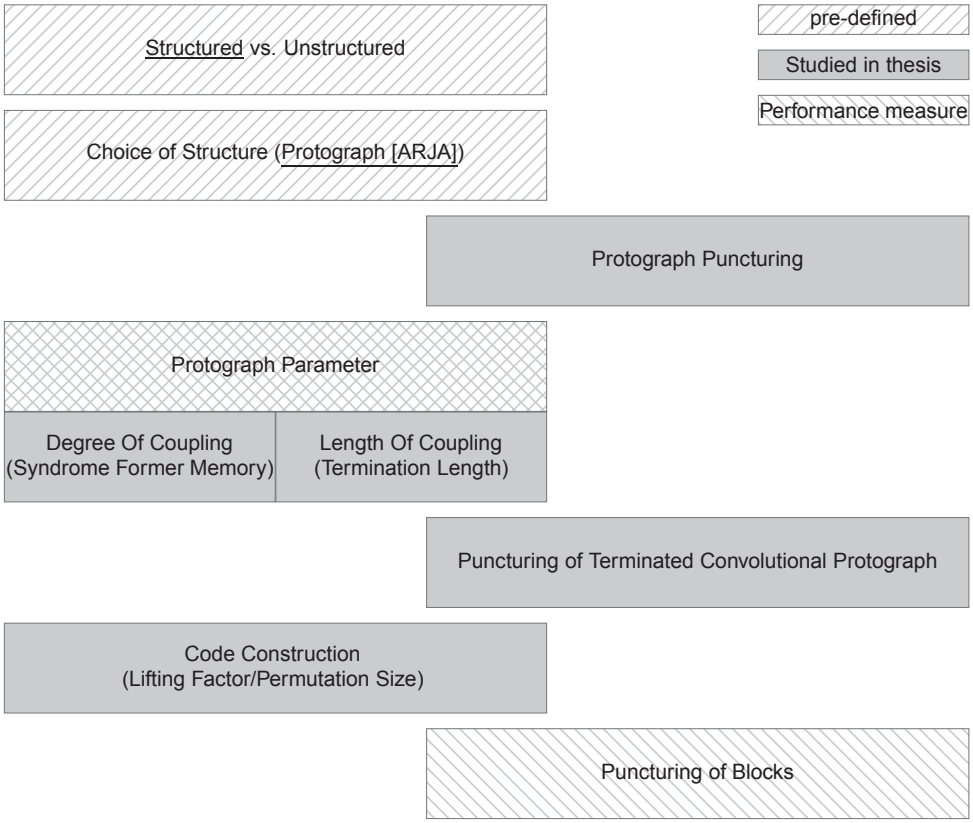


Figure 1.1: Contributions

2 Fundamentals

Communications systems, in general, have to deal with many challenges in order to fulfill their purpose of transmitting data from a transmitter to a receiver in a reliable manner. System designers do not only have to deal with its implementation, but with the theory behind it as well. Starting with the overall system model and highlighting some of its elements, this chapter continues with the main aspects of channel coding and Low-Density Parity-Check codes in particular. Subsequently, a known construction method, called a Protograph, which leads to structured LDPC codes is explained. Finally, the shift from the block code variant to its convolutional variant is shown and explained in detail.

2.1 System Model

The following section will introduce the communication system considered in this thesis. In general, the current work is a continuation of the work in [dS10]. Although the focus of dos Santos' thesis diverges from that of this thesis, the system model remains the same except for components used for blind channel estimation. The basic components on transmitter and receiver side are illustrated and important aspects of particular modules are explained in more detail later on. Additionally, the special role of the channel assumed is highlighted.

2.1.1 Transmission Chain

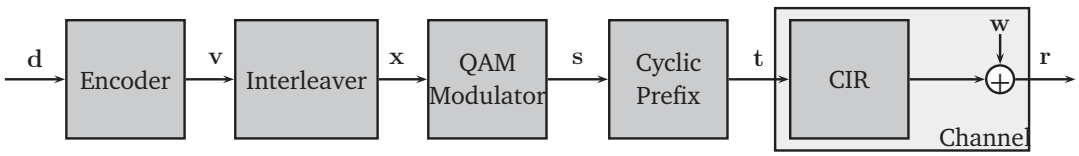


Figure 2.1: Transmitter chain

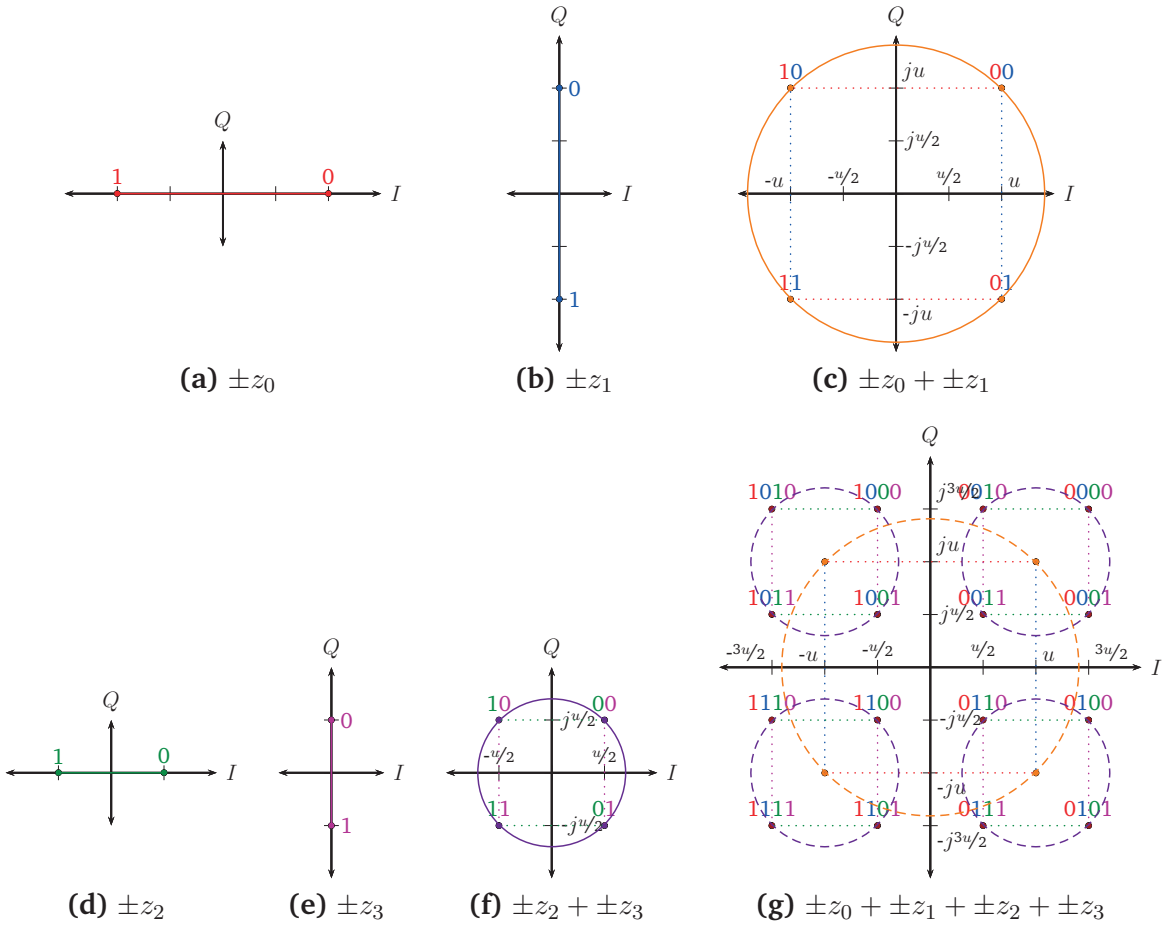
The transmission is carried out in a single carrier fashion with a single antenna on the transmitter side and a single antenna on the receiver side. In Fig. 2.1, the transmission chain including the channel is illustrated. In the first step, a binary information word $\mathbf{d}$ of length K is encoded by means of a forward error correction code resulting in the code word $\mathbf{v}$ of length N. Then, the interleaver permutes the positions of the code word vector in order to ensure independence of the bits[1]. This interleaved code word is then modulated to the symbol vector $\mathbf{s}$ of length $B = \frac{N}{Q}$ with 2^Q-Quadrature Amplitude Modulation **(QAM)**-symbols, where Q is the number of bits per symbol. The actual mapping procedure is explained in greater detail in §2.1.1. Subsequently, a cyclic prefix is prepended in order to use equalization in frequency domain (cf. §2.1.3). Eventually, this transmit vector is transmitted through the channel.
In general, a block-fading multipath channel is assumed with its Channel Impulse Response **(CIR)** given by $\mathbf{h} = [h(0)\ h(1)\ \ldots\ h(l)\ \ldots\ h(T-1)]$. In this context, block-fading means that the CIR does not change during the transmission of one symbol block. Additionally, white Gaussian noise samples $\mathbf{w} \in \mathbb{C}^N$ with variance σ_w^2 are added to the symbol vector. The received samples $r(i)$ at time i are given as

$$r(i) = \sum_{l=0}^{T-1} h(l)t(i-l) + w(i) \qquad . \tag{2.1}$$

Mapping

A mapping method different from Gray mapping is considered, since it is shown, e. g., in [tBS98], that Gray mapping is not optimal for use in turbo equalization. The mapping procedure of choice is accomplished by means of a linear mapping procedure as introduced in [KM03] and is briefly described as follows.

[1] This step is not absolutely necessary for LDPC block codes since connections between different bits in a code word are spread over the whole block. However, in the case of convolutional codes where bits depend on the state of the encoder and the incoming bits, this step is necessary. Since this study deals with convolutional codes for the sake of comparison and considering LDPCC as the objective, it is included in the transmission chain for block codes as well.



(a) $\pm z_0$ **(b)** $\pm z_1$ **(c)** $\pm z_0 + \pm z_1$

(d) $\pm z_2$ **(e)** $\pm z_3$ **(f)** $\pm z_2 + \pm z_3$ **(g)** $\pm z_0 + \pm z_1 + \pm z_2 + \pm z_3$

Figure 2.2: Illustration of Mapping Function Elements

The mapping procedure is carried out in two steps using linear function:

1. Map the interleaved binary code word vector $\mathbf{x}$ onto an interleaved antipodal coded code vector $\ddot{\mathbf{x}} = [\ddot{x}_0, .., \ddot{x}_n, ..\ddot{x}_{N-1}]^T$ using

$$\ddot{\mathbf{x}} = \mathbf{1} - 2\mathbf{x} \quad , \tag{2.2}$$

   where $\mathbf{1}$ is the all-one vector of length N.

2. Map the antipodal vector $\ddot{\mathbf{x}}$ onto the symbol vector $\mathbf{s} = [s_0, .., s_b, ..s_{B-1}]^T$ with $z_q \in \mathbb{S}$ (mapping alphabet $\mathbb{S}$) using

$$s_b = \mathbf{z}^T \underline{\ddot{\mathbf{x}}}_b \quad , \tag{2.3}$$

   where $\underline{\ddot{\mathbf{x}}}_b = \left[\ddot{x}_{bQ}, .., \ddot{x}_{(b+1)Q-1}\right]^T$ and the elements of the linear weighting vec-

tor $\mathbf{z} = \left[z_0, .., z_q, .., z_{Q-1}\right]$ of length Q are given by

$$z_q = \begin{cases} 2^{\frac{Q-q-4}{2}} u & \text{, if } q \text{ is even} \\ j2^{\frac{Q-q-3}{2}} u & \text{, if } q \text{ is odd} \end{cases} . \tag{2.4}$$

The normalization factor u is chosen such that $\mathbf{z}^T\mathbf{z} = 1$.

2.1.2 Channel

Although the channel is included in Fig. 2.1, a separate section is needed to address this issue for the reasons that follows. When the author's work on this topic started, the construction of channel codes in the turbo equalization process for UWB-channels was the main focus. The extremely wide bandwidth needs a high sampling rate and, therefore, has a high resolution in time in order to capture the energy of the signal, which would otherwise be treated as noise. Hence, higher resolution in time enables multiple paths of the signal, from the transmitter to the receiver, to be distinguished. Especially in the case of a UWB channel, a significant part of the energy, relative to the sampling rate, might be received very late.
An example of a typical UWB-NLOS-channel, as defined in [IEE07][2], is used throughout the thesis. The PDP is illustrated in Fig. 2.3. This channel has around three

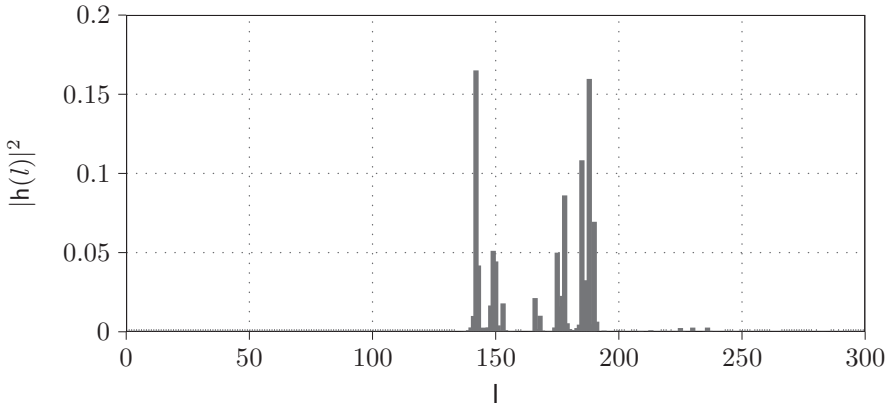


Figure 2.3: PDP of UWB-NLOS channel

hundred taps that must be considered, whereas only a few carry the energy of the

[2]This channel is chosen, because the starting point for the study is based on a German Science Foundation project studying UWB.

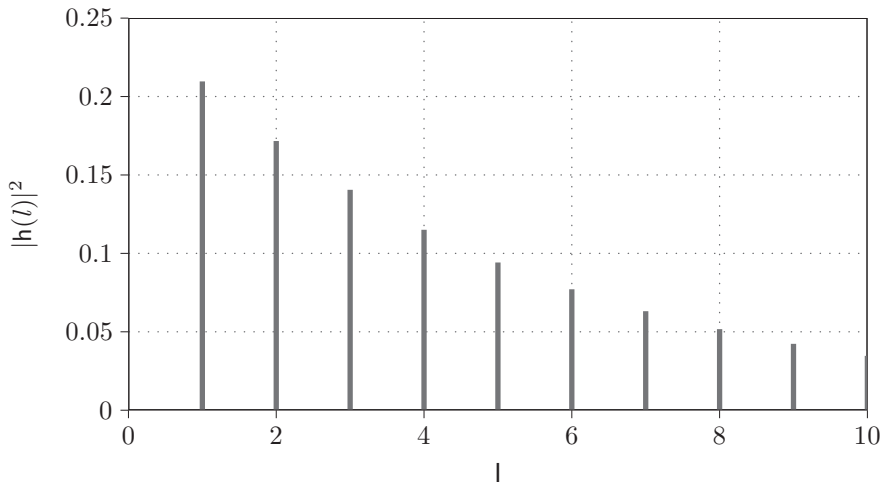


Figure 2.4: 10 tap exponential channel

signal transmitted, which yields to a relatively flat channel and low Intersymbol Interference **(ISI)**. It is a very typical example of a UWB channel, and hence, for the study of code matching to the channel it makes sense to examine another channel. The counter example regarding the ISI properties must be considered, which is a 10 tap exponentially decaying PDP channel, cf. Fig. 2.4, given by

$$\mathsf{h}(l) = \exp\left(-\frac{l+1}{10}\right) \quad , \; l = 0, .., 9 \qquad . \tag{2.5}$$

The influence of ISI on the process of code matching can be seen in an EXIT-Chart in more detail and will be explained in Example 2.19.

2.1.3 Receiver Chain

Compared to the straightforward transmitter chain, most of the signal processing is done at the receiver side. In general, there are two components, the *soft equalizer* and the *soft-decoder*, interacting with each other. The system will be explained by means of the signal flow through the receiver. Subcomponents are explained in greater detail after a general overview with the aid of Fig. 2.5.
At first, the distorted signal $\mathbf{r}$ is received by the soft equalizer, where it is equalized by means of the statistical knowledge of the channel (cf. PDP in §2.1.2), i. e., the PDP is known to the receiver. The received signal incorporates the overlap of multiple paths, which leads to the well-known Intersymbol Interference effect. Based

on the symbols equalized and the mapping applied, a vector containing the extrinsic Log Likelihood Ratio **(LLR)**-values of the interleaved bits $L_e^E(\hat{\mathbf{x}})$ is determined. Subsequently, this vector is deinterleaved and used as a-priori LLR-values $L_a^D(\hat{\mathbf{v}})$ for the decoder. The decoder[3] computes the a-posteriori LLR-values of the information word estimates $L_p^D\left(\hat{\mathbf{d}}\right)$. These can be used to determine the information word estimated utilizing the mapping $L(y) \in \mathbb{R} \mapsto \{0,1\}$. In a non-iterative system, this would be the last step and there would be an error if the estimate does not match the information word transmitted.

Iterative detection decoding (turbo equalization) has been proposed in [DJB+95] in order to improve the detection performance. For this, a matching of the EXIT-chart curves shall be carried out, which leads to the main topic of this thesis: How can LDPCC codes be engineered in a straight-forward way to generate an optimized/matched EXIT chart curve?

In a turbo receiver, the extrinsic LLR-values of the code word $L_e^D(\hat{\mathbf{v}})$, which are fed back to the soft-equalizer (after interleaving) as a priori information $L_a^E(\hat{\mathbf{x}})$, are computed as well. The soft equalizer can then, in the next iteration, utilize the symbol vector received, the statistical knowledge of the channel, and the a-priori information of the code word to improve the estimate of the code word transmitted, and the decoding process starts all over again. This iterative process can be repeated several times until a certain stopping criteria is satisfied. It is reasonable to define a maximum number of iterations, but allowing it to stop earlier if the information word is decoded correctly.

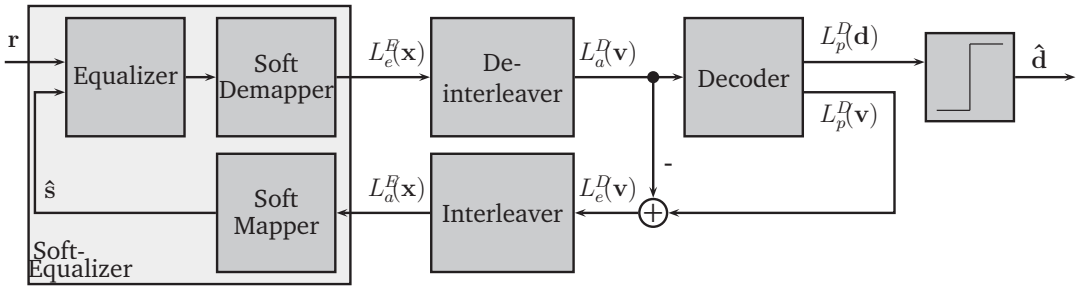


Figure 2.5: Receiver chain

[3]Various decoders are feasible as long as they are recognized as Soft-Input Soft-Output **(SISO)**-decoders, i. e., the decoder has to accept a vector of LLR-values as input and has to yield a vector of LLR-values, e. g., an LDPC decoder or a decoder implementing the BCJR algorithm. These values can be regarded either as a-posteriori or extrinsic values, since there is a distinct relationship between a-priori, a-posteriori and extrinsic values, which are explained later.

Soft Equalizer

The soft equalizer consists of three components that work together, (1) the equalizer, (2) a soft demapper, and (3) a soft mapper. The latter is explained first, since the output of the soft mapper is used for the calculations of the equalizer in each iteration, except for the first one.
In general, the soft mapper tries to calculate the soft symbols $\widehat{s_b}$ and their variances $\mathrm{Var}\left[s_b\right]$, which can be defined as

$$\widehat{s_b} = \mathrm{E}\left[s_b\right] = \sum_{\zeta_i \in \mathbb{S}} \zeta_i \, \mathrm{P}\left(s_b = \zeta_i\right) \tag{2.6}$$

$$\mathrm{Var}\left[s_b\right] = \sum_{\zeta_i \in \mathbb{S}} \left|\zeta_i - \widehat{s_b}\right|^2 \mathrm{P}\left(s_b = \zeta_i\right) \quad , \tag{2.7}$$

where ζ_i are the symbols from the constellation set $\mathbb{S}$. This calculation is based on the a-priori information of the equalizer coming from the channel decoder, since it is the only source of information available. Therefore, as a first step, the so-called 'soft bits' (and their variances) based on the a priori LLR values $L_a^E\left(\mathbf{x}\right)$ (cf. [KST04, OT02, TSK02]) calculated as

$$\widehat{\ddot{x}}_n = \tanh\left(\frac{L_a^E\left(x_n\right)}{2}\right) \tag{2.8}$$

$$\mathrm{Var}\left[\ddot{x}_n\right] = 1 - \left|\widehat{\ddot{x}}_n\right|^2 \quad . \tag{2.9}$$

Remark 2.1 (Interchangeable use of $L\left(\ddot{x}_n\right)$ and $L\left(x_n\right)$). In Eq. 2.8, $L_a^E\left(x_n\right)$ is used instead of $L_a^E\left(\ddot{x}_n\right)$ to define the calculation of the soft bits. This is interchangeable since there is a linear relationship between $\ddot{\mathbf{x}}$ and $\mathbf{x}$. However, it is necessary to distinguish them previously in order to write the equations in a more compact form.

Then, the soft symbols and their variances can be determined using the linear weighting vector from Eq. 2.4 and the soft bits by

$$\widehat{s_b} = \mathrm{E}\left[s_b\right] = \mathrm{E}\left[\sum_{q=0}^{Q-1} z_q \ddot{x}_{bQ+q}\right] = \sum_{q=0}^{Q-1} z_q \, \mathrm{E}\left[\ddot{x}_{bQ+q}\right] \tag{2.10}$$

$$= \sum_{q=0}^{Q-1} z_q \widehat{\ddot{x}}_{bQ+q} \tag{2.11}$$

and

$$\begin{aligned} \mathrm{Var}\left[s_b\right] &= \mathrm{Var}\left[\sum_{q=0}^{Q-1} z_q \ddot{x}_{bQ+q}\right] = \sum_{q=0}^{Q-1} z_q^2 \,\mathrm{Var}\left[\ddot{x}_{bQ+q}\right] && (2.12)\\ &= \sum_{q=0}^{Q-1} z_q^2 \,\mathrm{Var}\left[\ddot{x}_{bQ+q}\right] \quad . && (2.13)\end{aligned}$$

The equalized symbol vector $\mathbf{e}$ can be obtained by

$$\mathbf{e} = (1+\omega\bar{\varrho})^{-1}\left[\bar{\varrho}\widehat{\mathbf{s}} + \mathbf{F}^H\mathbf{\Psi}\mathbf{F}\left(\mathbf{r} - \widehat{\mathbf{H}}\widehat{\mathbf{s}}\right)\right] \quad , \tag{2.14}$$

where $\widehat{\mathbf{H}}$ is the circular convolution matrix using an estimated CIR and $\mathbf{F} \in \mathbb{C}^{B\times B}$ is the Fourier matrix and

$$\begin{aligned} \Psi &= \mathbf{\Xi}^H(\mathbf{\Xi}\mathbf{\Lambda}_a\mathbf{\Xi}^H + \sigma_w^2\mathbf{I})^{-1} \quad , && (2.15)\\ \bar{\varrho} &= \frac{1}{B}\,\mathrm{tr}\left(\mathbf{\Psi}\mathbf{\Xi}\right) \quad , && (2.16)\\ \omega &= \frac{1}{B}\sum_{b=0}^{B-1}\mathrm{E}\left[|\widehat{s}_b|^2\right] \quad . && (2.17)\end{aligned}$$

where $\mathbf{\Xi}$ and $\mathbf{\Lambda}_a$ are the diagonal matrices containing the Fourier transform of the estimated CIR vector and the variances of the soft symbols (cf. Eq. 2.8), respectively.
It is shown in [KM07] that the equalized symbol e_n is approximately a Gaussian random variable[4] $e_n \sim \mathcal{N}\left(\mu_{e_b}, \sigma_{e_b}^2\right)$, where its mean and variance can be determined by

$$\begin{aligned} \mu_{e_b} &= \bar{\varrho}\,(1+\omega\bar{\varrho})^{-1} && (2.18)\\ \sigma_{e_b}^2 &= \mu_{e_n}\left(1-\mu_{e_b}\right) \quad . && (2.19)\end{aligned}$$

The equalizer's extrinsic information can be calculated with help of utilizing the

[4]The assumption of a Gaussian distributed random variable is necessary for the validity of the use of EXIT charts (cf. §2.5.1).

equalized symbols by the max-log-map algorithm shown in [tBS98]:

$$L_e^E(x_n) = \max_{\underline{\ddot{\mathbf{x}}}'_b \in \mathbb{S}^{x_n=+1}} \left\{ \frac{1}{\sigma_{e_b}^2} \left| e_b - \mu_{e_b} \right|^2 + \frac{1}{2} \sum_{i=bQ, i\neq n}^{(b+1)Q-1} \underline{\ddot{\mathbf{x}}}' L_e^E\left(\underline{\ddot{\mathbf{x}}}'\right) \right\} - \ldots$$
$$\ldots \max_{\underline{\ddot{\mathbf{x}}}'_b \in \mathbb{S}^{x_n=-1}} \left\{ \frac{1}{\sigma_{e_b}^2} \left| e_b - \mu_{e_b} \right|^2 + \frac{1}{2} \sum_{i=bQ, i\neq n}^{(b+1)Q-1} \underline{\ddot{\mathbf{x}}}' L_e^E\left(\underline{\ddot{\mathbf{x}}}'\right) \right\} \quad , (2.20)$$

where $\underline{\ddot{\mathbf{x}}}'_b \in \mathbb{S}^{x_n=+1}$ and $\underline{\ddot{\mathbf{x}}}'_b \in \mathbb{S}^{x_n=-1}$ are the bit representations of the corresponding symbols in $\mathbb{S}$ wherein $x_n = +1$ and $x_n = -1$, respectively.

2.2 Low-Density Parity-Check Code Basics

Although LDPC codes were already defined by Gallager [Gal62] in the early 1960s, and a few papers were published, they did not enter the limelight until the 1990s, when their performance being close to the Shannon limit was brought to the public's attention [MN97]. Nowadays, they are very well known by the scientific community and are a part of almost all recent standards. In light of the community's familiarity with LDPC codes, this section is rather short and concentrates on the basic definitions necessary for further explanations of concepts in this thesis.

Definition 2.2 (Low-Density Parity-Check code)**.** An LDPC code is a linear block code $\mathfrak{C}$ of length N, dimension K and rate $\mathcal{R} = \frac{K}{N}$ that can be described by a parity-check matrix $\mathbf{H}$ of dimension $(M \times N)$ with linearly independent rows, where $M = N - K$, and the number of non-zero elements in the matrix $\mathbf{H}$ are small compared to the overall number of elements. The code $\mathfrak{C}$ is then defined as the set of all code words $\mathbf{v} = (v_0, \ldots, v_{N-1})$ that satisfy the equation

$$\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0} \quad , \tag{2.21}$$

where $\mathbf{0}$ is a vector consisting of $M = N-K$ zero elements and $(\cdot)^T$ indicates matrix transposition. Both, the elements of the code word vector $\mathbf{v}$ and the elements of the parity check matrix $\mathbf{H}_{j,i}, j = 0, \ldots, M-1, i = 0, \ldots, N-1$ are taken from the binary Galois field GF(2).

Example 2.3. The following example shows matrix representation of the (7,4)-Hamming code. Although the code is not really sparse, it is a very good, yet efficient example, of a block code.

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

Three parity-check equations are formed by four out of seven bits. This code has a rate $\mathcal{R} = 1 - {}^3\!/_7 = {}^4\!/_7$, can correct one bit errors, or can detect up to two bit errors.

Other properties, e. g., the Hamming weight, Hamming distance and the minimum distance are defined as they are known for linear block codes from literature

(e. g., [RU08]).

The most common graphical representation of LDPC codes is the Tanner Graph **(TG)** [Tan81].

Definition 2.4 (Tanner Graph)**.** A TG, introduced in [Tan81], is a bipartite graph $\mathcal{G}(\mathcal{V}, \mathcal{C}, \mathcal{E})$, that consist of two distinct sets of vertices with cardinalities $\left(\left|\mathcal{V}\right| = N, \left|\mathcal{C}\right| = M\right)$ and a set of edges $\mathcal{E} = \left\{\left(v_i, c_j\right), v_i \in \mathcal{V}, c_j \in \mathcal{C}\right\}$ that connects the vertices. The elements of $\mathcal{V}$ and $\mathcal{C}$ are named variable nodes and check nodes, respectively.

The TG reflects the parity check matrix $\mathbf{H}$ of an LDPC code $\mathfrak{C}$, where the columns correspond to the variable nodes, the rows correspond to check nodes and the elements determine whether or not there is a connection between the nodes, i. e.,

$$\left(v_i, c_j\right) \in \mathcal{E}, \quad \text{if } \mathbf{H}_{j,i} = 1 \quad . \tag{2.22}$$

Example 2.5. Fig. 2.6 shows the graphical representation of Example 2.3.

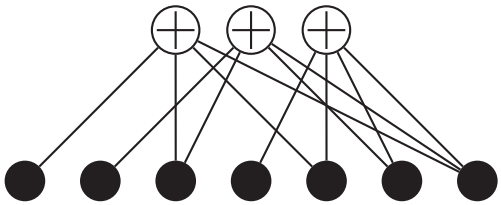


Figure 2.6: Tanner-Graph representation of (4,7)-Hamming Code (black circles: variable nodes, circles with plus: check nodes)

In order to analyze and predict the behavior of an LDPC code, a closer look at the cycles in the graph is helpful.

Definition 2.6 (Cycle)**.** A cycle is a path from a node back to itself via several edges, where no edge is used more than once. The number of edges included in this path is called the length of the cycle. In case of bipartite graphs, e. g., TG, the length is always even.

Definition 2.7 (Girth and Local Girth)**.** The local girth g_i is the shortest possible cycle that includes the variable node v_i. The girth g of an LDPC code $\mathfrak{C}$ is the shortest cycle of all local girths, i. e.,

$$g = \arg\min_i g_i \tag{2.23}$$

If you assume an extensive search of possible code words, which can be considered as a tree-like decoding process, there is no problem arising from cycles. However, such an elaborate method is very expensive w. r. t. memory usage, and hence, all commonly used algorithms for decoding like Belief Propagation **(BP)** or Min-Sum **(MS)** decoding iterate between the same nodes during decoding, and hence, are not tree-like. The price to pay for saving memory is a feedback of information, although it is processed in a way, to the same nodes as they originated. This leads to an error floor in the BER performance especially in high Signal-to-Noise Ratio **(SNR)** regimes. Therefore, the cycles in a graph (length, distribution, etc.) shed light on the performance of the code, since the nodes do not receive just new information but their own (processed) information as well, which can lead to failures (cp. stopping sets [TJVW03] and trapping sets [CD]).

2.2.1 Belief Propagation

As previously mentioned, the great benefit of LDPC codes over many other codes[5] is their superior error correction performance while still having a low complexity decoding algorithm. Although there are a lot of names for each of these decoding algorithms, most of them rely on the same basic idea, namely, Belief Propagation **(BP)**. The idea behind it, is to operate on the TG representation of the code. The variable nodes represent code word bits and the check node represent each check equation. The variable nodes are initialized with the input LLRs of the decoder and improve their reliability by incorporating the information they receive with help of the check equations. The bits, which are involved in each check equation, are indicated by the edges connected to a check node. Hence, the information generated by each node is passed as a *message* along the edges.
The messages passing along the edges from check node c_j to variable v_i and from variable node v_i to check node c_j will be denoted as ${}_sL_e^C\left(\left(v_i, c_j\right)\right)$ and ${}_sL_e^V\left(\left(v_i, c_j\right)\right)$, where s indicates the iteration number. Then the algorithm can be summarized as shown in Alg. 1.

[5] Recently, the idea of polar codes are introduced in [Ari09], that fit to this statement as well, but they have other advantages and disadvantage (e. g., universality, block length restriction, ease of proofs and decoding complexity) and are not part of this thesis.

Alg. 1 Belief Propagation Algorithm

Require: $L_a^D(\mathbf{v})$

for all variable nodes v_i, $i = 0, \ldots, N-1$ **do**
  for all edges $\left(v_i, c_j\right) \in \mathcal{E}$ connected to variable node v_i **do**
    (initialize extrinsic information of variable nodes with channel values)
    ${}_0L_e^V\left(\left(v_i, c_j\right)\right) = L_a^D(v_i)$
while $s < s_{max}$ (and not decoded yet) **do**
  for all check nodes c_j, $j = 0, \ldots, M-1$ **do**
    for all edges $\left(v_i, c_j\right) \in \mathcal{E}$ connected to check node c_j **do**
      (calculate extrinsic information of check nodes by means of
      extrinsic information received from the variable nodes)
      $${}_sL_e^C\left(\left(v_i, c_j\right)\right) = 2\,\mathrm{atanh}\left(\prod_{\left(v_{i'}, c_j\right) \in \mathcal{E} \setminus \left(v_i, c_j\right)} \tanh\left(\frac{{}_{(s-1)}L_e^V\left(\left(v_{i'}, c_j\right)\right)}{2}\right)\right)$$
  for all variable nodes v_i, $i = 0, \ldots, N-1$ **do**
    for all edges $\left(v_i, c_j\right) \in \mathcal{E}$ connected to variable node v_i **do**
      (calculate extrinsic information of variable nodes by means of ex-
      trinsic information received from the check nodes and the channel
      values)
      $${}_sL_e^V\left(\left(v_i, c_j\right)\right) = L_a^D(v_i) + \sum_{\left(v_i, c_{j'}\right) \in \mathcal{E} \setminus \left(v_i, c_j\right)} {}_sL_e^C\left(\left(v_i, c_{j'}\right)\right)$$
for all variable nodes v_i, $i = 0, \ldots, N-1$ **do**
  (calculate a-posteriori and extrinsic information of decoder by means of
  extrinisc information received from the check nodes and the channel
  values)
  $$L_e^D(v_i) = \sum_{\left(v_i, c_j\right) \in \mathcal{E}} {}_{s_{max}}L_e^C\left(\left(v_i, c_j\right)\right)$$
  $$L_p^D(v_i) = L_a^D(v_i) + L_e^D(v_i)$$

2.2.2 LDPC ensembles

Up to this point, it is possible to define an LDPC code either by its graph or by its parity check matrix. Both can also be used to gain some insights into its most likely behavior. Still, it is not possible to predict the performance of a code (in general) without the need for intensive simulations. Therefore, it is beneficial to use methods that simplify the analysis by introducing a more generalized description of a group, or ensemble of codes, and by analyzing this model. These descriptions (degree distributions, Protograph)were found in [Tho05] and can be distinguished by their ability to preserve the structure of the code ensemble description in the derived codes. Hence, the code ensemble descriptions can be classified as *structured* or as *unstructured* code ensemble descriptions. In this section the focus is on the latter, since §2.4.1 deals with a kind of structured code ensemble called PG.

The most common unstructured LDPC code ensemble description averages over all codes having the same variable and check-node degree distributions. In order to understand node degree distributions, it is necessary to define the term *node degree*.

Definition 2.8 (Node Degree)**.** The node degree of a variable node, or a check node is the number of edges connected to the node and is denoted by d_{v_i}, where $i = 0, .., N-1$ for variable nodes and d_{c_j}, where $j = 0, .., M-1$ for check nodes. The minimum (maximum) of all variable node degrees and check node degrees is denoted by $d_{v,\text{min}}$ or $d_{v,\text{max}}$ and $d_{c,\text{min}}$ or $d_{c,\text{max}}$, respectively.

Now, it is possible to define the unstructured LDPC code ensembles by means of polynomials. However, there are two different, but equivalent (in the sense that they can be transformed into each other), perspectives that have to be distinguished.

Definition 2.9 (Degree Distribution from an edge perspective)**.** An LDPC code ensemble can be defined by its degree distributions from an *edge perspective*. The variable node degree distribution is given by

$$\lambda(x) = \sum_{l=d_{v,\text{min}}}^{d_{v,\text{max}}} \lambda_l x^{l-1} \tag{2.24}$$

and the check node degree distribution is given by

$$\rho(x) = \sum_{l=d_{c,\min}}^{d_{c,\max}} \rho_l x^{l-1} \quad , \tag{2.25}$$

where λ_l and ρ_l are the fraction of edges that are connected to variable nodes of degree l and check nodes of degree l, respectively.

Definition 2.10 (Degree Distribution from a node perspective)**.** An LDPC code ensemble can be defined by its degree distributions from a *node perspective*, then the variable node degree distribution is given by

$$\alpha(x) = \sum_{l=d_{v,\min}}^{d_{v,\max}} \alpha_l x^l \tag{2.26}$$

and the check node degree distribution is given by

$$\gamma(x) = \sum_{l=d_{c,\min}}^{d_{c,\max}} \gamma_l x^l \quad , \tag{2.27}$$

where α_l and γ_l are the fraction of variable nodes of degree l and the fraction of check nodes of degree l, respectively.

Performance prediction can be carried out with help of the so-called Density Evolution **(DE)** which is explained in §2.5.

2.2.3 Lifting procedure

As stated earlier, the reasons for using code ensembles instead of the LDPC codes themselves are manifold, e.g., they enable prediction of properties and performances without the need for intensive Monte-Carlo simulations.
The derivation of an LDPC code from any generic code ensemble description is called *lifting* and the procedure can be summarized as follows.

1. Create a predefined number of variable- and check-nodes such that a number of code properties (e. g., block length, rate, etc.) are fulfilled and/or they meet the requirements of the code ensemble suggested.

2. Place and permute the edges such that they meet the requirements (e. g., degree distrubtions, protograph) of the code ensemble suggested.

This procedure will be explained by means of *degree distributions* as code ensemble description, since the course of action is very close to the one used for PGs. Nevertheless, the differences in the modus operandi are explained after a formal introduction of PGs in §2.4.1.

Permutation of Edges

The permutation of edges is a very crucial process, since unfavorable configurations in terms of large number of short cycles within the graph can occur. An exhaustive search of all possible edge configurations that meets the code ensemble requirements is not feasible after all. Therefore, methods to obtain local maxima, which are the local girths, are favored. In literature, there are two widely known methods used for the permutations of edges which are adapted for PGs later on. These algorithms are briefly introduced in the following and their advantages and disadvantages are outlined. The reader is referred to the original papers for further studies of this topic.

Progressive Edge Growth - Algorithm

The Progressive Edge Growth **(PEG)** algorithm is introduced in [EA05] for the degree distribution based code ensembles. Nowadays, the algorithm is widely used, and adapted to meet other requirements, e. g., protograph definitions. The basic idea is to determine the edges connected to each variable node successively, i. e. in a variable node-by-variable node manner, while maximizing the local girth of the node.
Starting with an unconnected variable node, edges are placed in succession in order to connect this node with the graph that has already been built until the designated degree of the node is reached. Therefore, a tree with the current variable node as a root element is spanned before every placement of a new edge. The connections of the tree are based on the current graph configuration. Hence, the edge is placed between the current variable node and the most distant check node available. In this sense, “most distant” means that no cycle, or the largest cycle

possible, is formed.

Remark 2.11. Various strategies can be applied w. r. t. to the order of variable nodes chosen, such as the connections allowed in each step or a choice of the connecting node, if several nodes have the same distance to the root variable node.

The greatest advantage of this method is its straight-forward implementation and its easy adaption to different requirements, e. g., to PGs. However, since it optimizes the local maxima of the girth alone and does not take the distribution of short cycles into account, the algorithm will, most likely, not find best graph configuration possible. The algorithm does not an exhaustive search, but rather maximizing the girth of the current node. Hence, it might be better to avoid a very large cycle in the beginning and may have a very short cycle at the end, in favor for a slightly shorter large cycle in the beginning and a slightly longer short cycle at the end of the algorithm processing.

Approximated Cycle EMD (ACE) - Algorithm

The second method that is introduced briefly at this point is the so-called ACE algorithm [TJVW03]. It addresses some of the issues of the PEG by introducing another metric based on the Extrinsic Message Degree **(EMD)**, which is explained in [TJVW03] as well. This may lead to shorter cycles earlier in the process, if elements of this short cycle are also part of larger cycles. In this case, the nodes receive a large amount of *new* information from other nodes as well. Therefore, shorter cycles are not that harmful, if there are some nodes with high degrees and/or these nodes are involved in larger cycles.

2.3 Low-Density Parity-Check Convolutional Codes

In this section, a very brief definition and explanations of LDPCC codes is given.

2.3.1 Definition

The extension of LDPC codes from block codes to its convolutional counterpart is recapitulated in this section. Both versions can define their set of valid codewords by means of the Parity-Check Matrix **(PCM)** fulfilling the following condition:

$$\mathbf{v}_\infty \cdot \mathbf{H}_{[-\infty,\infty]}{}^T = \mathbf{0}_\infty \quad , \tag{2.28}$$

However, the dimensions of the code words and the PCM can be infinite, which is denoted by the ∞ subscript. Additionally, the PCM has a particular band structure given by:

$$\mathbf{H}_{[-\infty,\infty]} = \begin{bmatrix} \ddots & & \ddots & & & & & \\ & \mathbf{H}_{m_s}(\tau) & \dots & \mathbf{H}_0(\tau) & & & & \\ & & \ddots & & \ddots & & & \\ & & & \mathbf{H}_{m_s}(\tau+T-1) & \dots & \mathbf{H}_0(\tau+T-1) & & \\ & & & & \ddots & & \ddots & \end{bmatrix} . \tag{2.29}$$

In general, each sub matrix $\mathbf{H}_i(t)$ can be different depending on the time instance $t = \tau$ and its dependence on previous time instances $t = \tau - i$, for $i = 0, \dots, m_s$. All matrices have the same dimensions. However, it also reasonable to repeat matrices after a certain number of time instances, which is called the periodicity T, i. e., $\mathbf{H}_i(t+T) = \mathbf{H}_i(t)$. For $T = 1$ the code is called *time-invariant* and *time-variant* otherwise. For $m_s = 0$, the code can be seen as an infinite number of independent LDPC-Block Codes **(BCs)**.

2.3.2 Decoding

Although LDPCC codes are defined to have an infinite PCM, and therefore, an infinite code word, most communication systems use packets for a transmission. This results to make use of a block based approach. This can be modeled by assuming all-zero sub matrices $\mathbf{H}_i(t)$ for $t < t_1$ and $t > t_2$.
Then, there are two different perspectives available for approaching the decoding issue for such a code. On one hand, the code can be treated as a normal block code, where the structure of the PCM does not have any influence on the decoding pro-

cess. Then, decoding can be carried out with the help of an ordinary block based BP decoder, or its related algorithms as well. On the other hand, the decoder can exploit the band structure, i. e. the nodes at the end of the PCM are not directly connected to the nodes at the beginning, but by iterating locally and providing higher reliability information to nodes at the end of the PCM. Such a decoding strategy is called window decoding (e. g., [SPL09], [LPF11]), which subsequently defines a region that is updated for a while before it is changed.
Since the optimal performance should be achieved and it is not absolutely clear (at the moment) what the best choice for the window size is, the block code approach is used throughout this thesis. My colleague and also PhD student is studying this property ([uHPL$^+$12], [uHLF12]).

2.4 Protographs

The term Protograph **(PG)** was used in [Tho05], for the first time, in order to name a graph that describes an LDPC code ensemble, where each code obtained by a *lifting* procedure that has the same structure as the protograph. The basic properties and representations as well as methods for lifting are shown in this section. Furthermore, the concept of PGs can be extended to a form which has many properties in common with CC. Codes that have this nature are the main focus of this thesis and are explained in greater detail.

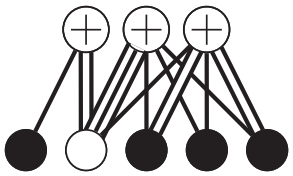
2.4.1 Basic Properties and Representations

Definition 2.12. A PG is a relatively small bipartite multigraph $\mathcal{G}\left(\mathcal{V}_P, \mathcal{C}_P, \mathcal{E}_P\right)$, that consist of two distinct sets of vertices with cardinalities $\left(\left|\mathcal{V}_P\right| = N_P, \left|M_P\right| = M_P\right)$ and a set of edges $\mathcal{E}_P = \left\{\left(v_i, c_j\right), v_i \in \mathcal{V}_P, c_j \in \mathcal{C}_P\right\}$ that connects the vertices.

In comparison to Tanner Graphs **(TGs)** [Tan81], whose description is very similar, the set of edges is rather a multiset than a simple set, i. e. each edge is associated with a multiplicity $\left|\left(v_i, c_j\right)\right| = m, m \in \mathbb{N}^+$. In case of a TG, the multiplicity $m = 1, \forall \left(v_i, c_j\right) \in \mathcal{E}_P$. The set $\mathcal{V}_P$ can be separated into sets for punctured vertices $\mathcal{V}_P^p$ and unpunctured vertices $\mathcal{V}_P^u$, where $\mathcal{V}_P = \mathcal{V}_P^p \cup \mathcal{V}_P^u$ and $\mathcal{V}_P^p \cap \mathcal{V}_P^u = \varnothing$. The size of these sets are given by $\left|\mathcal{V}_P^u\right| = N_P^u$ and $\left|\mathcal{V}_P^p\right| = N_P^p$. The reader should accept for now that there is such a distinction, but section §2.5.2 explains the reasons and the consequences of such a distinction in greater detail.
An example of a PG is illustrated in Fig. 2.7a, where vertices with a $+$ represent $\mathcal{C}_P$, called check nodes, and all other vertices represent $\mathcal{V}_P$, called variable nodes. In order to distinguish the variable nodes further, all unpunctured variable nodes are filled and all punctured variable nodes are not. The multiplicity of an edge is shown by parallel edges between the same vertices.

Example 2.13. The two representation types of the ARJA protograph, introduced by Divsalar, et al. [DJDT05], are shown in Fig. 2.7.

The equivalent representation in matrix form is shown in Fig. 2.7b, which is called the base matrix $\mathbf{B}$ of the protograph. It is the biadjacency matrix of the graph, where each column corresponds to a variable node and each row corresponds to a check node. The elements of the base matrix are 0, if there is no



(a) graph representation

$$\mathbf{B} = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 0 & 3 & 1 & 1 & 1 \\ 0 & 1 & 2 & 1 & 2 \end{pmatrix}$$

(b) matrix representation

Figure 2.7: Protograph - graphical and matrix representation

connection between the nodes or represent the multiplicity of the edge.
Both representations are used throughout the thesis depending on which representation is more suitable to explain certain relationships.
As stated before, PGs can be regarded as a kind of LDPC code ensemble description, where the resulting code has the same structure as the PG itself. This statement translates into the direct mapping of a node of the protograph to a group of nodes and of an edge to a group of edges in the TG of the LDPC codes derived. In light of this relationship, the elements of the PG (variable nodes, check nodes and edges) can be regarded as variable node classes, check node classes and edge classes, respectively.
The procedure to derive LDPC codes from a code ensemble is called *lifting* and the general idea is explained in §2.2.3. For PGs, there are more restrictions, e. g., the number of nodes (multiple of the PG nodes) and how to connect these nodes (the structure of the PG must be preserved).

In the following some properties of the PG are defined.

Definition 2.14. The puncturing ratio $\mathcal{P}$ describes the ratio of punctured variable nodes to total number of variable nodes, and is written as

$$\mathcal{P} = \frac{N_P^p}{N_P} = 1 - \frac{N_P^u}{N_P} = \frac{N_P^p}{N_P^p + N_P^u} \tag{2.30}$$

Definition 2.15. The protograph (code ensemble) rate $\mathcal{R}_P$ is given by:

$$\mathcal{R}_P = \left(1 - \frac{M_P}{N_P}\right) \cdot \left(1 - \frac{1}{\mathcal{P}}\right) = \frac{N_P - M_P}{N_P} \cdot \frac{N_P}{N_P^u} = \frac{N_P - M_P}{N_P^u} \tag{2.31}$$

2.4.2 Convolutional Protograph

A significant portion of this work is focused on the use of PGs for Low-Density Parity-Check Convolutional codes. In general, there are several ways to define these codes. However, they are very similar and can be transformed into one another. This thesis utilize the Convolutional Protograph **(CPG)** in order to show the convolutional structure of the codes derived.

Starting point for the considerations is a PG, e. g., Fig. 2.7. Its base matrix can be split into $m_s + 1$ matrices of the same dimensions, having the property

$$\mathbf{B} = \sum_{i=0}^{m_s} \mathbf{B}_i \quad , \tag{2.32}$$

where each matrix $\mathbf{B}_i$ is called a *partial base matrix* and m_s is called *syndrome former memory*. A convolutional structure of the code can be achieved by means of a band structured matrix given by

$$\mathbf{B}_{[-\infty,\infty]} = \begin{bmatrix} \ddots & & \ddots & & & & \\ & \mathbf{B}_{m_s} & \dots & \mathbf{B}_0 & & & \\ & & \ddots & & \ddots & & \\ & & & \mathbf{B}_{m_s} & \dots & \mathbf{B}_0 & \\ & & & & \ddots & & \ddots \end{bmatrix} \quad . \tag{2.33}$$

Although, this structure can be called a pure convolutional code, a limitation of the length is necessary for several reasons:

Fixed Beginning It is unavoidable to define a starting point for the transmission . Without loss of generality, the time instance of the beginning can be set to $t = 0$.

Termination Although a transmission can be very long, it will eventually end and various kinds of endings are feasible.

Modified Structure The way the code starts and ends modifies the structure in a certain way and can improve the code performance. A detailed analysis of this change is the main focus of this thesis and is carried out in the following chapters.

The final terminated convolutional base matrix is given as

$$\mathbf{B}_{[0,L-1]} = \begin{bmatrix} \mathbf{B}_0 & & \\ \vdots & \ddots & \\ \mathbf{B}_{m_s} & \dots & \mathbf{B}_0 \\ & \ddots & \vdots \\ & & \mathbf{B}_{m_s} \end{bmatrix}_{(L+m_s)M_P \times LN_P} . \tag{2.34}$$

Example 2.16. A Terminated Convolutional Protograph for $m_s = 2$ and $L \to \infty$ based on the PG in Example 2.13 is shown

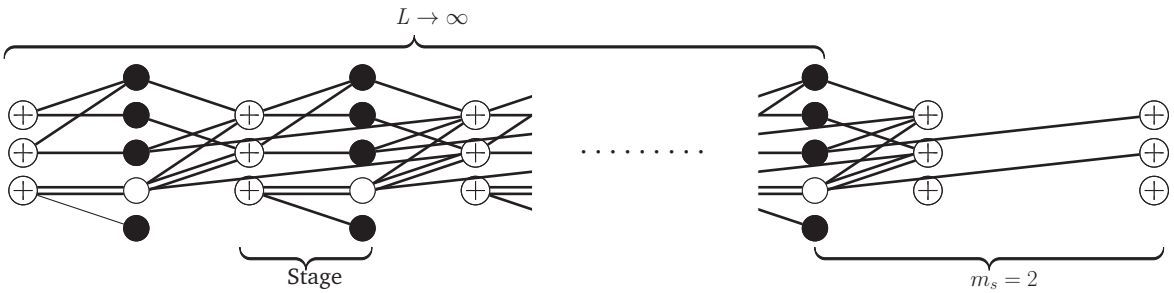


Figure 2.8: Unwrapped convolutional protograph with $m_s = 2$ and without termination ($L \to \infty$)

Having defined and analyzed the Terminated Convolutional Protograph **(TCPG)**, ordinary *lifting* methods like ACE and PEG can be used derive the final codes.

2.5 Analysis Methods

This work aims to find design rules or recommendations of Protograph-based Low-Density Parity-Check Convolutional **(PG-LDPCC)** codes for the specific application scenario of turbo equalization. In this regard, certain parts can be analyzed alone, whose results can be used to study the interplay between several components in order to eventually determine the performance of the overall system. This section deals with the methods used throughout the thesis to analyze and evaluate the codes proposed. The first technique is a very powerful tool to analyze the performance of any iterative system, and is called Extrinsic Information Transfer **(EXIT)** chart. Then, the general methodology, requirements for the input and output parameters as well as limitations of this tool are worked out. The second part of this section focuses on the analysis of code ensembles defined by certain parameters by means of a method called Density Evolution **(DE)**. In a nutshell, this method tracks the probability density functions throughout the decoding process in order to predict the performance of the code ensemble under certain conditions. DE is explained with the aid of *unstructured* codes, since the basic principle is the same as that of *structured* codes. As is the case for PG-based codes, it is possible to find closed form equations for certain channels assumed, and it was first defined for these kinds of code ensembles. Additionally, the characteristic features needed for the extension of DE for PGs are shown.

2.5.1 System Analysis - EXIT Charts

EXIT charts were introduced for the analysis of iteratively decoded parallel concatenated convolutional codes, commonly known as Turbo Codes, in [tB01]. Later they were extended to various kinds of iterative systems, all having the *turbo principle* (cf. [Hag02]) in common. Nowadays EXIT charts are widely used as a standard tool. Hence, the basic idea, requirements, limitations and its relation to the measures in system model assumed (§2.1) are briefly introduced.
Up to this point, all explanations regarding information transfer are done by means of the system model. However, analysis is partially carried out based on information theory as well. Therefore, the relationship between the measures used before and the information theoretic measure must be illustrated, where the relationship is justified based on [Hag02] (due to the tutorial nature of that paper). In order

to avoid repeating basic knowledge of information theory, the author assumes that the reader is familiar with the definition of *Entropy*, *Conditional Entropy* and *Mutual Information* ***(MI)*** (otherwise, be referred, e. g., to [Mac03]). Then, the mutual information between the code word encoded and LLR values with respect to the code word can be defined as follows.

Definition 2.17 (Mutual Information between code word and LLR-values)**.** The MI $I(L;X)$ between antipodal encoded equally likely binary inputs, i. e., $x \in \{+1,-1\}$, $\Pr(x=-1)=\Pr(x=+1)=\frac{1}{2}$, with consistent and symmetrically distributed LLR values is defined as

$$I(L;X)=1-\mathrm{E}\left[\log_2\left(1+\mathrm{e}^{-L}\right)\right]=1-\frac{1}{N}\sum_{n=0}^{N-1}\log_2\left(1+\mathrm{e}^{-x_n\cdot L_n}\right) \quad , \tag{2.35}$$

where the ergodic theorem, i. e., replacing the expectation by the time average, is applied and N is the number of samples (usually considered to be quite large).

With this relation in mind, it is possible to shift consideration to an information theoretic level, which is used for explaining the details of EXIT charts.
The basic idea of an EXIT chart is to illustrate the exchange and improvement of reliability information in a system over several iterations in order to predict the most likely behavior of such a system.

Definition 2.18 (SISO module)**.** A SISO module yields reliability information with respect to a measure X based on the reliability information with respect to the same measure and additional side information. The reliability measure before the processing through the module is called a priori information I_a, the reliability measure after the processing is called a posteriori information I_p and the information gain is called extrinsic information I_e, i. e.

$$I_p=I_a+I_e \quad . \tag{2.36}$$

It is common to make use of LLR values in order to represent the reliability information.

Since there is a direct relation between the MI and the LLRs under the conditions mentioned in Definition 2.17, the respective LLR values before and after the SISO module are called a-priori LLR L_a and - posteriori LLR L_p. The gain in information

can be expressed by means of LLR values as well and is called the extrinsic LLR L_e, such that

$$L_p = L_a + L_e \quad . \tag{2.37}$$

It is also possible to find a mapping $f(I_a) = I_e$ and as a result, $f(I_a) + I_a = I_p$. The extrinsic transfer function f can be illustrated in a coordinate system, with abscissa I_a and ordinate I_e.

In any turbo-like system, there is an iterative exchange of these information measures by means of LLR values, where the output information of one is used as the input of the other and vice versa. In particular, there is an exchange of extrinsic information between the modules. In the case considered, there are the two modules, namely, the soft equalizer and the decoder, which are both SISO modules. A superscript D is used for the decoder and the superscript E is used for the soft equalizer in all measures, e. g., I_e^D is the extrinsic mutual information of the decoder and L_p^E is the a-posteriori LLR of the soft equalizer.
The extrinsic functions of the equalizer and the decoder can be shown in the same EXIT chart, where the axes for the latter are swapped because the output of one is the input of the other.

Example 2.19. The extrinsic transfer curves for the two channels considered can be found in Fig. 2.9 and Fig. 2.10. The curves for the equalizer for the different channels are calculated by using the the previously discussed approach by averaging over a sufficiently large number of repetitions and applying the *law of large numbers*. While the UWB-NLOS channel considered has a relatively flat shape, the curve of the exponentially decaying 10 tap channel has a very steep shape due to a higher ISI. Therefore, additional information from the decoder in the UWB case can increase the output of the equalizer only slightly. In comparison to the first channel, the latter can make use of this information more efficiently. When both figures are compared to each other and the SNR values are taken into account, it can be seen that although I_e is in the same range for $I_a = 0$, the SNR values are much lower for the UWB-NLOS channel. However, I_e would be same for both channels for $I_a = 1$, because the equalizer could eliminate every ISI and only the additive white Gaussian noise is left.

Remark 2.20. The first module activated never has any information from the other module and all of its extrinsic information generated is based on the side informa-

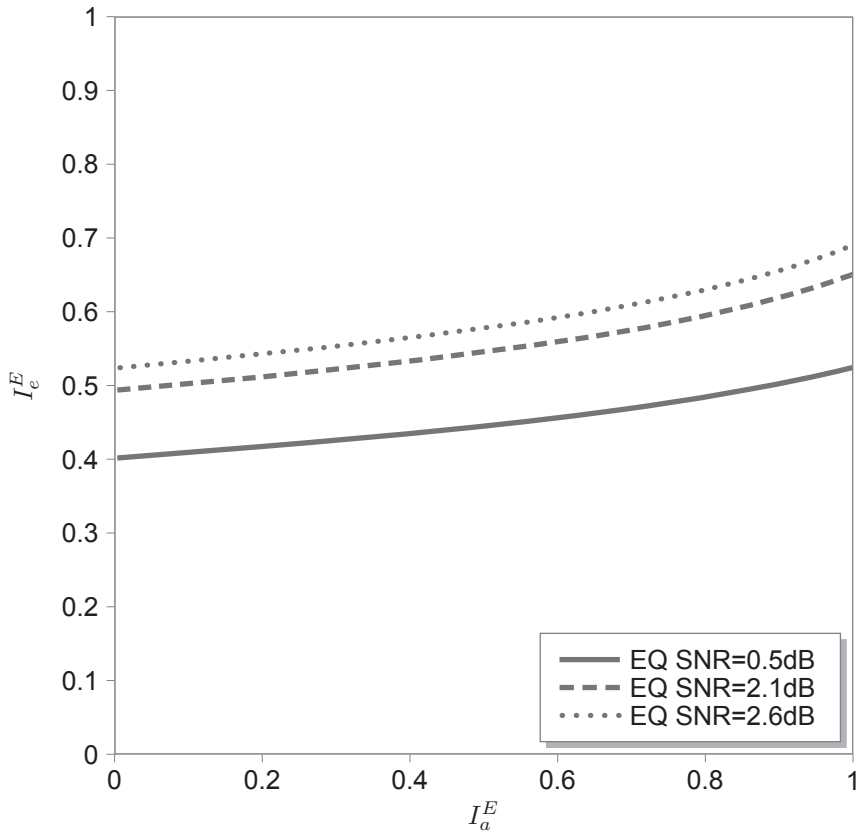


Figure 2.9: Extrinsic Transfer Curves for Soft-Equalizer for UWB NLOS channel - low ISI

tion, e. g., mapping rules, channel estimator, etc..

The exchange of information throughout several iterations can be represented by trajectory between the curves and the reliability of the information is the higher the closer the trajectory comes to point $(1, 1)$ without intersection (Not shown here, but in cf. [Hag02])
In general, the idea of the EXIT chart can be extended even to more than two dimensions, if the number of modules involved in an iterative processing increase. However, this is not done very often due to a lack of proper visualization for higher dimensions.

2.5.2 Code Analysis - Density Evolution

As previously discussed, it is possible to define an extrinsic transfer curve for any module able to exploit soft information to produce improved soft information by

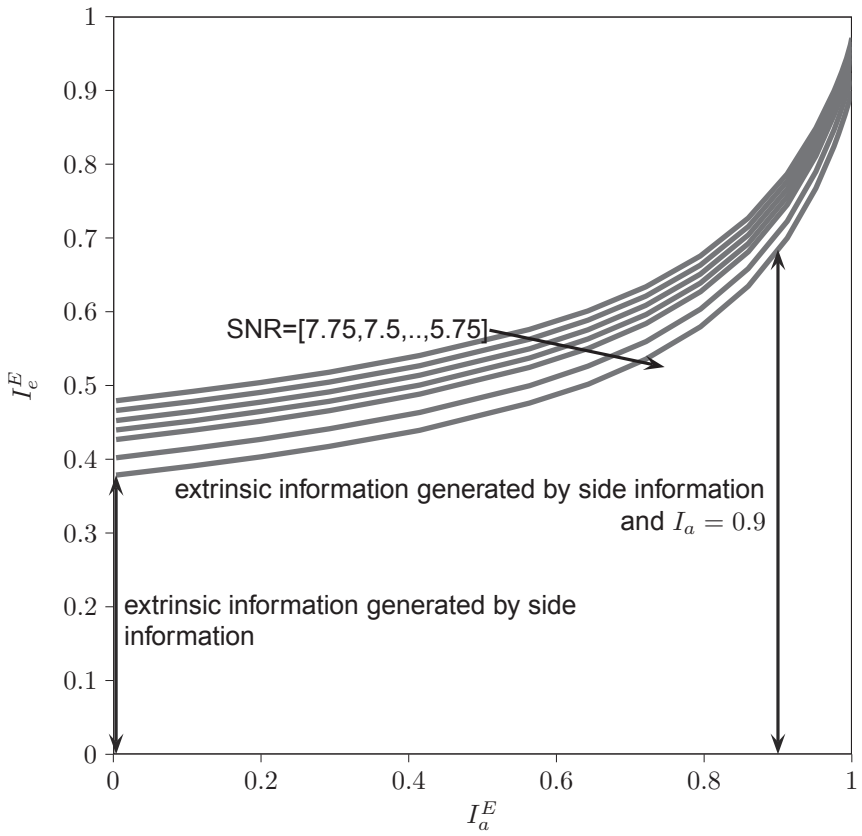


Figure 2.10: Extrinsic Transfer Curves for Soft-Equalizer for 10 tap channel - high ISI

measuring this information before and after the module. However, such an approach still needs a significant number of simulation runs, but once such an extrinsic transfer curve is obtained, it can be used for performance predictions in larger systems.

However, in case of LDPC codes, another approach can also be used to determine an extrinsic transfer curve of this code by utilization the code ensemble descriptions. This method is called Density Evolution **(DE)** (the idea was introduced in [RU01]), where the basic idea is to track the Probability Density Function **(PDF)** through the decoding process for a given input distribution and any given code ensemble[6]. In case of the Binary Erasure Channel **(BEC)** and related channels, there

[6]In theory, it is possible to track this for a given code, e. g., for a given parity-check matrix. However, such an analysis is performed on a code ensemble than a particular code, since tracking of large number of PDFs is computationally intensive and, more importantly, it is still averaging over all the codes having this structure.

are known closed form solutions.
In the next section, the DE for an unstructured code ensemble by means of degree distributions, under the assumption that the input approximately follows a Gaussian distribution, is shown. This is sufficient for the understanding of the DE algorithm and highlights some specific challenges that arise. Following which, a modification needed for the application of the algorithm to structured ensemble description, and particularly to PGs, are highlighted and explained.

Quantized Density Evolution

As described, e. g., in [Chu00] and [Ric07] DE predicts the decoding behavior. Hence, it is necessary to take a closer look at the operations executed at the nodes involved. On the variable node side, there is an addition of LLR values. Assuming the messages are not LLR values themselves, but rather the PDFs of LLR distributions, the operation executed in this case is a convolution of the PDFs. On the check node side there is no such a function available, i. e., a different approach, which is a *quantized version* of density evolution, must be used. $Q\left(L_w\right)$ is the quantified version of an LLR L_w given by

$$Q(L) = \begin{cases} \left\lfloor \frac{L_w}{\Delta} + \frac{1}{2} \right\rfloor \Delta & L_w \geq \frac{\Delta}{2} \\ \left\lceil \frac{L_w}{\Delta} + \frac{1}{2} \right\rceil \Delta & L_w \leq -\frac{\Delta}{2} \\ 0 & \text{otherwise,} \end{cases} \tag{2.38}$$

where Δ is the quantization interval. A clipping of the quantified LLRs into the range $[-B, +B]$ to make computation numerically feasible is undertaken and the clipped, quantized, LLR $\hat{Q}\left(L_w\right)$ is defined as

$$\hat{Q}\left(L_w\right) = \begin{cases} \text{sgn}\left(Q((L_w))\right) \cdot B & \left|Q\left(L_w\right)\right| \geq B \\ Q\left(L_w\right) & \text{otherwise.} \end{cases} \tag{2.39}$$

The number of quantization steps q can be calculated by $q = \frac{2B}{\Delta}$, but it is recommended that the quantization interval be determined by fixing the values of$q$ and B.

Let $p_w = (p_w[-B/\Delta], ..., p_w[B/\Delta])$ be the Probability Mass Function **(PMF)** of a

quantified and limited LLR $\hat{Q}\left(L_w\right)$, where the elements are defined as

$$p_w[k] = P(\hat{Q}\left(L_w\right) = k\Delta) \text{ for } k \in \left\{-\frac{B}{\Delta}, ..., \frac{B}{\Delta}\right\} \tag{2.40}$$

A non-linear operation on check node side must evaluated, and hence, a two-input operator $T(a,b)$ that maps two quantified and clipped LLR inputs onto another quantified and clipped LLR, is defined as

$$T(a,b) = \hat{Q}\left(2\,\mathrm{atanh}\left(\tanh\left(\frac{\hat{Q}(a)}{2}\right)\tanh\left(\frac{\hat{Q}(b)}{2}\right)\right)\right) \quad . \tag{2.41}$$

The two-input operator reflects the decoding strategy of belief propagation. Nevertheless, it can be easily adapted to other decoding strategies, e. g., the operator $T_s(a,b)$ for the *sig-min approximation* is given by

$$T_s(a,b) = \hat{Q}(\text{sign}\left\{a \cdot b\right\} \cdot \min\left\{|a|, |b|\right\}. \tag{2.42}$$

After having defined the operations on LLRs, another two-input operator $\Gamma(p_1, p_2)$ that operates on PMFs, has to be defined. The elements $\Gamma[k]$ can be calculated by

$$\Gamma(p_1, p_2): \ \Gamma\left[k\right] = \sum_{(i,j):k\Delta = T(i\Delta, j\Delta)} p_1[i]p_2[j] \quad , \tag{2.43}$$

which adds up all probabilities having a specific quantified and limited LLR value. In order to simplify the notation, let the n-th power of a one-input operator $\Gamma(p_1)$ be the recursive application of $\Gamma(.,.)$ $(n-1)$-times , i. e.,

$$\Gamma^n(p_1) = \Gamma(p_1, \Gamma(p_1, \Gamma(.., p_1))) \quad . \tag{2.44}$$

For the sake of completeness, the convolution of twoPMFs is denoted as

$$p_1 * p_2: \ \left(p_1 * p_2\right)[k] = \sum_{-B}^{B} p_1[i]p_2[j-i] \quad , \tag{2.45}$$

and the $(n-1)$-times recursive application of the convolution is denoted as

$$\otimes^n\left(p_1\right) = p_1 * (p_1 * (.. * p_1))) \quad . \tag{2.46}$$

However, since this the convolution is a computationally intensive method and the PMFs are reused a couple of the times, it is reasonable to make use of a corresponding base system. Hence, the PMFs are transformed in the Fourier domain, where a multiplication is the corresponding operation for a convolution in the real domain. After processing of the convolutions, the PMFs are required to be re-transformed to the real domain for further processing. The Fourier transformation is denoted by $\mathscr{F}\{p\}$.

Density Evolution for Unstructered Codes

For the input of the DE, a certain PMF must be assumed since it mimics the decoding process. Without loss of generality, the all-zero code word is assumed and a PMF according to a transmission over an Additive White Gaussian Noise **(AWGN)** channel is generated. There the variance σ^2 can be used to adjust the a-priori information that is reflected by the PMF. Once such an input PMF p_a^D is defined, the following algorithm Alg. 2 is applied using the equations from Eq. 2.24 - Eq. 2.27.

Alg. 2 Density Evolution for Degree Distributions

Require: p_a^D

1: ${}_0p_e^V = p_a^D$
2: **while** $s < s_{max}$ (and not decoded yet) **do**
3: $\quad {}_sp_a^C = {}_{s-1}p_e^V$
4: $\quad {}_sp_e^C = \sum_{l=d_{c,\min}}^{d_{c,\max}} \rho_l \Gamma^{l-1}\left({}_sp_a^C\right)$
5: $\quad {}_sp_a^V = {}_sp_e^C$
6: $\quad {}_sp_e^V = \sum_{l=d_{v,\min}}^{d_{v,\max}} \lambda_l \left(p_a^D * \left(\otimes^{l-1} {}_sp_a^V\right)\right)$
7: $p_e^D = \sum_{l=d_{v,\min}}^{d_{v,\max}} \alpha_l \left(\otimes^l {}_sp_a^V\right)$
8: $p_p^D = p_a^D(v_i) * p_e^D(v_i)$

Density Evolution for Protograph-Based Codes

In general, an analysis of PGs with the help of a DE is feasible, with the aforementioned degree distributions, since these distributions can be derived from the PG as well. However in this case, not only is a loss of information about the structure

of the parity-check equations involved, but there is also a limitation of avoiding degree-1 nodes. Considering these two facts, it seems natural to extend the DE for the application to the PGs, where these drawbacks are eliminated.
Taking a closer look at the algorithm, it can be seen that, after each iteration of the variable and the check nodes, the PMFs are averaged (given the degree distributions from an edge perspective) in order to be used as inputs for the other components. Hence, the algorithm assumes that there is only one connection from the input of the decoder to the variable node decoder, and exactly one edge between the variable node decoder and the check node decoder, i. e., the algorithm deals with maximum two PMFs at a time in the variable node decoder. Nothing else is possible due to lack of further structural information.
In the case of a PG, each node can be seen as a sub-decoder of the particular component having distinct inputs and outputs to various other sub-decoders of the other component, i. e., the algorithm is applied directly to the graph and averaging is no longer necessary. Hence, the algorithm can be simplified in some areas and extended in others. All such modifications applicable to PGs will not be shown here in order to accurately reflect the relations in the graph, since it would padding out the number of variables employed in this thesis. This additional variables would make the thesis unnecessarily complex without contributing to much to actual topic of this thesis. Therefore, only the changes mandatory are pointed out and the reader is referred to [Ric07], which has a very good description of the algorithm.

1. Each node can have a distinct input PMF, denoted by $p_a^D\left(v_i\right), \forall v_i \in \mathcal{V}_P$

2. All a-priori and extrinsic PMFs must be considered with respect to the edge they are transmitted over, i. e., $p_a^C\left(\left(v_i, c_j\right)\right)$, $p_a^V\left(\left(v_i, c_j\right)\right)$, $p_e^C\left(\left(v_i, c_j\right)\right)$, and $p_e^V\left(\left(v_i, c_j\right)\right)$ must be distinguished.

3. Every extrinsic PMF is calculated based on distinct a-priori PMFs and the input PMF,
   i. e., $p_e^C\left(\left(v_i, c_j\right)\right) = \mathrm{f}\left(\left\{p_a^C\left(\left(v_{i'}, c_j\right)\right)\right\}\right)$ and
   $p_e^V\left(\left(v_i, c_j\right)\right) = \mathrm{g}\left(\left\{p_a^V\left(\left(v_i, c_{j'}\right)\right)\right\}, p_a^D\left(v_i\right)\right)$

4. Unary functions ($\Gamma^l(\cdot)$ and $\otimes^l(\cdot)$) are needed only for $\left|\left(v_i, c_j\right)\right| > 1$, i. e., $p_e^C\left(\left(v_i, c_j\right)\right) = \mathrm{f}\left(\left\{p_a^C\left(\left(v_{i'}, c_j\right)\right)\right\}, \Gamma^{l-1}\left(\left(v_i, c_j\right)\right)\right)$, where $i \neq i'$ and $p_e^V\left(\left(v_i, c_j\right)\right) = \mathrm{g}\left(\left\{p_a^V\left(\left(v_i, c_{j'}\right)\right)\right\}, \otimes^{l-1}\left(\left(v_i, c_j\right)\right), p_a^D\left(v_i\right)\right)$, where $j \neq j'$.

5. Each node can have a distinct output PMF, denoted by $p_e^D\left(v_i\right), \forall v_i \in \mathcal{V}_P$.

2.6 Main Points of this Chapter

This section summarizes the most important assumptions, boundary conditions and tools applied.

System Model

- The modulator uses a linear mapping in order to benefit most from the iterative receiver structure.
- A cyclic prefix is added to the symbols in order to enable an equalization in frequency domain.
- Two channels with almost opposite characteristics w. r. t. ISI are considered. The low ISI channel follows a real life UWB-NLOS PDP. The high ISI channel follows a 10 tap exponentially decaying PDP.
- Block fading is assumed.
- The soft equalization is done in frequency domain and the output of the soft equalizer is approximately Gaussian distributed.

LDPC codes

- The codes studied in this thesis are based on Protograph.
- All code words are transmitted as blocks. In the case of the convolutional variants the CPG are terminated to meet this requirement.
- The lifting is done by means of the PEG and the ACE algorithm.

Analysis

- Extrinsic transfer curves of codes are determined by means of quantized density evolution.
- Extrinsic transfer curves of the soft equalizer are determined by means of simulations.
- The interplay of soft equalizer and decoder are analyzed by means of EXIT charts.