The Information Bottleneck Method in Communications

Vom Promotionsausschuss der
Technischen Universität Hamburg
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von
Jan Christopher Lewandowsky

aus
Wuppertal, Deutschland

2020

Vorsitzender des Prüfungsausschusses:
Prof. Dr. Herbert Werner

1. Gutachter:
Prof. Dr.-Ing. Gerhard Bauch

2. Gutachter:
Prof. Dr.-Ing. Stephan ten Brink

Tag der mündlichen Prüfung:
21.07.2020

Abstract

The Information Bottleneck method is a generic information theoretical framework which aims for the compression of an observed random variable to a compressed random variable. The focal aim in designing this compression is to preserve *relevant information*. The method originates from machine learning and so far only has a few practical applications in communications.

This thesis describes the application of the Information Bottleneck method to problems of receiver-sided signal processing in communications. The complexity of the receiver-sided baseband processing algorithms for demodulation and channel decoding causes a severe bottleneck in modern digital communication receivers. Their implementation complexity is mainly influenced by the bit width used to represent the signals processed in the hardware of the receiver and the arithmetical operations involved in the signal processing algorithms. Practical receiver implementations, therefore, have to be strongly quantized. Moreover, the involved signal processing algorithms have to be as simple as possible.

The Information Bottleneck method provides algorithms which aim to maximize the preserved relevant information for a given bit width, hence motivating to apply the Information Bottleneck method to receiver design.

The problems covered in this thesis are the design of scalar channel output quantizers, the decoding of low-density parity-check codes and channel estimation and detection algorithms. It is shown that the Information Bottleneck design principle allows to build signal processing blocks for these problems which allow for very small bit widths, typically around four to five bits per sample. Anyway, performance close to that of signal processing algorithms with double precision can be achieved. A key to achieve this small bit width is the signal representation using only quantization indices instead of real or complex representation values. Moreover, all operations required in the obtained signal processing blocks degenerate to simple lookup operations. As a result, the aforementioned design goals for communication receivers are inherently achieved by the receiver design with the Information Bottleneck method.

Keywords: Information Bottleneck method, low-density parity-check codes, quantization, mutual information, lookup tables

Zusammenfassung

Die Information Bottleneck[1] Methode ist ein informationstheoretisches Verfahren, welches auf die Kompression einer beobachteten Zufallsvariable abzielt. Das wichtigste Ziel dieser Kompression ist die Erhaltung sogenannter *relevanter Information*. Die Methode hat ihren Ursprung im Maschinenlernen und bisher existieren nur wenige Anwendungen der Methode in der Kommunikationstechnik.

Die vorliegende Arbeit beschreibt die Anwendung der Information Bottleneck Methode für Problemstellungen der Signalverarbeitung in Kommunikationsempfängern. Die Komplexität der Basisbandsignalverarbeitung für die Demodulation und die Kanaldecodierung verursacht einen Flaschenhals in modernen Empfängern. Diese Komplexität wird wesentlich durch die Bitbreite der verarbeiteten Signale in der Hardware sowie die arithmetischen Operationen der Signalverarbeitungsalgorithmen bestimmt. Praktische Implementierungen müssen deshalb stark quantisiert arbeiten und die angewendeten Algorithmen so aufwandsarm wie möglich gehalten werden.

Die Information Bottleneck Methode bietet Algorithmen, die die erhaltene relevante Information für eine vorgegebene Quantisierungsbitbreite maximieren. Dies motiviert die Anwendung dieser Methode für die Signalverarbeitung. Die konkret mit der Methode in dieser Arbeit behandelten Probleme sind das Design von Quantisierern für das Empfangssignal, die Decodierung von low-density parity-check codes[2] sowie die Kanalschätzung und die Detektion. Es wird gezeigt, dass die Information Bottleneck Methode ermöglicht, Signalverarbeitungsblöcke für diese Aufgaben zu konstruieren, die mit sehr geringen Bitbreiten von typisch nur vier bis fünf Bits je Abtastwert auskommen. Dennoch kann die Performanz von Signalverarbeitungsalgorithmen mit doppelter Gleitkommagenauigkeit nahezu erreicht werden. Der wesentlichste Aspekt dabei ist die Repräsentation der verarbeiteten Signale nur durch Indizes von Quantisiererfächern anstelle von reellen oder komplexen Zahlen. Überdies sind alle Operationen, die für die Signalverarbeitung erforderlich sind, ausschließlich Leseoperationen in statischen Speichern (sogenannte Lookup-Operationen[3]). Die

[1] Information Bottleneck: (englisch) Informations-Flaschenhals

[2] Low-density parity-check codes: (englisch) Kanalcodes mit spärlichen Paritätsmatrizen

[3] Lookup: (englisch) Nachschlagen

Information Bottleneck Methode hilft so, die oben genannten Ziele im Design von Kommunikationsempfängern zu erreichen.

Schlagwörter: Information Bottleneck Methode, low-density parity-check codes, Quantisierung, Transinformation, Lookup-Tabellen

Acknowledgements

I want to express my great gratitude to my supervisor Professor Gerhard Bauch, whom I have met first during my studies at the Universität der Bundeswehr Munich. Your enthusiasm for signal processing and communications which I could feel in your lectures has had a huge impact on my own interest in the field. When I asked you for the possibility to write my Ph.D. thesis under your supervision, you believed in me from the very first moment and encouraged me to start. I thank you for your persistence and your demands for scientific clarity and accuracy that you have taught me over the last years.

I also want to thank the entire team from the Institute of Communications of the Hamburg University of Technology for the many hours that we have spent discussing on our field. Special thanks are meant to my good friend Maximilian Stark for our hours and days of discussion, programming and reviewing each other's results.

Moreover, I want to express my special gratitude to Professor Konrad Pilzweger, my former maths professor from the Universität der Bundeswehr Munich. Our discussions on the maths behind the Information Bottleneck method were more than valuable for this work.

Besonderer Dank gebührt überdies meinen Eltern, meinen Geschwistern sowie meiner Partnerin Nina. Ohne Eure Unterstützung und Euer Verständnis, wäre es mir mitnichten möglich gewesen, diese Arbeit anzufertigen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

The permanent availability of high-speed communication services is taken for granted by almost everybody today. People simply are connected all over the world by their smartphones and their computers without even thinking about it. Sending an instant message from a driving bus in Hamburg to a smartphone lying on the table of a restaurant in New York is a normal routine, despite the technical process behind this action is tremendously complex. Social networks, video streaming platforms, smartphone apps and instant messaging services are just a few examples that indicate the role of reliable high-speed communication links for our everyday lives. Emerging trends like the Internet of things, smart devices, machine-to-machine communications in production processes and autonomous vehicles will push our need for fast and reliable communication systems further.

Providing reliable high-speed communication services to the users requires sophisticated techniques to be applied at the transmitter and the receiver, for example, to cope with transmission errors that occur on the physical transmission channel. In modern digital communication systems, the receiver-sided baseband processing algorithms for demodulation and channel decoding are often complex and result in hardware bottlenecks concerning the occupied chip area, the routing complexity, the power consumption and the latency [1].

The implementation complexity of a digital receiver is strongly influenced by the bit width used for the analog-to-digital conversion and the complexity of the arithmetical operations applied in the signal processing algorithms. Strongly quantized implementations with low bit widths and simple arithmetical operations are needed in practice to meet the implementation requirements. State-of-the-art practices to build such receivers are using fixed point precision for the signal processing and mathematical approximations to replace too complex operations with simpler ones [1, 2]. Lowering the implementation complexity like this, however, often results in severe performance degradations in comparison to high precision implementations of the optimum signal processing algorithms. A well known example for an algorithm which is widely used in practice, but has suboptimum performance is the min-sum decoding algorithm for error correcting low-density parity-check (LDPC) codes [2].

One can conclude that simple and quantized receiver algorithms that offer performance competitive with the optimum or close-to-optimum high precision

algorithms are of great practical interest. This thesis aims to develop such algorithms using an unconventional approach which involves a framework from machine learning. This framework is called the *Information Bottleneck method* [3]. Naftali Tishby *et al.* introduced the Information Bottleneck method in [3] without any specific application. It solves the problem of compressing an observed random variable to a compact compressed random variable while aiming to maintain *relevant information* shared between a relevant random variable of interest and the obtained compressed variable.

The Information Bottleneck method has been used comprehensively in machine learning so far. Applications reaching from image processing [4, 5] over the analysis of interacting neurons in neuroscience [6], language processing [7] and numerous other problems [8–12] are described in the literature. In contrast, so far, the method only has a few practical applications in communications.

Some preliminary works investigate the applicability of the method for the design of relevant information preserving channel output quantizers [13, 14]. Moreover, several ideas for quantizing and forwarding relevant information at a communication relay were investigated in [15–21], among which [15–18] also focussed on network coding. Works with a strong focus on the algorithmical processing of Information Bottleneck algorithms in the communications context are [22–27].

The essential task of any communication receiver is to provide *relevant information* on the transmitted data to its user. This motivates the application of the Information Bottleneck method also for the design of receiver algorithms with low complexity which is the main subject of this thesis. As we will see, designing receiver components with the Information Bottleneck method automatically minimizes the required number of bits needed to represent the processed signals in the hardware. Moreover, the only required operations in the resulting systems are simple lookup operations in static tables. Therefore, the proposed receiver design with the Information Bottleneck method inherently achieves the mentioned design aims for communication receivers.

The most important contributions of this thesis are:

- A novel Information Bottleneck algorithm for the quantization of discrete memoryless channels is introduced.

- A very simple graphical model called *Information Bottleneck graphs* is developed. Information Bottleneck graphs are extended factor graphs which allow to model mutual information relations between a huge number of variables that result from application of the Information Bottleneck method.

- A framework for the construction of completely integer based LDPC decoders for regular LDPC codes with the Information Bottleneck method is developed and investigated.

- A technique called *message alignment* is introduced which enables to also build Information Bottleneck based decoders for arbitrary irregular LDPC codes and higher order modulation schemes.

- The Information Bottleneck design principle is extended to channel estimation and coherent detection, such that finally entire signal processing chains are constructed which perform all signal processing only based on integers and lookup operations.

- Exemplary implementations of Information Bottleneck based LDPC decoders on a signal processing platform are presented to evaluate and quantify practical benefits of the proposed receiver design approach.

Related works by other authors that also deal with similar techniques in the design of quantized LDPC decoders are [28–35]. Especially [28] has had a huge impact on this thesis, as some of its key ideas were adapted in the design of LDPC decoders with the Information Bottleneck method described later.

The remainder of the thesis is structured as follows. This introduction ends with an overview of the notation used. The next chapter introduces the required backgrounds on Bayesian statistics, information theory, LDPC codes and factor graphs. Afterwards, Chapter 3 introduces the Information Bottleneck method and describes some Information Bottleneck algorithms from the literature.

Chapter 4 introduces Information Bottleneck graphs which will be used to model Information Bottleneck signal processing chains in the remainder of the thesis.

Chapter 5 starts with a section on Information Bottleneck based quantizer design for various memoryless communication channels under several input modulation formats. The focal idea is to construct quantizers which preserve the maximum possible amount of relevant information on the useful part of the received signal for a given bit width. The remainder of Chapter 5 is dedicated to the application of the Information Bottleneck method to receiver-sided signal processing problems.

The problem of decoding a regular LDPC code only based on quantized indices with the Information Bottleneck method is considered first. Afterwards, the approach is generalized to higher order modulation formats and irregular LDPC codes by the introduction of message alignment.

Following the investigation of LDPC decoding, the Information Bottleneck method is also applied to channel estimation and coherent detection. In the end, two receiver chains which include channel estimation, detection and LDPC decoding are constructed which only process quantization indices and perform all signal processing using only lookup tables. Monte Carlo bit error rate simulations are used extensively in Chapter 5 to evaluate the performance of the resulting receiver algorithms and to compare it to state-of-the-art techniques.

Chapter 6 presents Information Bottleneck based and conventional LDPC decoder implementations on a digital signal processor (DSP) and compares them in order to evaluate and quantify practical gains of the proposed Information Bottleneck approach to receiver design. Since these results focus on DSP implementations, they are particularly interesting in the context of software defined radio (SDR) applications.

Chapter 7 summarizes the lessons learned from the application of the Information Bottleneck method to the mentioned problems in receiver design and states some remaining challenges. It also tries to lift the insights gained from the preceding chapters towards a general understanding of Information Bottleneck based receiver design, before Chapter 8 concludes the thesis.

Notation

An overview of the notation and the symbols used frequently in the thesis is provided in the following table. This table is restricted to fundamental

mathematical expressions and functions. More specific notations are explained in context, right when they appear.

$\arg\max_x f(x)$	argument maximizing the function $f(x)$
$\arg\min_x f(x)$	argument minimizing the function $f(x)$
$\mathbf{A}$, $\mathbf{A}^{\mathrm{T}}$, $\mathbf{A}^{-1}$	matrix $\mathbf{A}$, transposed matrix $\mathbf{A}^{\mathrm{T}}$, inverse matrix $\mathbf{A}^{-1}$
$\mathbf{b}$, $\mathbf{b}^{\mathrm{T}}$	column vector $\mathbf{b}$, transposed vector $\mathbf{b}^{\mathrm{T}}$
$\cosh(.)$	hyperbolic cosine
$\delta(.)$	Kronecker delta function
$\frac{\mathrm{d}}{\mathrm{d}\,x}$, $\frac{\partial}{\partial x}$,	derivative, partial derivative with respect to x
$D_{\mathrm{JS}}^{(\pi_0,\pi_1)}\{.\,\vert\,.\}$	Jensen-Shannon divergence with weights (π_0, π_1)
$D_{\mathrm{KL}}\{.\,\vert\,.\}$	Kullback-Leibler divergence
$\exp(.)$	exponential function
$\mathbb{E}_{\mathsf{X}}\{f(x)\}$	expectation over $f(x)$
$f_{\mathrm{Q}}(\tilde{x})$	function $f_{\mathrm{Q}} : \tilde{\mathcal{X}} \to \mathcal{X}$ representing a scalar quantizer
GF(.)	Galois field of order .
$\mathrm{H}(\mathsf{X})$	entropy of random variable X
$\mathrm{H}(\mathsf{X}\vert\mathsf{Y})$	conditional entropy of random variable X given Y
$\mathrm{i}(\mathsf{X} = x)$	self-information of event $\mathsf{X} = x$
$\mathrm{I}(\mathsf{X};\mathsf{Y})$	mutual information of random variables X and Y
$\log(.)$, $\log_2(.)$	natural logarithm, binary logarithm
$L(.)$	LLR of .
$\max_x f(x)$	maximum of function $f(x)$
$\min_x f(x)$	minimum of function $f(x)$
mod .	modulo operator
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2
$p(x)$	probability distribution of random variable X
$p(x, y)$	joint probability distribution of random variables X and Y
$p(x\vert y)$	conditional probability distribution of X given Y
$\Pr(\mathsf{X} = x)$	probability of event $\mathsf{X} = x$
$\mathrm{rank}_2(.)$	rank of a matrix in GF(2)
$\mathrm{Re}\{.\}$	real part of complex number .
$\sup\{.\}$	supremum of .
x	realization of random variable X

X	discrete random variable X
$\mathcal{X}$	event space of random variable X
$\|\mathcal{X}\|$	cardinality of set $\mathcal{X}$
$\tilde{x}$	A tilde is used to stress that a random variable is continuous.
$\mathcal{X} \times \mathcal{Y}$	Cartesian product of sets $\mathcal{X}$ and $\mathcal{Y}$
$\oplus$	binary sum operator (exclusive or operation of bits)
$\boxplus$	box-plus operator from [36]
$\sum\oplus$	binary sum (exclusive or operation of bits)
$\sum\boxplus$	box-plus sum
$\otimes$	convolution operator
$.^*$	complex conjugate of .
$\lceil . \rceil$	nearest integer larger than or equal to .

Acronyms

ALU	arithmetical and logical unit
ASK	amplitude shift keying
AWGN	additive white Gaussian noise
BPSK	binary phase shift keying
C6474	Texas Instruments TMS320C6474 DSP
DSP	digital signal processor
DVB-S	digital video broadcasting over satellite
DDR2 RAM	double data rate 2 random access memory
FB	feedback
FW	forward
KL-means	Kullback-Leibler means
L1, L1D, L2	level one, level one data, level two cache
LAN	local area network
LDPC	low-density parity-check
LLR	log-likelihood ratio
LU	lower-upper
MMSE	minimum mean squared error
QAM	quadrature amplitude modulation
SDR	software defined radio

Chapter 2

Preliminaries

This chapter provides an introduction to the mathematical and technical concepts used throughout the thesis. It starts with some fundamentals on probability theory and Bayesian statistics. Afterwards, the important concepts of information, entropy and mutual information are introduced. Preliminaries on binary LDPC codes and their analysis using density evolution are explained, before a section on factor graphs and the sum-product algorithm closes the chapter.

2.1 Fundamentals of Probability Theory and Bayesian Statistics

This section deals with fundamental terms of probability theory and Bayesian statistics. These mathematical concepts are required to introduce information theoretical terms later.

2.1.1 Random Variables and Probability Distributions

A random variable is a mathematical object to formally describe the outcome of a random experiment. In this thesis, a random variable is denoted by a capital sans serif letter, for example, X. X takes values $x \in \mathcal{X}$ which are termed *realizations* of X. The set $\mathcal{X}$ is called the *event space* of random variable X. In the case where $\mathcal{X}$ has a finite number of elements, X is termed a *discrete* random variable. In contrast, a random variable is called continuous, if it can take an infinite number of real values. In the following, we will first concentrate on discrete variables. A generalization of the corresponding concepts for continuous random variables is provided at the end of this section.

A discrete random variable X takes values $x \in \mathcal{X}$ with probability $\Pr(\mathsf{X} = x) > 0$. By definition, the event space $\mathcal{X}$ is complete in the sense that it contains all possible realizations x that X can take. As a direct consequence, the sum over all $\Pr(\mathsf{X} = x)$ fulfills

$$\sum_{x \in \mathcal{X}} \Pr(\mathsf{X} = x) = 1. \tag{2.1}$$

Equation (2.1) is often termed the law of total probability.

One of the most important mathematical objects used in this work is the probability distribution of a discrete random variable. The probability distribution of X is a function which is defined on the support $\mathcal{X}$ and simply assigns $\Pr(\mathsf{X} = x)$ to each $x \in \mathcal{X}$. In this thesis, it is compactly denoted as

$$p(x) = \Pr(\mathsf{X} = x) \quad \forall x \in \mathcal{X}. \tag{2.2}$$

Typically, the probability distribution of X is referred to only as *the distribution* of X. In some technical applications it is also common to term a probability distribution $p(x)$ a *belief* on X [37] because it provides the probabilities for all $x \in \mathcal{X}$.

2.1.2 Joint and Conditional Probability Distributions

When two random variables X and Y are considered, these two variables can interact. A complete characterization of this interaction is given by their *joint distribution*. Fully equivalent to the case where only one random variable is considered, the joint distribution is defined as

$$p(x, y) = \Pr(\mathsf{X} = x, \mathsf{Y} = y) \quad \forall (x, y) \in \mathcal{X} \times \mathcal{Y}. \tag{2.3}$$

The joint distribution $p(x, y)$ assigns the probability of the *joint event* $\mathsf{X} = x$ and $\mathsf{Y} = y$ to any tuple $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Similarly as in (2.1), there is a law of total probability, that is,

$$\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) = 1. \tag{2.4}$$

The distributions $p(x)$ and $p(y)$ can be obtained from $p(x, y)$ by *marginalization* of the joint distribution $p(x, y)$ as

$$p(x) = \sum_{y \in \mathcal{Y}} p(x, y), \tag{2.5}$$

$$p(y) = \sum_{x \in \mathcal{X}} p(x, y). \tag{2.6}$$

The probability distributions $p(x)$ and $p(y)$ are called the marginal distributions of $p(x, y)$. The joint distribution $p(x, y)$ characterizes the co-occurrence of X and Y in a probabilistic manner.

Although the joint distribution $p(x, y)$ fully characterizes the probabilistic connection between X and Y, often more intuitive insights can be gained by rather considering the *conditional distribution* of X given Y. It is defined as

$$p(x|y) = \frac{p(x,y)}{p(y)} = \Pr(\mathsf{X} = x|\mathsf{Y} = y) \quad \forall (x,y) \in \mathcal{X} \times \mathcal{Y}. \tag{2.7}$$

The conditional distribution $p(x|y)$ assigns the probability $\Pr(\mathsf{X} = x|\mathsf{Y} = y)$ of event $\mathsf{X} = x$ for a given $\mathsf{Y} = y$ to (x, y). Knowing the joint distribution $p(x, y)$, it is simple to determine the conditional distribution $p(x|y)$ as

$$p(x|y) = \frac{p(x,y)}{\sum\limits_{x \in \mathcal{X}} p(x,y)}. \tag{2.8}$$

If the conditional distribution $p(x|y)$ differs from $p(x)$ for any $y \in \mathcal{Y}$, this tells that X is *not* statistically independent of Y. In contrast, knowing Y changes the belief on X. If two variables X and Y are statistically independent, their joint distribution factorizes as

$$p(x, y) = p(x)p(y) \tag{2.9}$$

and as a direct consequence $p(x|y)$ equals $p(x)\,\forall y \in \mathcal{Y}$.

Equivalently to (2.7) one can define the conditional distribution of Y given X as

$$p(y|x) = \frac{p(x,y)}{p(x)}. \tag{2.10}$$

With this definition, one obtains a central theorem of Bayesian statistics which is commonly known as *the Bayes rule*. The Bayes rule tells that

$$p(y|x)p(x) = p(x,y) = p(x|y)p(y) \Leftrightarrow p(y|x) = p(x|y)\frac{p(y)}{p(x)}. \tag{2.11}$$

So far, only random variables have been considered which can take scalar values. However, all of the introduced concepts can also be applied to describe multivariate random variables which have tuples or, equivalently, vectors as realizations.

Consider a multivariate X which takes tuples $x = (x_0, x_1, \ldots, x_{N-1})$ as realizations, where all x_n are from some finite set $\mathcal{X}_n$. Then a probability distribution of X can be defined as

$$\begin{aligned} &p(x) = p(x_0, x_1, \ldots, x_{N-1}) = \\ &\Pr(\mathsf{X}_0 = x_0, \mathsf{X}_1 = x_1, \ldots, \mathsf{X}_{N-1} = x_{N-1}) \quad \forall x \in \mathcal{X}_0 \times \mathcal{X}_1 \times \ldots \times \mathcal{X}_{N-1}. \end{aligned} \tag{2.12}$$

As it can be seen in Equation (2.12), the distribution $p(x)$ simply is a different interpretation of a joint probability distribution $p(x_0, x_1, \ldots, x_{N-1})$ of N scalar random variables X_n.

2.1.3 Continuous Random Variables

As it has been mentioned already, there exist *discrete* and *continuous* random variables. Continuous random variables and their realizations will often be marked with a tilde throughout the thesis, for example, random variable $\tilde{\mathsf{X}}$ with realization $\tilde{x} \in \tilde{\mathcal{X}}$. Discrete and continuous random variables can be characterized using quite similar, but not identical concepts.

A continuous random variable $\tilde{\mathsf{X}}$ can take an infinite number of values from a subset of the real numbers. Therefore, the probability that a particular value from its event space occurs is zero in general. In contrast,

$$\Pr\left(\tilde{\mathsf{X}} \leq \tilde{x}\right) = F(\tilde{x}) \geq 0. \tag{2.13}$$

The function $F(\tilde{x})$ is termed the cumulative distribution function of $\tilde{\mathsf{X}}$. Using the definition of $F(\tilde{x})$ and considering two $\tilde{x}_a, \tilde{x}_b \in \tilde{\mathcal{X}}$, where $\tilde{x}_b > \tilde{x}_a$, it is easy to understand that the probability that a realization $\tilde{x}$ lies between $\tilde{x}_a$ and $\tilde{x}_b$ is given by

$$\Pr\left(\tilde{\mathsf{X}} \in]\tilde{x}_a, \tilde{x}_b]\right) = F(\tilde{x}_b) - F(\tilde{x}_a). \tag{2.14}$$

Based on this, one considers a small $\Delta > 0$ and two possible realizations $\tilde{x}_b = \tilde{x} + \Delta$ and $\tilde{x}_a = \tilde{x}$ which lie very close to each other. From this one defines the differential quotient

$$p(\tilde{x}) = \lim_{\Delta \to 0} \frac{F(\tilde{x} + \Delta) - F(\tilde{x})}{\Delta} = \frac{\mathrm{d}}{\mathrm{d}\,\tilde{x}} F(\tilde{x}) \tag{2.15}$$

as the *probability density function* $p(\tilde{x})$ of $\tilde{\mathsf{X}}$. With the definition of $p(\tilde{x})$, similar as for the discrete case, there is a law of total probability which reads

$$\int_{-\infty}^{+\infty} p(\tilde{x})\,\mathrm{d}\,\tilde{x} = 1. \tag{2.16}$$

This indicates that the probability density function $p(\tilde{x})$ acts similar to the probability distribution $p(x)$ of a discrete random variable. However, typically integrals step into the position of sums in the equations involving continuous random variables. The concepts of joint and conditional probability distributions can be extended to continuous variables similarly.

2.2 Fundamentals of Information Theory

A very fundamental question of information theory is how to quantify information adequately. This section provides the information measures used throughout the thesis. The focus lies on information measures for discrete random variables.

2.2.1 Quantifying Information

For a random variable X with probability distribution $p(x)$, we aim to quantify the amount of information which observing a particular event $\mathsf{X} = x$ provides to an observer who knows $p(x)$. The *self-information* of event $\mathsf{X} = x$ is defined as

$$\mathrm{i}(\mathsf{X} = x) = -\log \Pr(\mathsf{X} = x) = -\log p(x), \tag{2.17}$$

To illustrate the idea, consider the following thought experiment.

Example 2.2.1.1. A binary X with $x \in \{0, 1\}$ is considered and a potential observer of X is told that $\Pr(\mathsf{X} = 0) = 99\%$ while $\Pr(\mathsf{X} = 1) = 1\%$ before having a chance to get a glance at a particular realization $x \in \mathcal{X}$. The observer now knows that in most cases $\mathsf{X} = 0$. Finally, the observer also observes the event $\mathsf{X} = 0$. Clearly, the observer is not really surprised by the result because he or she had expected it. Therefore, the observer has gained only little new *information*. In contrast, if $\mathsf{X} = 1$ had been observed, the observer would have gained a lot of new information. The self-information $\mathrm{i}(\mathsf{X} = 0) \approx 0.01 << \mathrm{i}(\mathsf{X} = 1) \approx 4.61$ quantifies the amount of information provided by both events adequately. — *End of the example.*

It is quite common to add a pseudo-unit to information measures. This pseudo-unit is determined by the base of the logarithm used in Equation (2.17). In communications, typically the $\log_2$ is used and the self-information is measured in the pseudo-unit bit. If, as it was done in the aforementioned example, the natural logarithm is used, the self-information is measured in the natural unit of information which is abbreviated nat.

2.2.2 Expected Information, Entropy

Instead of considering only a single event $\mathsf{X} = x$, a natural extension is asking what amount of information will be provided to an observer of X *on average*. This is measured by the *entropy* $\mathrm{H}(\mathsf{X})$ of random variable X. The entropy is the expected amount of information provided by X, that is,

$$\mathrm{H}(\mathsf{X}) = \mathbb{E}_{\mathsf{X}}\{\mathrm{i}(\mathsf{X} = x)\} = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{2.18}$$

In Equation (2.18), $\mathbb{E}_{\mathsf{X}}\{.\}$ denotes the expectation with respect to X. A vivid interpretation of $\mathrm{H}(\mathsf{X})$ is a measure of *uncertainty*. A random variable with high uncertainty provides a lot of information to a potential observer, while a random variable with low uncertainty can only provide a small amount of expected information.

Example 2.2.2.1. For the random variable X considered in Example 2.2.1.1, the entropy $\mathrm{H}(\mathsf{X}) = -0.99 \cdot \log 0.99 - 0.01 \log 0.01 \approx 0.06$ nat. The random variable does only provide a minor amount of information to an observer on average. In contrast, maximum entropy $\mathrm{H}(\mathsf{X})$ is achieved for $\Pr(\mathsf{X} = 0) = \Pr(\mathsf{X} = 1) = 0.5$. In this case $\mathrm{H}(\mathsf{X}) \approx 0.7$ nat. When $\mathrm{H}(\mathsf{X})$ is expressed in bit in this case, $\mathrm{H}(\mathsf{X}) = 1$ bit.

2.2.3 Joint and Conditional Entropy

For two random variables X and Y, distinct kinds of entropies can be considered. First, the expected amount of information from observing X and Y together can be considered. This is quantified by the *joint entropy* of X and Y, which is given by

$$\mathrm{H}(\mathsf{X}, \mathsf{Y}) = \mathbb{E}_{\mathsf{X},\mathsf{Y}}\{\mathrm{i}(\mathsf{X} = x, \mathsf{Y} = y)\} = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y). \tag{2.19}$$

Second, the average amount of information that follows from observing X *after* having observed Y can be quantified. This is expressed by the *conditional entropy* of X given Y. The conditional entropy of X given Y is defined as

$$\mathrm{H}(\mathsf{X}|\mathsf{Y}) = \mathbb{E}_{\mathsf{Y}}\{\mathrm{H}(\mathsf{X}|\mathsf{Y} = y)\} = -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log p(x|y). \tag{2.20}$$

Of course, one can equivalently define $\mathrm{H}(\mathsf{Y}|\mathsf{X})$ by a straightforward adaption of (2.20) as $\mathrm{H}(\mathsf{Y}|\mathsf{X}) = \mathbb{E}_{\mathsf{X}}\{\mathrm{H}(\mathsf{Y}|\mathsf{X} = x)\}$.

2.2.4 Mutual Information

With the notions of entropy and conditional entropy, it is possible to introduce the most important measure used throughout the entire thesis. This measure is Shannon's *mutual information*. Despite mutual information was first used by Shannon in his landmark paper from 1948 [38], mutual information still is one of the most important measures in modern communications and will play the central role from now on. Above it has been discussed that two random variables X and Y can interact with each other and that the average amount of information of X and Y can be quantified using $\mathrm{H}(\mathsf{X})$ and $\mathrm{H}(\mathsf{Y})$, whereas $\mathrm{H}(\mathsf{X}|\mathsf{Y})$ and $\mathrm{H}(\mathsf{Y}|\mathsf{X})$ quantify the remaining uncertainties of one variable given the other. The mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ now is a measure which quantifies how informative X is about Y and vice versa. The mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ between X and Y is defined as

$$\begin{aligned} \mathrm{I}(\mathsf{X};\mathsf{Y}) &= \mathrm{H}(\mathsf{X}) - \mathrm{H}(\mathsf{X}|\mathsf{Y}) && (2.21)\\ &= \mathrm{H}(\mathsf{Y}) - \mathrm{H}(\mathsf{Y}|\mathsf{X}). && (2.22) \end{aligned}$$

With the provided understanding of entropy and conditional entropy, an interpretation of Equations (2.21) and (2.22) is the following: $\mathrm{I}(\mathsf{X};\mathsf{Y})$ is the average information an observer gets about X from observing Y and also from observing Y about X. Expanding all entropies in Equation (2.21) or Equation (2.22) yields

$$\mathrm{I}(\mathsf{X};\mathsf{Y}) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}. \tag{2.23}$$

Just as for information and entropies, it is quite common to add a pseudo-unit, bit or nat, to the mutual information which is again determined by the respective base of the logarithm used in Equation (2.23).

Considering two extreme cases shows that $\mathrm{I}(\mathsf{X};\mathsf{Y})$ in fact measures the mutual coupling of X and Y. First consider X and Y to be statistically independent. Due to the factorization of joint distribution $p(x,y)$ from Equation (2.9), it follows from Equation (2.23) that $\mathrm{I}(\mathsf{X};\mathsf{Y}) = 0$. This tells that knowing Y does not allow any inference on X and vice versa due to the fact that X and Y are independent. The second considered extreme case is that X and Y are coupled by the relation $\mathsf{X} = \mathsf{Y}$. In this case, knowing X tells everything about Y and vice versa and, therefore, the following proposition holds.

Proposition 2.2.1. *Mutual information of identical variables*

If $\mathsf{X} = \mathsf{Y}$, then

$$\mathrm{I}(\mathsf{X};\mathsf{Y}) = \mathrm{H}(\mathsf{X}) = \mathrm{H}(\mathsf{Y}). \tag{2.24}$$

Proof: See Appendix A.1.

The mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ can also be expressed using a distance measure for probability distributions which is termed the *Kullback-Leibler divergence*. For two probability distributions $p(x)$ and $q(x)$ defined over the same event space $\mathcal{X}$, the Kullback-Leibler divergence is given by

$$D_{\mathrm{KL}}\{p(x) \,|\, q(x)\} = \sum_{x\in\mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{2.25}$$

The definition in Equation (2.25) allows to rewrite the mutual information from Equation (2.23) as

$$\mathrm{I}(\mathsf{X};\mathsf{Y}) = D_{\mathrm{KL}}\{p(x,y) \,|\, p(x)p(y)\}. \tag{2.26}$$

2.2.5 Conditional Mutual Information

Of course, random variables X and Y with mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ can be multivariate variables. Consider any random variable X and a two dimensional $\mathsf{Y} = (\mathsf{Y}_0, \mathsf{Y}_1)$ with realization $y = (y_0, y_1)$. Then X and Y share the mutual information

$$\mathrm{I}(\mathsf{X};\mathsf{Y}_0,\mathsf{Y}_1) = \sum_{x\in\mathcal{X}} \sum_{y_0\in\mathcal{Y}_0} \sum_{y_1\in\mathcal{Y}_1} p(x,y_0,y_1) \log \frac{p(x,y_0,y_1)}{p(x)p(y_0,y_1)}. \tag{2.27}$$

Equation (2.27) can be rewritten to derive an important relation which is stated in the following proposition.

Proposition 2.2.2. *The chain rule of mutual information*

$$\mathrm{I}(\mathsf{X};\mathsf{Y}_0,\mathsf{Y}_1) = \mathrm{I}(\mathsf{X};\mathsf{Y}_0|\mathsf{Y}_1) + \mathrm{I}(\mathsf{X};\mathsf{Y}_1), \tag{2.28}$$

where $\mathrm{I}(\mathsf{X};\mathsf{Y}_0|\mathsf{Y}_1)$ denotes the so called *conditional mutual information*

$$\mathrm{I}(\mathsf{X};\mathsf{Y}_0|\mathsf{Y}_1) = \mathbb{E}_{\mathsf{Y}_1}\{\mathrm{I}(\mathsf{X};\mathsf{Y}_0)|\mathsf{Y}_1 = y_1\} = \sum_{y_1\in\mathcal{Y}_1} p(y_1) \sum_{x\in\mathcal{X}} \sum_{y_0\in\mathcal{Y}_0} p(x,y_0|y_1) \log \frac{p(x,y_0|y_1)}{p(x|y_1)p(y_0|y_1)}. \tag{2.29}$$

Proof: See Appendix A.2.

The conditional mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y}_0|\mathsf{Y}_1)$ from Equation (2.29) quantifies the amount of additional information that can be gained about X from getting to know Y_0 if Y_1 is already known.

2.3 Preliminaries on Binary Low-Density Parity-Check Codes

Binary LDPC codes were first presented by Gallager in 1963 as a central part of his Ph.D. thesis [39]. However, they have been neglected for around 30 years after their discovery, mostly due to the lack of the required computational resources for their decoding. In the 1990's the discovery of turbo codes [40] drastically changed the meaning of channel coding because for the first time a practical scheme was discovered which offered error correction performance close to the theoretical Shannon capacity with affordable decoding complexity. Also around this time, MacKay and Neal rediscovered LDPC codes [41] and it was revealed soon, that carefully designed LDPC codes can outperform turbo coding schemes in terms of their error correction performance. This pushed the development of LDPC codes and they are today still some of the best known channel codes. Due to this, they have been used in several communication standards in the past, for example, the IEEE 802.11 wireless LAN [42] and the ETSI DVB-S2 [43] standards. Moreover, they are still considered for future standards.

In the following, binary LDPC codes are introduced similarly as in [44]. Binary LDPC codes are defined over the Galois field GF(2). An extension of LDPC codes over higher order Galois fields is known and described in [45], but will not be described in this thesis in detail. In the following, when referring to LDPC codes, the attribute *binary* will, therefore, be neglected.

2.3.1 Mathematical Description

An LDPC code is defined as the set of all length N_v column vectors $\mathbf{c}$ which fulfil

$$\mathbf{H} \cdot \mathbf{c} = \mathbf{0}_{N_\mathrm{c} \times 1}, \tag{2.30}$$

where $\mathbf{H} \in \{0,1\}^{N_\mathrm{c} \times N_\mathrm{v}}$ is a binary parity-check matrix with N_c rows and N_v columns and $\mathbf{0}_{N_\mathrm{c} \times 1}$ is a length N_c column vector which holds only zeros. The arithmetic in Equation (2.30) follows the rules in GF(2), that is, the sum in the matrix-vector product is a modulo 2 sum which corresponds to the binary XOR-operation of the codeword bits. The matrix $\mathbf{H}$ implies a total of N_c parity-check conditions on the N_v codeword bits in the codeword $\mathbf{c}$. $\mathbf{H}$ is considered to be sparse in the sense that the total number of non-zero elements of $\mathbf{H}$ is only a small fraction of its total number of elements.

The numbers of non-zero elements in the rows and the columns are termed the row weights and the column weights of $\mathbf{H}$, respectively. If all columns of $\mathbf{H}$ have an identical weight and also all rows have an identical weight, an LDPC code is called *regular*. Otherwise it is called *irregular*. The difference between regular and irregular LDPC codes is illustrated in the following example.

Example 2.3.1.1. Consider the two distinct parity-check matrices

$$\mathbf{H}^{(\mathrm{a})} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{H}^{(\mathrm{b})} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

with $(N_\mathrm{c}, N_\mathrm{v}) = (6, 8)$. The row weight of all rows in $\mathbf{H}^{(\mathrm{a})}$ is 4 and the column weight of all its columns is 3. Therefore, $\mathbf{H}^{(\mathrm{a})}$ is a parity-check matrix of a regular LDPC code. In contrast, not all row weights and not all column weights of $\mathbf{H}^{(\mathrm{b})}$ are identical. From the left to the right, the column weights of $\mathbf{H}^{(\mathrm{b})}$ are $(4, 4, 3, 3, 3, 3, 3, 3)$ and the row weights are $(4, 4, 5, 5, 4, 4)$ from the top to the bottom. As a result, $\mathbf{H}^{(\mathrm{b})}$ is a parity-check matrix of an irregular LDPC code. — *End of the example.*

It is worth mentioning that matrices $\mathbf{H}^{(\mathrm{a})}$ and $\mathbf{H}^{(\mathrm{b})}$ were chosen for exemplary illustration, but are not really sparse matrices because they are too small. Anyway, they enable to explain fundamental principles of LDPC codes, as it is done in the following.

It is very useful to visualize a parity-check matrix $\mathbf{H}$ in a Tanner graph. A Tanner graph is a bipartite graph which consists of variable nodes and

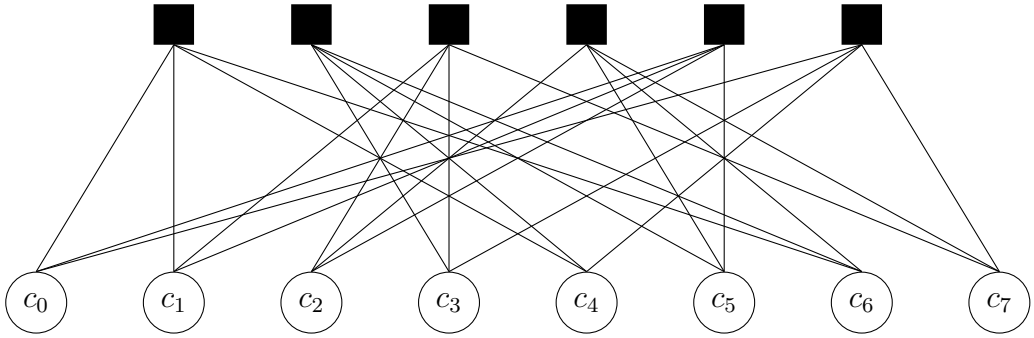
check nodes. Let $H_{m,n}$ denote the element in row m and column n of an arbitrary $N_\mathrm{c} \times N_\mathrm{v}$ parity-check matrix $\mathbf{H}$. Each check node corresponds to a row $m \in \{0, 1, \ldots, N_\mathrm{c} - 1\}$ of $\mathbf{H}$ and each column $n \in \{0, 1, \ldots, N_\mathrm{v} - 1\}$ corresponds to a variable node in the Tanner graph of $\mathbf{H}$. The check nodes are typically denoted by square symbols and the variable nodes are denoted by circle symbols. An edge is placed between check node m and variable node n, if $H_{m,n} = 1$. The following example illustrates the principle of visualizing parity-check matrices in Tanner graphs.

Example 2.3.1.2. The Tanner graph of $\mathbf{H}^{(\mathrm{a})}$ is depicted in Figure 2.1a and the Tanner graph of $\mathbf{H}^{(\mathrm{b})}$ is depicted in Figure 2.1b. The check nodes are drawn at the top and the variable nodes are placed at the bottom of the Tanner graphs. For the regular code in Figure 2.1a, the number of edges connected to all check nodes is identical and also the number of edges connected to all variable nodes is identical. It equals to the respective row or column weight of $\mathbf{H}^{(\mathrm{a})}$. Also for the irregular code in Figure 2.1b the row weights determine the number of connected edges to the check nodes and the column weights determine the number of connected edges to the variable nodes. However, as the row and column weights differ, not all nodes of a type have an identical number of connected edges. — *End of the example.*

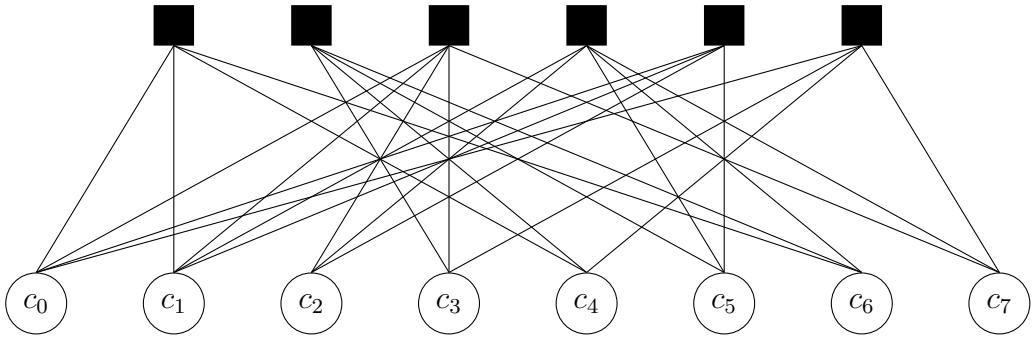
The number of edges connected to a certain node is termed the *degree* of this node. For a regular LDPC code, it is sufficient to consider a single variable node degree d_v and a single check node degree d_c because all nodes of a certain type share the same degree. For an irregular code, it is required to distinguish the degrees $d_\mathrm{c}^{(m)}$, $m \in \{0, 1, \ldots, N_\mathrm{c} - 1\}$ for each check node and $d_\mathrm{v}^{(n)}$, $n \in \{0, 1, \ldots, N_\mathrm{v} - 1\}$ for each variable node.

Rather than considering a particular parity-check matrix $\mathbf{H}$, it is often appropriate to analyze *ensembles* of LDPC codes. A $(d_\mathrm{v}, d_\mathrm{c})$ regular LDPC ensemble is characterized by all possible parity-check matrices $\mathbf{H}$ which imply a fixed variable node degree d_v and a fixed check node degree d_c. An irregular LDPC ensemble is determined by all parity-check matrices which have the same fractions of variable nodes and check nodes of certain degrees.

The description of irregular LDPC ensembles can basically be done in two distinct ways. The first one is considering the fractions ρ'_k of degree k check nodes and the fraction λ'_l of degree l variable nodes. A different, but more



(a) Tanner graph of parity-check matrix $\mathbf{H}^{(\mathrm{a})}$. Since $\mathbf{H}^{(\mathrm{a})}$ describes a regular LDPC code, all variable nodes (circle nodes) have the same number of connected edges and all check nodes (square nodes) have the same number of connected edges.



(b) Tanner graph of parity-check matrix $\mathbf{H}^{(\mathrm{b})}$. Since $\mathbf{H}^{(\mathrm{b})}$ describes an irregular LDPC code, not all variable nodes (circle nodes) have the same number of connected edges and not all check nodes (square nodes) have the same number of connected edges.

Figure 2.1: Tanner graphs of parity-check matrices $\mathbf{H}^{(\mathrm{a})}$ and $\mathbf{H}^{(\mathrm{b})}$.

frequently used perspective is considering the fraction ρ_k of edges connected to degree k check nodes and the fraction λ_l of edges connected to degree l variable nodes. Using these measures, it is possible to characterize irregular LDPC ensembles using polynomials in some variable d which have ρ_k and λ_l as coefficients. These polynomials are defined as

$$\rho(d) = \sum_{k=2}^{d_{\mathrm{c}}^{\max}} \rho_k d^{k-1}, \tag{2.31}$$

$$\lambda(d) = \sum_{l=1}^{d_{\mathrm{v}}^{\max}} \lambda_l d^{l-1}. \tag{2.32}$$

In Equations (2.31) and (2.32), $d_{\mathrm{c}}^{\max}$ and $d_{\mathrm{v}}^{\max}$ denote the maximum appearing check node and variable node degrees. Despite the polynomials in Equations (2.31) and (2.32) are not probability distributions in a strict mathematical sense, they are typically termed the check node and the variable node *degree distributions* of an irregular LDPC ensemble. Sometimes the attribute *from an edge perspective* is added or the term *edge degree distribution* is used to stress the chosen perspective. Both perspectives on the ensemble characteristics are discussed for parity-check matrix $\mathbf{H}^{(\mathrm{b})}$ in the following example.

Example 2.3.1.3. With the row weights of $\mathbf{H}^{(\mathrm{b})}$ it is simple to calculate the fractions ρ'_k as $\rho'_2 = 0, \rho'_3 = 0, \rho'_4 = 2/3, \rho'_5 = 1/3$ by dividing the number of degree k check nodes by the total number of check nodes. Similarly, one finds $\lambda'_1 = 0, \lambda'_2 = 0, \lambda'_3 = 3/4, \lambda'_4 = 1/4$ using the column weights of $\mathbf{H}^{(\mathrm{b})}$.

The total number of edges in the Tanner graph of a parity-check matrix equals the sum of all row weights which is also identical to the the sum of all its column weights. This reveals that there are 26 edges in the Tanner graph of $\mathbf{H}^{(\mathrm{b})}$. The row weights of $\mathbf{H}^{(\mathrm{b})}$ show that four check nodes have degree four. Therefore, 16 edges are connected to degree four check nodes. The remaining ten edges are connected to degree five check nodes. As a result, $\rho_2 = 0, \rho_3 = 0, \rho_4 = 8/13, \rho_5 = 5/13$.

The column weights of $\mathbf{H}^{(\mathrm{b})}$ allow to calculate the fractions λ_l. Two variable nodes have degree four. Therefore, eight edges are connected to degree four variable nodes. The remaining 18 edges are connected to degree three variable nodes. As a result, $\lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 9/13, \lambda_4 = 4/13$. Finally, this yields

$$\rho(d) = \frac{8}{13}d^3 + \frac{5}{13}d^4, \quad \lambda(d) = \frac{9}{13}d^2 + \frac{4}{13}d^3.$$

— *End of the example.*

2.3.2 Encoding and Code Rate

LDPC codes are linear block codes. Therefore, it is possible to obtain a length N_{v} codeword $\mathbf{c}$ from K information bits $\mathbf{u}$ using the linear transformation

$$\mathbf{c} = \mathbf{G} \cdot \mathbf{u}, \tag{2.33}$$

where $\mathbf{G} \in \{0,1\}^{N_\mathrm{v} \times K}$ is an $N_\mathrm{v} \times K$ generator matrix of the code. The generator matrix $\mathbf{G}$ fulfills

$$\mathbf{H} \cdot \mathbf{G} = \mathbf{0}_{N_\mathrm{c} \times K}. \tag{2.34}$$

In the following, it is assumed that $\mathbf{H}$ has full row rank, that is, $\mathrm{rank}_2(\mathbf{H}) = N_\mathrm{c}$, meaning that all rows in $\mathbf{H}$ are linearly independent. A simple way to determine $\mathbf{G}$ for a given $\mathbf{H}$ is performing Gauss-Jordan elimination on the last $N_\mathrm{v} - K$ columns of $\mathbf{H}$ to transform these columns into an $(N_\mathrm{v} - K) \times (N_\mathrm{v} - K)$ eye matrix $\mathbf{I}_{N_\mathrm{v}-K}$ which has only ones on its diagonal and zeros elsewhere. This step yields a modified parity-check matrix $\mathbf{H}'$ with form

$$\mathbf{H}' = \begin{bmatrix} \mathbf{A}_{N_\mathrm{c} \times K} & \mathbf{I}_{N_\mathrm{v}-K} \end{bmatrix}. \tag{2.35}$$

From $\mathbf{H}'$ a systematic generator matrix $\mathbf{G}$ can be found as

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{A}_{\mathrm{N_c} \times K} \end{bmatrix}. \tag{2.36}$$

As it can be seen from the structure of $\mathbf{G}$, the first K bits in $\mathbf{c} = \mathbf{G} \cdot \mathbf{u}$ equal to the uncoded bits $\mathbf{u}$ and the last N_c bits are parities. Please note that Equation (2.35) also reveals that $N_\mathrm{c} = N_\mathrm{v} - K$. Therefore, for a full rank parity-check matrix $\mathbf{H}$, the code rate R is

$$R = \frac{K}{N_\mathrm{v}} = \frac{N_\mathrm{v} - N_\mathrm{c}}{N_\mathrm{v}} = 1 - \frac{N_\mathrm{c}}{N_\mathrm{v}}. \tag{2.37}$$

For a $(d_\mathrm{v}, d_\mathrm{c})$ regular LDPC code, the overall number of ones in $\mathbf{H}$ which equals the total number of edges in its Tanner graph is $N_\mathrm{c} d_\mathrm{c} = N_\mathrm{v} d_\mathrm{v}$. From this, it is easy to obtain the ratio $N_\mathrm{c}/N_\mathrm{v}$ as a function of the node degrees d_v and d_c. Plugging the result into Equation (2.37) provides the code rate

$$R = 1 - \frac{N_\mathrm{c}}{N_\mathrm{v}} = 1 - \frac{d_\mathrm{v}}{d_\mathrm{c}}. \tag{2.38}$$

Similarly, for an irregular code the number of ones in $\mathbf{H}$ is $N_\mathrm{c} \sum_{k=2}^{d_\mathrm{c}^\mathrm{max}} k\rho'_k = N_\mathrm{v} \sum_{l=1}^{d_\mathrm{v}^\mathrm{max}} l\lambda'_l$. Again, solving for $N_\mathrm{c}/N_\mathrm{v}$ and plugging the result into Equation (2.37) yields the code rate

$$R = 1 - \frac{\sum\limits_{l=1}^{d_\mathrm{v}^\mathrm{max}} l\lambda'_l}{\sum\limits_{k=2}^{d_\mathrm{c}^\mathrm{max}} k\rho'_k} = 1 - \frac{\sum\limits_{k=2}^{d_\mathrm{c}^\mathrm{max}} \rho_k/k}{\sum\limits_{l=1}^{d_\mathrm{v}^\mathrm{max}} \lambda_l/l}. \tag{2.39}$$

The derivation of the second equality in Equation (2.39) can be found in Appendix A.3.

Example 2.3.2.1. Leveraging Equation (2.38) one finds that the code rate R for the regular LDPC code implied by $\mathbf{H}^{(\mathrm{a})}$ is $R = 1 - \frac{3}{4} = \frac{1}{4}$. Equation (2.39) tells that the rate R of the irregular LDPC code described by $\mathbf{H}^{(\mathrm{b})}$ is also

$$R = 1 - \frac{\frac{8}{4 \cdot 13} + \frac{5}{5 \cdot 13}}{\frac{9}{3 \cdot 13} + \frac{4}{4 \cdot 13}} = \frac{1}{4}.$$

By application of row additions and row swaps, the last $N_{\mathrm{v}} - K = 6$ columns of $\mathbf{H}^{(\mathrm{a})}$ and $\mathbf{H}^{(\mathrm{b})}$ can be transformed into eye matrices as

$$\mathbf{H}^{(\mathrm{a})'} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{H}^{(\mathrm{b})'} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

From this, application of Equation (2.36) yields the systematic generator matrices

$$\mathbf{G}^{(\mathrm{a})} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{G}^{(\mathrm{b})} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

— *End of the example.*

A shortcoming of explicitly determining the generator matrix $\mathbf{G}$ for encoding is that unlike $\mathbf{H}$, the matrix $\mathbf{G}$ is in general not sparse. This results in a high encoding complexity [44].

Another simple method for encoding with linear complexity in the codeword length is based on a sparse lower-upper (LU) decomposition of a submatrix of

$\mathbf{H}$ and described in [46]. For this method, first $\mathbf{H}$ is separated into its first K and its last $N_\mathrm{v} - K$ columns, such that

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{(N_\mathrm{v}-K)\times K} & \mathbf{H}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)} \end{bmatrix}. \tag{2.40}$$

Likewise, the codeword $\mathbf{c}$ is separated into its systematic part and its parities

$$\mathbf{c} = \begin{bmatrix} \mathbf{u} \\ \mathbf{c}_{N_\mathrm{v}-K} \end{bmatrix}. \tag{2.41}$$

Hence, their product

$$\mathbf{H}\cdot\mathbf{c} = \begin{bmatrix} \mathbf{H}_{(N_\mathrm{v}-K)\times K} & \mathbf{H}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u} \\ \mathbf{c}_{N_\mathrm{v}-K} \end{bmatrix} = \mathbf{0}_{(N_\mathrm{v}-K)\times 1}. \tag{2.42}$$

According to the arithmetical rules in GF(2), this is equivalent to

$$\mathbf{H}_{(N_\mathrm{v}-K)\times K}\cdot\mathbf{u} = \mathbf{H}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)}\cdot\mathbf{c}_{N_\mathrm{v}-K}. \tag{2.43}$$

Equation (2.43) reveals that for a given uncoded sequence $\mathbf{u}$, the missing $(N_\mathrm{v}-K)$ codeword bits $\mathbf{c}_{N_\mathrm{v}-K}$ can be determined, if $\mathbf{H}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)}$ is invertible in GF(2). In this case the multiplication of Equation (2.43) from the left by the inverse $\mathbf{H}^{-1}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)}$ would deliver $\mathbf{c}_{N_\mathrm{v}-K}$, but $\mathbf{H}^{-1}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)}$ is dense in general.

The sparse LU decomposition of $\mathbf{H}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)}$ with so called pivoting can be used to determine the missing codeword bits $\mathbf{c}_{N_\mathrm{v}-K}$ efficiently. This decomposition of $\mathbf{H}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)}$ is defined by the equation

$$\mathbf{P}\cdot\mathbf{H}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)} = \mathbf{L}\cdot\mathbf{U}, \tag{2.44}$$

where $\mathbf{P}$ is an $(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)$ row permutation matrix, $\mathbf{L}$ is an $(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)$ lower triangular matrix and $\mathbf{U}$ is an $(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)$ upper triangular matrix. One aims for $\mathbf{L}$ and $\mathbf{U}$ being as sparse as possible by choosing a proper algorithm to find a sparse LU decomposition [46].

Multiplication of Equation (2.43) by $\mathbf{P}$ from the left yields

$$\mathbf{P}\cdot\mathbf{H}_{(N_\mathrm{v}-K)\times K}\cdot\mathbf{u} = \underbrace{\mathbf{P}\cdot\mathbf{H}_{(N_\mathrm{v}-K)\times(N_\mathrm{v}-K)}}_{\mathbf{L}\cdot\mathbf{U}}\cdot\mathbf{c}_{N_\mathrm{v}-K} = \mathbf{L}\cdot\mathbf{U}\cdot\mathbf{c}_{N_\mathrm{v}-K}. \tag{2.45}$$

Now let us define the vector

$$\mathbf{z} = \mathbf{U}\cdot\mathbf{c}_{N_\mathrm{v}-K}. \tag{2.46}$$

With this definition one obtains

$$\mathbf{L} \cdot \mathbf{z} = \mathbf{P} \cdot \mathbf{H}_{(N_\mathrm{v}-K)\times K} \cdot \mathbf{u}. \tag{2.47}$$

Note that on the right hand side of Equation (2.47) a product of a sparse matrix $\mathbf{P} \cdot \mathbf{H}_{(N_\mathrm{v}-K)\times K}$ and the known systematic bits $\mathbf{u}$ appears. Since now $\mathbf{L}$ has lower triangular form, it is simple to determine the elements in $\mathbf{z}$, i.e., $z_0, z_1, \ldots, z_{N_\mathrm{v}-K-1}$ successively by forward substitution. Once $\mathbf{z}$ is obtained, one leverages the upper triangular form of $\mathbf{U}$ in Equation (2.46), to determine the missing codeword bits $\mathbf{c}_{N_\mathrm{v}-K}$ by backward substitution. The principle of encoding by forward and backward substitution is illustrated in the following example.

Example 2.3.2.2. We consider encoding of the uncoded bits $\mathbf{u} = [1, 0]^\mathrm{T}$ with the LDPC code described by $\mathbf{H}^{(\mathrm{a})}$. The matrix $\mathbf{H}^{(\mathrm{a})}$ is split into its first K and its last $N_\mathrm{v} - K$ columns $\mathbf{H}^{(\mathrm{a})}$ yielding

$$\mathbf{H}^{(\mathrm{a})}_{(N_\mathrm{v}-K)\times K} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \mathbf{H}^{(\mathrm{a})}_{(N_\mathrm{v}-K)\times (N_\mathrm{v}-K)} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The LU decomposition of $\mathbf{H}^{(\mathrm{a})}_{(N_\mathrm{v}-K)\times (N_\mathrm{v}-K)}$ delivers

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{U} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The product $\mathbf{P} \cdot \mathbf{H}^{(\mathrm{a})}_{(N_\mathrm{v}-K)\times K} \cdot \mathbf{u} =$

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

This determines the right hand side of

$$\mathbf{L} \cdot \mathbf{z} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

Using forward substitution this directly reveals that $z_0 = 0$, $z_1 = 0$, $z_2 = 1$, $z_3 = 1$, $z_4 = 1, z_5 = 0$. With this intermediate result, one can perform backward substitution to obtain $\mathbf{c}_{N_\mathrm{v}-K}$ as

$$\mathbf{z} = \mathbf{U} \cdot \mathbf{c}_{N_\mathrm{v}-K} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

This reveals that $c_7 = 0, c_6 = 1, c_5 = 0, c_4 = 0, c_3 = 1, c_2 = 1$. Finally, the codeword is $\mathbf{c} = \left[\mathbf{u}^\mathrm{T}, \mathbf{c}^\mathrm{T}_{N_\mathrm{v}-K}\right]^\mathrm{T} = [1, 0, 1, 1, 0, 0, 1, 0]^\mathrm{T}$. —*End of the example.*

Efficient encoding of LDPC codes is a topic well studied in the literature. Another famous approach which requires a lot of preprocessing of the matrix $\mathbf{H}$ can be found in [47], but shall not be elaborated here in detail. The general idea of this approach is to use as many row and column permutations on $\mathbf{H}$ as possible in order to bring $\mathbf{H}$ to so called *approximate lower triangular form*. These operations do not affect the sparsity of $\mathbf{H}$ and this special form of $\mathbf{H}$ also allows for efficient encoding.

Moreover, a desired efficient encoding can be achieved by using LDPC codes with structured parity-check matrices $\mathbf{H}$ right from scratch. A well known example for this are repeat-accumulate codes [48, 49] which are widely used in practice. They allow to build encoders using only very simple repeater, interleaver and differential encoder blocks.

2.3.3 Message Passing Decoding Algorithms

LDPC codes can be decoded using iterative message passing algorithms on the Tanner graph of a parity-check matrix. In the following, first the belief propagation decoding algorithm is described. Afterwards, it is explained how this algorithm can be modified to obtain the min-sum decoding algorithm which is often used in practice due to its lower implementation complexity.

Consider the following setup. An LDPC codeword $\mathbf{c}$ is transmitted over a discrete-time memoryless channel which delivers a vector $\mathbf{r}$ of channel outputs. The channel is described by its transition probability distribution $p(\mathbf{r}|\mathbf{c})$. Since the channel is memoryless, this transition probability distribution factorizes as

$$p(\mathbf{r}|\mathbf{c}) = \prod_{n=0}^{N_\mathrm{v}-1} p(r_n|c_n). \tag{2.48}$$

The aim of iterative belief propagation decoding is to obtain maximum a posteriori estimates $\hat{c}_n$ for each codeword bit c_n in $\mathbf{c}$, i.e.,

$$\hat{c}_n = \arg\max_{c_n} p(c_n|\mathbf{r}) \, \forall n \in \{0, 1, \ldots, N_\mathrm{v} - 1\}. \tag{2.49}$$

The first step to perform belief propagation decoding is the calculation of channel log-likelihood ratios (LLRs) from the channel values $\mathbf{r}$. For the memoryless channel, they can be calculated as

$$L^{\mathrm{ch}}(c_n) = \log \frac{p(r_n|\mathsf{C}_n = 0)}{p(r_n|\mathsf{C}_n = 1)}. \tag{2.50}$$

LLRs are exchanged iteratively to successively generate extrinsic information in belief propagation decoding. In the first decoding iteration, each variable node n passes $L^{\mathrm{ch}}(c_n)$ to all its connected check nodes in the Tanner graph of $\mathbf{H}$.

Now consider an arbitrary check node m and let us denote its connected codeword bits $b_0, b_1, \ldots, b_{d_\mathrm{c}^{(m)}-1}$ to obtain a simple notation. The check node

receives LLRs $L(b_0),\ L(b_1), \ldots, L(b_{d_{\mathrm{c}}^{(m)}-1})$ along its connected edges n in the Tanner graph of $\mathbf{H}$. Using these LLRs, the node evaluates

$$L_{n'}^{\mathrm{ext}}(b_{n'}) = \sum_{n \neq n'}^{\boxplus} L(b_n) \quad \forall n' \in \left\{0, 1, \ldots, d_{\mathrm{c}}^{(m)} - 1\right\}, \tag{2.51}$$

where $\boxplus$ denotes the box-plus operation from [36]. The box-plus operation which is used in Equation (2.51) for two LLRs $L(b_0)$ and $L(b_1)$ is given by

$$L(b_0) \boxplus L(b_1) = \log \frac{1 + \exp\left(L(b_0) + L(b_1)\right)}{\exp\left(L(b_0)\right) + \exp(L(b_1))}. \tag{2.52}$$

This operation is used recursively in Equation (2.51) to obtain the box-plus sum of $d_{\mathrm{c}}^{(m)} - 1$ LLRs. The incoming message $L(b_{n'})$ received along the target edge n' is excluded in (2.51) because *extrinsic* information shall be generated which is passed back along this edge. Once, all check nodes have generated extrinsic LLRs for all their connected edges, these LLRs are passed back to the variable nodes.

Now consider an arbitrary variable node n which represents codeword bit c_n in the Tanner graph of $\mathbf{H}$ and has degree $d_{\mathrm{v}}^{(n)}$. This node receives $d_{\mathrm{v}}^{(n)}$ LLRs $L_0(c_n), L_1(c_n), \ldots, L_{d_{\mathrm{v}}^{(n)}-1}(c_n)$ along its connected edges m and also has access to the channel LLR $L^{\mathrm{ch}}(c_n)$. It is important that unlike for the check nodes, all incoming LLRs and the channel LLR represent the same codeword bit c_n. As a consequence, variable nodes sum up their received LLRs to generate extrinsic information. To do so, variable node n evaluates

$$L_{m'}^{\mathrm{ext}}(c_n) = L^{\mathrm{ch}}(c_n) + \sum_{m \neq m'} L_m(c_n) \quad \forall m' \in \left\{0, 1, \ldots, d_{\mathrm{v}}^{(n)} - 1\right\} \tag{2.53}$$

for its connected edges m'. Again, for each target edge m', the corresponding incoming message $L_{m'}(c_n)$ is excluded in Equation (2.53) because *extrinsic* information shall be generated.

The described message passing between variable and check nodes is repeated iteratively, until a certain exit criterion, for example, an allowed maximum number of decoder iterations is met. An alternative exit criterion is checking if all parity-checks are satisfied. Also a combination of both criteria can be used, such that the decoder performs a maximum of $i_{\max}$ decoding iterations or stops earlier, if all parity-checks are satisfied. Finally, the variable nodes perform their final bit decision which is a hard decision on the sum *of all* incoming messages as a posteriori information shall be obtained.

The computationally most expensive part of belief propagation decoding is the evaluation of the box-plus operation in Equation (2.52). However, one can rewrite this operation [2] as

$$L(b_0) \boxplus L(b_1) = \operatorname{sign}(L(b_0))\operatorname{sign}(L(b_1)) \min(|L(b_0)|, |L(b_1)|) + f_c(|L(b_0) + L(b_1)|) - f_c(|L(b_0) - L(b_1)|), \quad (2.54)$$

where $f_c(x)$ is a Jacobian logarithm correction function

$$f_c(x) = \log\left(1 + \exp(-x)\right). \quad (2.55)$$

Equation (2.54) exactly corresponds to $L(b_0) \boxplus L(b_1)$ and is not an approximation. However, its direct evaluation is still complex, as it requires evaluation of the log(.) and the exp(.) functions. Nevertheless, $f_c(x)$ can be implemented in a one dimensional lookup table, to avoid the explicit evaluation of the log(.) and exp(.) functions in $f_c(x)$ [2].

Neglecting the terms including $f_c(x)$ in (2.54) simplifies the check node operation and yields the so called *min-sum* approximation

$$L(b_0) \boxplus L(b_1) \approx \operatorname{sign}(L(b_0))\operatorname{sign}(L(b_1)) \min(|L(b_0)|, |L(b_1)|). \quad (2.56)$$

If this approximation is used for the check node operation in a message passing decoder, the resulting decoding algorithm is called the *min-sum* decoding algorithm. The min-sum decoding algorithm requires less computational resources than the belief propagation decoding algorithm, but also offers worse error correction capability. It shall be noted, however, that some adaptions can be applied to the min-sum decoding algorithm to improve its performance. These adaptions are based on scaling or adding offsets to the messages passed in the decoder [2,50]. However, while they improve the performance, they also add some additional complexity to the decoding.

Finally, the following numerical example describes one iteration of belief propagation decoding.

Example 2.3.3.1. Consider a binary symmetric error and erasure channel with erasure probability $p_?$ and error probability p_e. This channel inputs the codeword bits $c_n \in \{0, 1\}$ and has three possible outputs $r_n \in \{0, ?, 1\}$. The codeword bit c_n is received correctly with probability $1-p_?-p_\mathrm{e}$. The codeword

bit is erased with probability $p_?$ and it is received incorrectly with probability p_e. The considered numerical values for the channel transition probabilities in this example are $(1 - p_? - p_\mathrm{e}, p_?, p_\mathrm{e}) = (0.89, 0.1, 0.01)$.

The uncoded bits $\mathbf{u} = [1, 0]^\mathrm{T}$ are encoded with the LDPC code described by $\mathbf{H}^{(\mathrm{a})}$. The corresponding codeword $\mathbf{c} = [1, 0, 1, 1, 0, 0, 1, 0]^\mathrm{T}$ is transmitted over the channel. The received word is $\mathbf{c} = [1, ?, 1, 1, 0, 0, 1, 1]^\mathrm{T}$. As a result, there is one erasure in the systematic part of the codeword and the last parity bit is received incorrectly, as it is highlighted in red. According to Equation (2.50), the channel LLRs for the respective channel outputs $r_n \in \{0, ?, 1\}$ are

$$L^{\mathrm{ch}}(c_n) = \begin{cases} \log \frac{0.89}{0.01} \approx +4.49 & r_n = 0 \\ \log \frac{0.1}{0.1} = 0 & r_n = ? \\ \log \frac{0.01}{0.89} \approx -4.49 & r_n = 1 \end{cases} .$$

As a consequence, the channel LLRs for our received word are $[-4.49, 0.00, -4.49, -4.49, +4.49, +4.49, -4.49, -4.49]^\mathrm{T}$.

Figures 2.2 and 2.3 illustrate one complete iteration of belief propagation decoding for the considered setup. In both figures, the boxes next to the variable and the check nodes hold the LLRs passed between the nodes along the shown edges in the Tanner graph. Their order was chosen identical to the order of the connected edges of each node, such that it is easy to see which LLR is passed along which edge.

Figure 2.2 shows the initial step of belief propagation decoding. The channel LLRs are passed via all connected edges of the variable nodes to the respective check nodes. The check nodes receive these LLRs and perform their update according to Equation (2.51).

The outgoing messages of the check nodes that result from this update are shown on the right of Figure 2.3. They are propagated back to the variable nodes through the quasi random structure of the Tanner graph, as it is can be seen from following the edges connected to the check nodes to the variable nodes. Typically, the message passing decoding is performed several times. Anyway, in this example, the variable nodes perform their final bit decision already after the first decoding iteration by summing up all their incoming LLRs and the respective channel LLR, as it is depicted on the left of Figure 2.3. The decision LLRs of the decoder are $[-1.10, +3.39, -1.10, -4.49, +4.49, +4.49,$

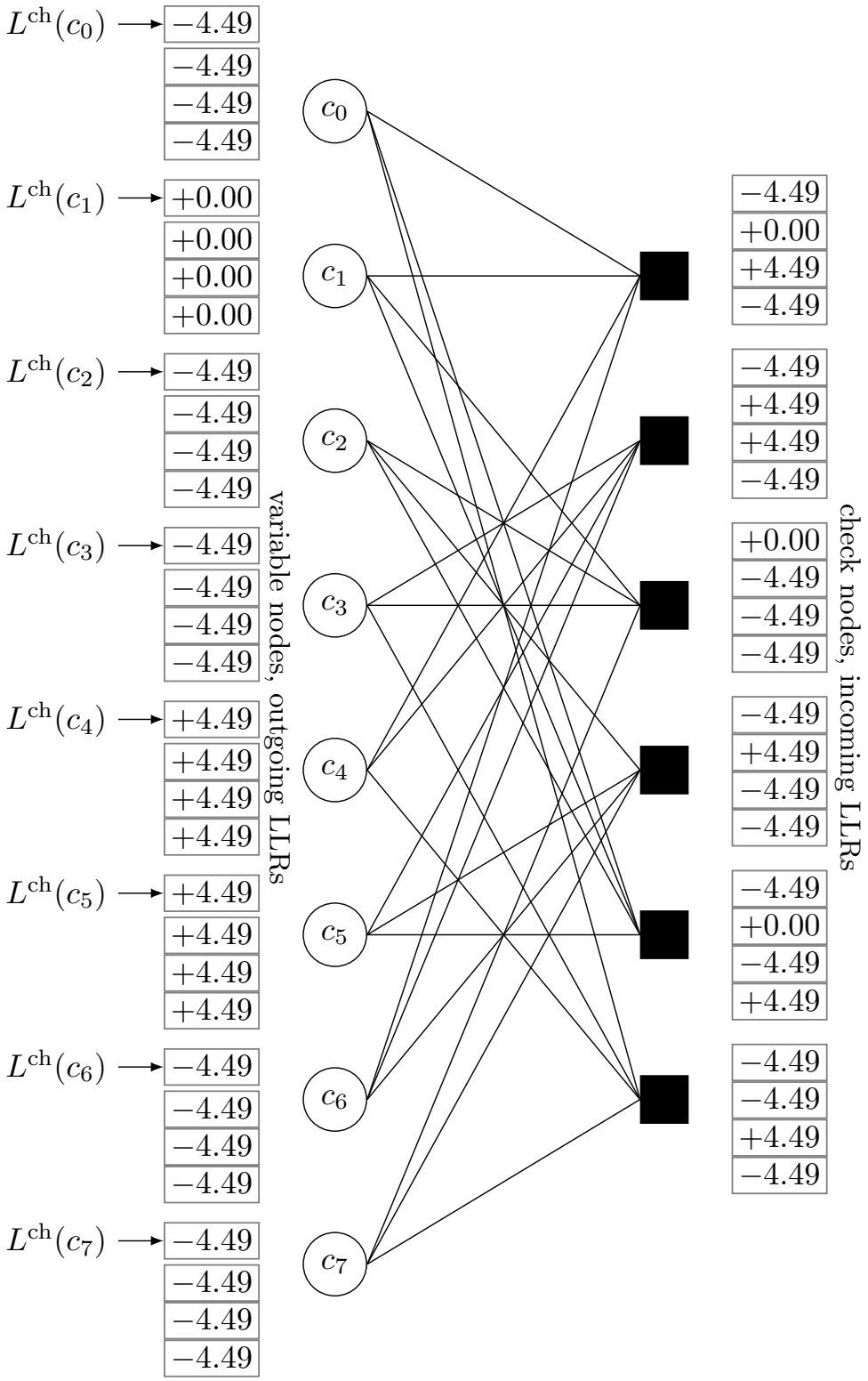


Figure 2.2: Belief propagation on the Tanner graph of $\mathbf{H}^{(\mathrm{a})}$. In the first step, the variable nodes pass their channel LLRs to the check nodes.

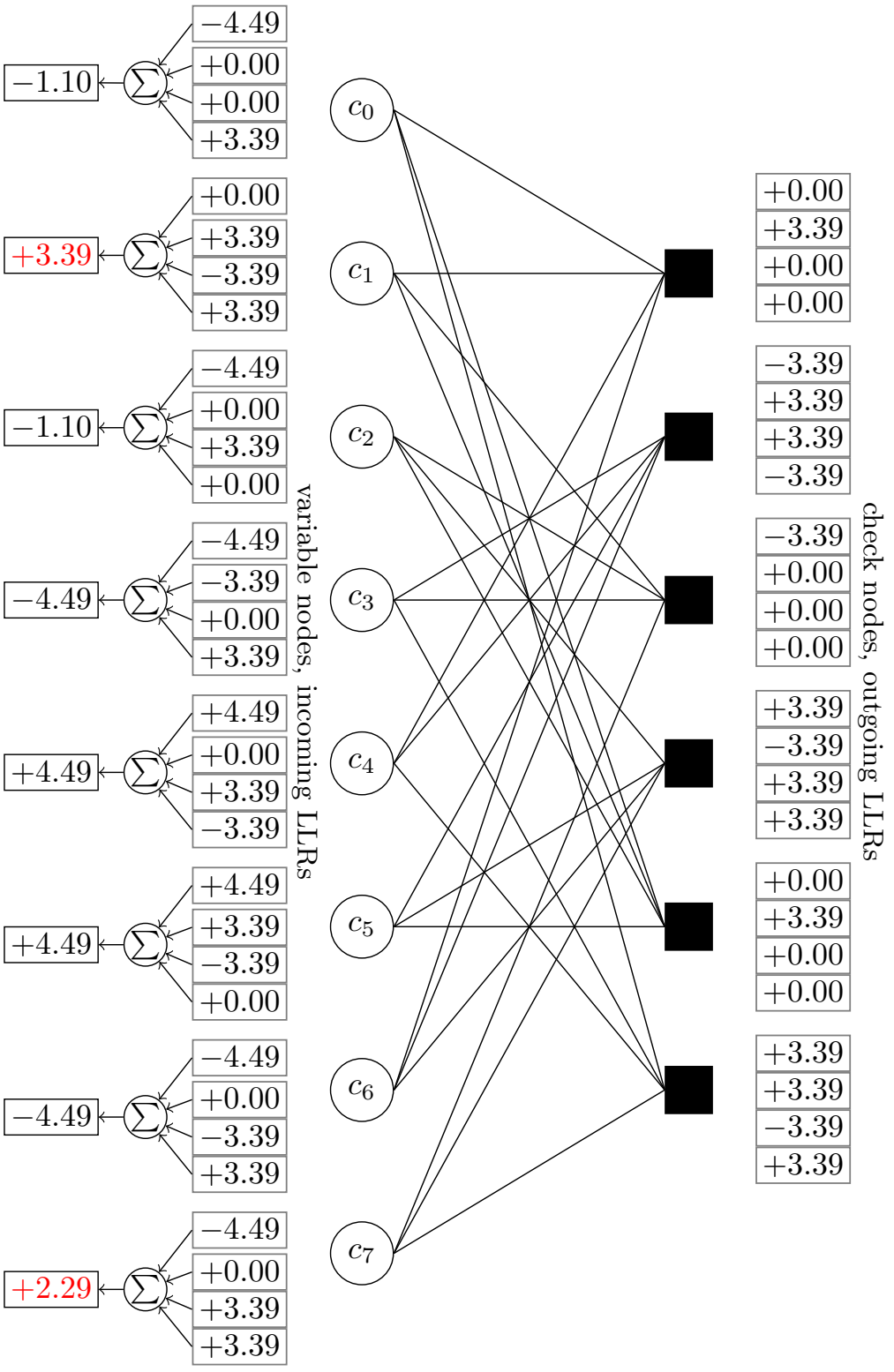


Figure 2.3: After the check nodes have performed their updates, they pass back extrinsic LLRs to the variable nodes.

$-4.49, +2.29]^{\mathrm{T}}$ which correspond to the hard decision $\hat{\mathbf{c}} = [1, 0, 1, 1, 0, 0, 1, 0]^{\mathrm{T}}$. This reveals that the erasure of codeword bit c_1 and the error in codeword bit c_7 were corrected successfully by the described belief propagation decoding. — *End of the example.*

2.3.4 Density Evolution and Threshold Analysis

A fundamental question arising when dealing with LDPC codes is if an LDPC code is capable of guaranteeing an error free transmission for a given channel. A numerical tool to answer this question at least asymptotically, i.e., for LDPC codes with infinite codeword lengths, is called *density evolution* [51, 52]. The general idea of density evolution is to interpret the messages passed inside an LDPC decoder as random variables and to track the evolution of their probability distributions over the decoder iterations. Classical density evolution schemes consider LLR messages. Since LLRs are real numbers and, therefore, continuous the probability distributions tracked in density evolution are actually probability density functions, giving the technique its name. Anyway, in a numerical implementation of density evolution on a computer, the continuous variables have to be approximated by finely discretized discrete random variables [51]. As a result, LLRs are handled as finely discretized discrete random variables in this section, similar as in [51]. This has the side effect of simplyfing the following derivation by eliminating the need for any integrals.

In its classical form, the following vital model simplifications are applied for density evolution [44]:

- LLRs received along distinct edges in the Tanner graph in message passing decoding are independent.

- The transmission channel is memoryless.

- The transmission channel and the node operations in the decoder fulfill certain symmetry conditions, such that it is enough to consider the all-zeros codeword to analyze the error correction performance of an ensemble. These symmetry conditions can be found in [53], but shall not be elaborated here in detail.

Let $L^{(i)}(c_n)$ denote an LLR passed for an arbitrary code bit c_n of an LDPC code in the i-th iteration of belief propagation decoding. Due to the restriction to symmetric channels it is enough to only track the distribution of this LLR for the realization $c_n = 0$, that is, $p\left(L^{(i)}(c_n)|\mathsf{C}_n = 0\right)$. In the following, for a simpler notation, $L^{(i)}(c_n)$ is abbreviated as $L^{(i)}$ because the choice of the respective c_n is irrelevant for the moment. Moreover, the condition $\mathsf{C}_n = 0$ is dropped in the notation $p(L^{(i)}|\mathsf{C}_n = 0)$, yielding just $p\left(L^{(i)}\right)$.

Due to the restriction to the all-zeros codeword, an estimate of the error probability in iteration i can be gained from the probability that $L^{(i)}$ is negative which can be calculated as

$$p_{\mathrm{e}}^{(i)} = \sum_{L^{(i)}<0} p\left(L^{(i)}\right). \tag{2.57}$$

Density evolution is an iterative process which passes probability distributions between a check node processing unit and a variable node processing unit. Let us introduce $L_{\mathrm{v}\to\mathrm{c}}^{(i)}$ which describes the incoming LLRs of the check nodes passed from the variable nodes in iteration i and similarly $L_{\mathrm{c}\to\mathrm{v}}^{(i)}$ which refers to the variable node incoming messages from the check nodes in iteration i. In the initial decoding iteration ($i = 0$), the variable nodes pass their channel LLRs to the check nodes. As a direct consequence $p\left(L_{\mathrm{v}\to\mathrm{c}}^{(0)}\right)$ is identical to $p(L^{\mathrm{ch}})$ in the initial decoding iteration.

At the check nodes, $d_{\mathrm{c}} - 1$ LLRs are processed according to Equation (2.51) to generate extrinsic LLRs. Consider an arbitrary check node with degree d_{c}. In order to determine the distribution $p\left(L_{\mathrm{c}\to\mathrm{v}}^{(0)}\right)$ of the check node to variable node messages, one can leverage that the box-plus sum can be split into recursive box-plus operations of two LLRs. Let us, without loss of generality, consider the generation of the outgoing message for bit $b_{d_{\mathrm{c}}-1}$ which is given by the box-plus sum of $L(b_0), L(b_1), \ldots, L(b_{d_{\mathrm{c}}-2})$ for the considered node. Let

$$x_m = \sum_{n=0}^{m+1} \oplus\, b_n \tag{2.58}$$

for $m \in \{0, 1, \ldots, d_{\mathrm{c}} - 3\}$ denote a partial sum of $m + 2$ bits connected to the considered check node. One first calculates the distribution of the box-plus

sum of the first two incoming LLRs $L(x_0) = L(b_0) \boxplus L(b_1)$. This distribution is given by

$$p(L(x_0)) = \sum_{\substack{L(b_0),L(b_1):\\ L(x_0)=L(b_0)\boxplus L(b_1)}} p(L(b_0))p(L(b_1)). \tag{2.59}$$

The derivation of Equation (2.59) can be found in Appendix A.4. Afterwards, one calculates the distributions of the LLRs $L(x_m)$ for $m \geq 1$, where $x_m = x_{m-1} \oplus b_{m+1}$ as

$$p(L(x_m)) = \sum_{\substack{L(x_{m-1}),L(b_{m+1}):\\ L(x_m)=L(x_{m-1})\boxplus L(b_{m+1})}} p(L(x_{m-1}))p(L(b_{m+1})). \tag{2.60}$$

Finally, $p\left(L\left(x_{d_\mathrm{c}-3}\right)\right)$ equals to the desired distribution of the check node to variable node messages for a check node with degree d_c because according to Equation (2.58), $x_{d_\mathrm{c}-3}$ is the sum of $d_\mathrm{c} - 1$ bits connected to the check node. To finally get the distribution of all incoming variable node messages one has to incorporate the probability that the message received along any edge has been delivered by a degree d_c node which is given by the expectation of all distributions $p\left(L\left(x_{d_\mathrm{c}-3}\right)\right)$ for all d_c over the edges of the Tanner graph. Therefore, the desired distribution is

$$p\left(L_{\mathrm{c}\to\mathrm{v}}^{(0)}\right) = \sum_{d_\mathrm{c}=3}^{d_\mathrm{c}^{\max}} \rho_{d_\mathrm{c}} p(L(x_{d_\mathrm{c}-3})). \tag{2.61}$$

This distribution is passed back to the variable node density evolution.

A degree d_v variable node sums up $d_\mathrm{v} - 1$ LLRs from its connected check nodes and the channel LLR to generate extrinsic information during the iterative decoding process, resulting in a sum of d_v LLRs in total. Let us again consider an arbitrary variable node which belongs to a codeword bit c_n. The node receives incoming LLRs $L_0(c_n), L_1(c_n), \ldots, L_{d_\mathrm{v}-1}(c_n)$ from its connected check nodes and channel LLR $L^{\mathrm{ch}}(c_n)$ from the transmission channel. Again without loss of generality consider generation of the outgoing LLR for the edge which delivers $L_{d_\mathrm{v}-1}(c_n)$ during decoding, such that according to Equation (2.53) this outgoing LLR is given by the sum of $L^{\mathrm{ch}}(c_n)$ and $L_0(c_n), L_1(c_n), \ldots, L_{d_\mathrm{v}-2}(c_n)$. It is well known, that the distribution of the sum of independent random vari-

ables is given by the convolution of the summands' distributions. Hence, one calculates

$$p\left(L_{d_\mathrm{v}}^{\mathrm{ext}}(c_n)\right) = p\left(L^{\mathrm{ch}}(c_n)\right) \otimes p\left(L_0(c_n)\right) \otimes p\left(L_1(c_n)\right) \otimes \cdots \otimes p\left(L_{d_\mathrm{v}-2}(c_n)\right), \quad (2.62)$$

where $f(x) \otimes g(x)$ denotes the convolution of functions $f(x)$ and $g(x)$. In Equation (2.62), all distributions $p\left(L_m(c_n)\right)$ are given by $p\left(L_{\mathrm{c}\to\mathrm{v}}^{(0)}\right)$ which has been calculated by the check node density evolution unit since all incoming LLRs $L_m(c_n)$ are messages from the check nodes. Finally, to obtain the distribution $p\left(L_{\mathrm{v}\to\mathrm{c}}^{(1)}\right)$ which is passed back to the check node density evolution unit, one again has to incorporate the probability that a variable node with degree d_v has delivered an incoming message to a check node. Hence, one obtains

$$p\left(L_{\mathrm{v}\to\mathrm{c}}^{(1)}\right) = \sum_{d_\mathrm{v}=1}^{d_\mathrm{v}^{\max}} \lambda_{d_\mathrm{v}} p\left(L_{d_\mathrm{v}}^{\mathrm{ext}}(c_n)\right). \quad (2.63)$$

The distribution (2.63) is now passed backed to the check node density evolution unit which starts again from Equation (2.59) to determine $p\left(L_{\mathrm{c}\to\mathrm{v}}^{(1)}\right)$ for the next decoding iteration $i = 1$. The described passing of probability distributions between the variable and check node density evolution unit is repeated iteratively, such that $p\left(L_{\mathrm{v}\to\mathrm{c}}^{(i)}\right)$ and $p\left(L_{\mathrm{c}\to\mathrm{v}}^{(i)}\right)$ for all iterations i can be determined.

Example 2.3.4.1 shall illustrate the principle of density evolution exemplarily for the $(d_\mathrm{v}, d_\mathrm{c}) = (3, 6)$ regular LDPC ensemble. The curves presented compare to a similar density evolution example from [44]. The numerical density evolution algorithm was implemented as described in [51].

Example 2.3.4.1. Consider application of a $(d_\mathrm{v}, d_\mathrm{c}) = (3, 6)$ regular LDPC code for data transmission over a symmetric binary input additive white Gaussian noise (AWGN) channel with binary phase shift keying (BPSK) modulation. The symbol duration is T_s, such that the binary LDPC codeword bits c_n are mapped onto bipolar symbols s_k in the symbol clock kT_s according to the rule $s_k = -2c_k + 1$. AWGN with variance $\sigma_{\tilde{n}}^2$ disturbs the received signal. The received samples $\tilde{r}_k$ are given by

$$\tilde{r}_k = s_k + \tilde{n}_k.$$

The conditional distribution $p(\tilde{r}_k|\mathsf{S}_k = \pm 1)$ for the considered channel model is given by a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean $\mu = s_k$ and variance $\sigma^2 = \sigma_{\tilde{n}}^2$, i.e.,

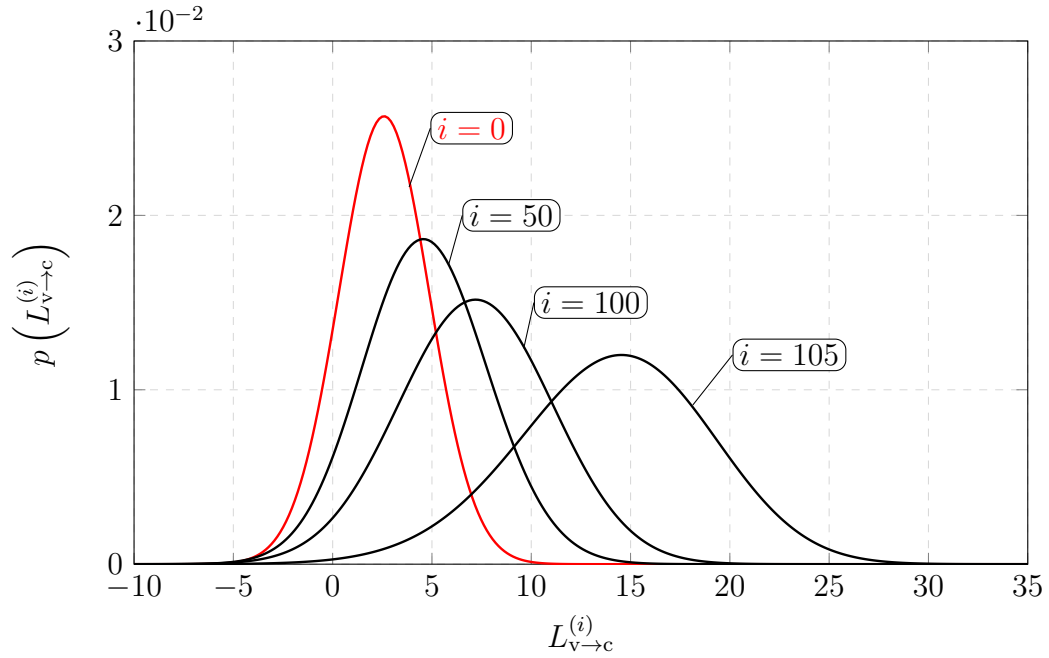
$$p(\tilde{r}_k|\mathsf{S}_k = \pm 1) = \frac{1}{\sqrt{2\pi}\sigma_{\tilde{n}}} \exp\left(-\frac{(\tilde{r}_k \mp 1)^2}{2\sigma_{\tilde{n}}^2}\right).$$

As a result, the channel LLR $L^{\mathrm{ch}}(c_k)$ is given by

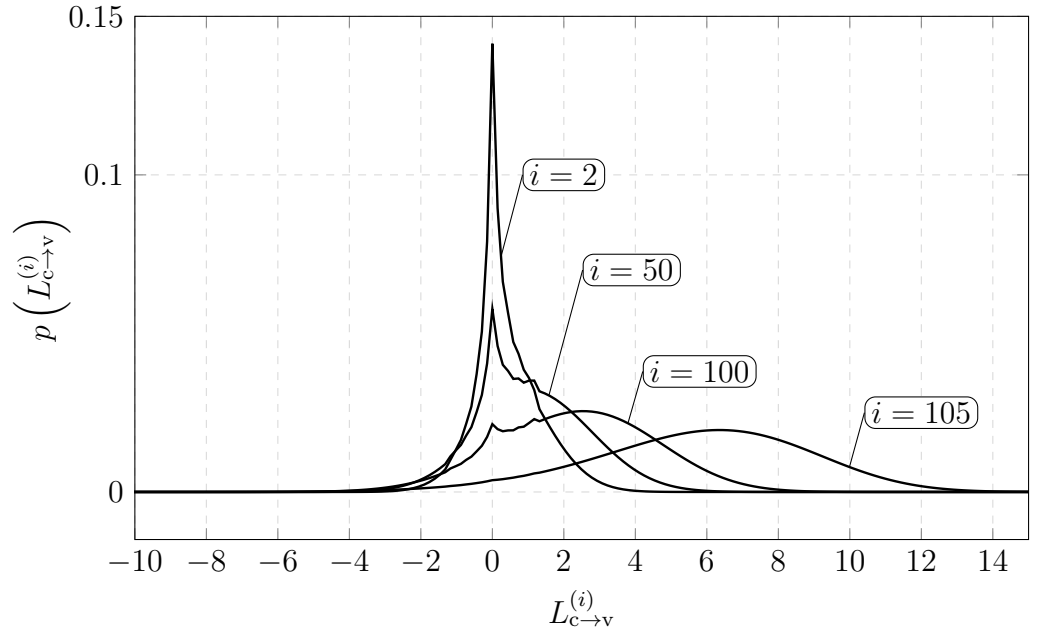
$$L^{\mathrm{ch}}(c_k) = \frac{2}{\sigma_{\tilde{n}}^2}\tilde{r}_k.$$

Hence, one finds that the channel LLR for a given $s_k \in \{-1, +1\}$ also follows a Gaussian distribution $\mathcal{N}(2s_k/\sigma_{\tilde{n}}^2, 4/\sigma_{\tilde{n}}^2)$. Assuming that the all-zeros codeword has been sent and, therefore, $s_k = +1\,\forall k$, yields the initial distribution $p\left(L_{\mathrm{v}\to\mathrm{c}}^{(0)}\right) = \mathcal{N}(2/\sigma_{\tilde{n}}^2, 4/\sigma_{\tilde{n}}^2)$ for the density evolution process. The evolution of the probability distributions of the messages exchanged over the decoder iterations is exemplarily shown in Figure 2.4 for $E_b/N_0 = 1.12$ dB. Figure 2.4a depicts the distributions of the check node incoming messages for several iterations i. Thus, the curve plotted for $i = 0$ is the distribution of the channel LLRs $\mathcal{N}(2/\sigma_{\tilde{n}}^2, 4/\sigma_{\tilde{n}}^2)$ which is plotted in red in Figure 2.4a. Figure 2.4b depicts the distributions of the variable node incoming messages. A qualitative comparison of the curves for the check node incoming messages and the variable node incoming messages tells that the check node incoming messages follow probability distributions with a Gaussian shape in all decoder iterations. This does not hold for the variable node incoming messages. In contrast, the distributions $p\left(L_{\mathrm{c}\to\mathrm{v}}^{(i)}\right)$ for $i \leq 100$ have some peak around $L_{\mathrm{c}\to\mathrm{v}}^{(i)} \approx 0$ which tells that LLRs with small magnitudes have a rather high probability of occurence. Finally, Figure 2.5 shows the estimated asymptotical bit error probability $p_{\mathrm{e}}^{(i)}$ calculated from $p\left(L_{\mathrm{v}\to\mathrm{c}}^{(i)}\right)$ for several E_b/N_0. As it can be seen, for $E_b/N_0 < 1.12$ dB, $p_{\mathrm{e}}^{(i)}$ does not approach zero even for a very large number of decoder iterations. In contrast, the curves for $E_b/N_0 \geq 1.12$ approach $p_{\mathrm{e}}^{(i)} = 0$. —*End of the example.*

The asymptotical error correction behaviour observed in Example 2.3.4.1 is typical for LDPC ensembles. If the channel noise level is above a certain *threshold*, LDPC codes cannot even asymptotically correct all errors, not even with an infinite number of iterations. However, if the channel noise level is



(a) Evolution of the probability distribution of the check node incoming LLRs for the $(3, 6)$ regular LDPC ensemble for $E_b/N_0 = 1.12$ dB.



(b) Evolution of the probability distribution of the variable node incoming LLRs for the $(3, 6)$ regular LDPC ensemble for $E_b/N_0 = 1.12$ dB.

Figure 2.4: Exemplary probability distributions from density evolution.

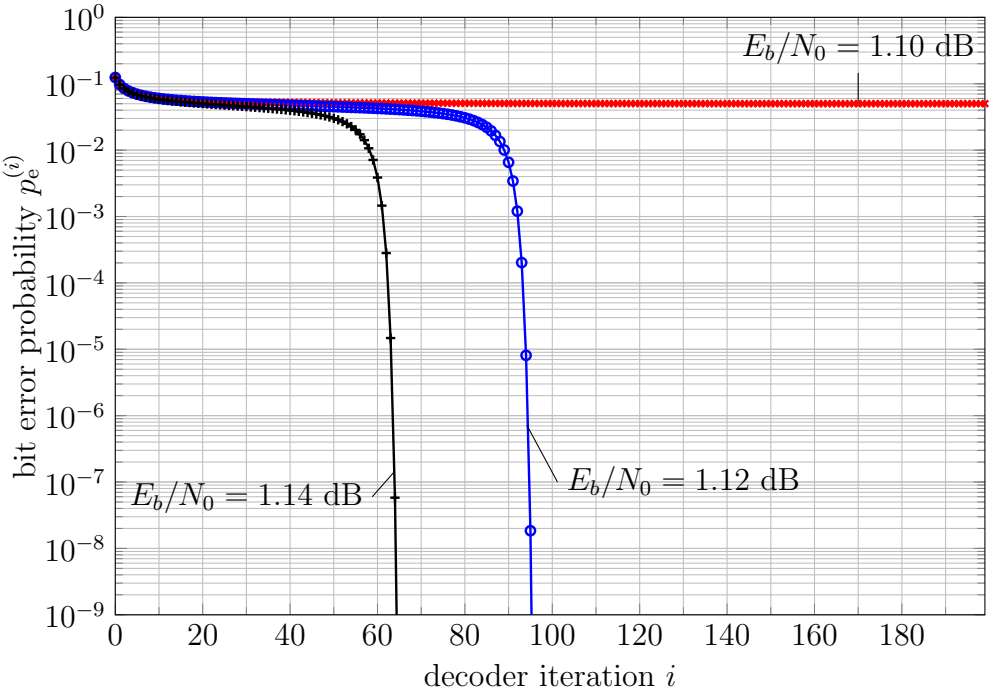


Figure 2.5: Asymptotical bit error probabilities $p_e^{(i)}$ of the $(d_v, d_c) = (3, 6)$ regular LDPC ensemble for transmission over a symmetric AWGN channel with belief propagation decoding for several E_b/N_0.

below this threshold, iterating further and further yields the desired result $p_e^{(i)} \to 0$. For transmission channels which can be characterized using a single parameter, for example, the channel noise variance $\sigma_{\tilde{n}}^2$ or equivalently the standard deviation $\sigma_{\tilde{n}}$, a so called *ensemble threshold* can be defined as the maximum value of this parameter for which a considered LDPC ensemble is asymptotically able to correct all errors. For the AWGN channel from Example 2.3.4.1, the threshold can be defined as [44]

$$\sigma_{\tilde{n}}^{\text{th}} = \sup\left\{\sigma_{\tilde{n}} : \lim_{i\to\infty} p_e^{(i)}(\sigma_{\tilde{n}}) = 0\right\}. \tag{2.64}$$

Finding an ensemble threshold is illustrated in the next example.

Example 2.3.4.2. For the coding and modulation scheme applied in Example 2.3.4.1, the standard deviation of the channel noise can be calculated from E_b/N_0 using the relation

$$\sigma_{\tilde{n}}^2 = \frac{1}{2E_b/N_0 R} \Rightarrow \sigma_{\tilde{n}} = \frac{1}{\sqrt{2E_b/N_0 R}}.$$

Please note, that E_b/N_0 has to be used in linear scale. Example 2.3.4.1 shows that the decoding threshold of the $(3, 6)$ regular ensemble lies between $E_b/N_0 = 1.10$ dB and $E_b/N_0 = 1.12$ dB. These E_b/N_0 correspond to $\sigma_{\tilde{n}} \approx 0.8810$ and 0.8790, respectively. Hence, the decoding threshold $\sigma_{\tilde{n}}^{\text{th}}$ of the $(3, 6)$ regular LDPC ensemble for the considered channel is approximately $\sigma_{\tilde{n}}^{\text{th}} \approx 0.88$. — *End of the example.*

Density evolution can be used to make forecasts on the error correction performance of arbitrary LDPC ensembles under message passing decoding. Hence, it can also be used to analyze and optimize irregular LDPC ensembles. A typical aim of this density evolution optimization is finding code ensembles with large threshold values $\sigma_{\tilde{n}}$ or, equivalently, low threshold-E_b/N_0. This aim originally motivated the application of density evolution in the context of LDPC codes [53]. Nonetheless, in this thesis, density evolution will also play an important role in the design of novel LDPC decoders.

2.4 Factor Graphs and the Sum-Product Algorithm

This section introduces a very powerful framework for modelling communication and other systems. The considered framework is called factor graphs. Factor graphs have gained huge interest in the communications community because they allow for a very intuitive hierarchical modelling of communication systems. Moreover, a generic and efficient algorithm exists for factor graphs of joint probability distributions which allows to efficiently calculate marginal probability distributions. This algorithm is called the sum-product algorithm and applicable, for example, when a posteriori estimates of unknown quantities shall be obtained. In fact, a variant of the sum-product algorithm has already been introduced in the preceding section for decoding of LDPC codes. It turns out that the belief propagation decoding algorithm from Section 2.3.3 is a log-domain variant of the sum-product algorithm. A huge variety of literature on the topic of factor graphs exists and, therefore, in the following only the key concepts shall be addressed. The introduction provided is mainly based on [54–56]. More detailed studies of factor graphs can be found there. Factor

graphs will be used to introduce the concept of Information Bottleneck graphs in Chapter 4.

2.4.1 A Graphical Model for the Factorization of Functions

Factor graphs provide a way of visually expressing how a function of several variables can be factorized into the product of factors which might only depend on a subset of the arguments of the original function. It is worth mentioning, that several styles of factor graphs exist [56]. However, in this thesis the notation from [54] will be used for representation of factor graphs. In this notation factor graphs consist of variable nodes, factor nodes and edges connecting both types of nodes.

In the theory of factor graphs the factorization structure of a so called *global* function f into several so called *local* factors is considered. Each local factor in the factor graph is drawn as a square node and each argument of the global function is represented by a variable node. The latter are typically drawn as circle nodes. Edges between variable nodes and factor nodes are placed only if a variable is an argument of a function with the respective factor node. The best way to further explain the concept of factor graphs is switching over to the following example.

Example 2.4.1.1. Consider a function $f(x_0, x_1, x_2, x_3, x_4)$ of five different arguments x_0, x_1, x_2, x_3, x_4 and assume that the function can be written as

$$f(x_0, x_1, x_2, x_3, x_4) = f_0(x_0, x_1) f_1(x_1, x_2, x_3) f_2(x_0, x_4).$$

Hence, the function $f(x_0, x_1, x_2, x_3, x_4)$ can be factorized into three independent factors which all only depend on a subset of the arguments x_0, x_1, x_2, x_3, x_4 of the global function $f(x_0, x_1, x_2, x_3, x_4)$. Figure 2.6 shows a factor graph of $f(x_0, x_1, x_2, x_3, x_4)$ according to the introduced factorization structure, where factor nodes are drawn as square nodes and variable nodes are drawn as circles, just as it has been explained. Obviously, the edges appearing in the factor graph express the *variable is an argument of local function* relation of all variable and factor nodes present in the factor graph. —*End of the example.*

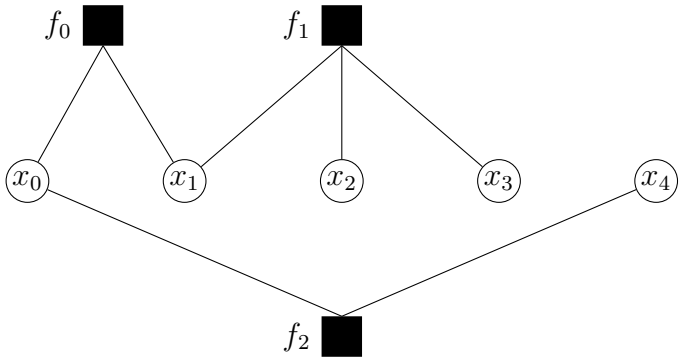


Figure 2.6: Exemplary factor graph of function $f(x_0, x_1, x_2, x_3, x_4) = f_0(x_0, x_1)f_1(x_1, x_2, x_3)f_2(x_0, x_4)$.

2.4.2 Factor Graphs of Probability Distributions

Factor graphs are particularly useful to describe the factorization structure of probability distributions because in a factor graph of a probability distribution, edges directly illustrate dependencies of all the included variables. This again shall be illustrated using an example.

Example 2.4.2.1. Consider a joint probability distribution $p(x_0, x_1, x_2, x_3)$. Every such distribution can be factorized as

$$p(x_0, x_1, x_2, x_3) = p(x_3|x_0, x_1, x_2)p(x_2|x_0, x_1)p(x_1|x_0)p(x_0).$$

Now consider some dependencies and independencies of the random variables involved, for example, independent X_0 and X_1 together with the Markov chain relation $(\mathsf{X}_0, \mathsf{X}_1) \to \mathsf{X}_2 \to \mathsf{X}_3$. In this setup, the factorization of $p(x_0, x_1, x_2, x_3)$ becomes

$$p(x_0, x_1, x_2, x_3) = p(x_3|x_2)p(x_2|x_0, x_1)p(x_1)p(x_0),$$

that is, some of the dependencies between the distinct variables disappear. Figure 2.7 shows a corresponding factor graph. The fact that X_2 directly depends on X_0 and X_1 and that X_3 directly depends on X_2 is clearly visible in this factor graph. —*End of the example.*

A particularly useful feature of factor graphs of joint probability distributions is that they allow for *hierarchical* modelling by opening and closing factor nodes [37]. The process of closing factor nodes corresponds to gathering several

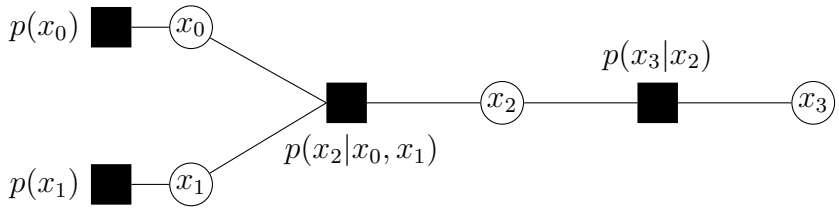


Figure 2.7: Exemplary factor graph of a joint probability distribution $p(x_0, x_1, x_2, x_3)$ which factorizes as $p(x_0, x_1, x_2, x_3) = p(x_3|x_2)p(x_2|x_0, x_1)p(x_1)p(x_0)$.

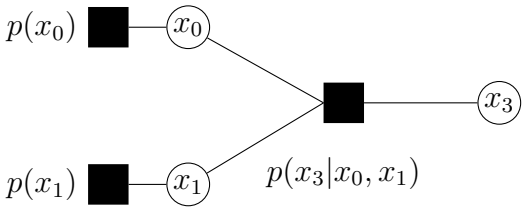
local factors and eventually some of their connected variable nodes in a superordinate factor node. In contrast, the process of opening nodes corresponds to a further factorization of factor nodes and eventually includes the introduction of additional variables. The foundations for opening and closing nodes in a factor graph of a joint probability distribution are that the product of several factors can be interpreted as a single factor node and that variables can simply be eliminated using the sum operation. The following example illustrates the principles of closing and opening nodes.

Example 2.4.2.2. Consider $p(x_0, x_1, x_2, x_3)$ with the factor graph shown in Figure 2.7 again. Given this factor graph one wants to obtain the factor graph of the marginal distribution

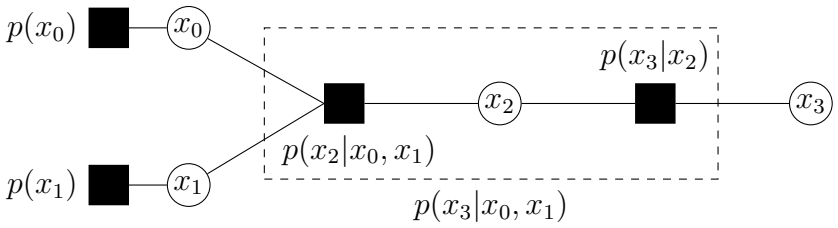
$$p(x_0, x_1, x_3) = \sum_{x_2 \in \mathcal{X}_2} p(x_0, x_1, x_2, x_3) = p(x_0)p(x_1) \underbrace{\sum_{x_2 \in \mathcal{X}_2} p(x_3|x_2)p(x_2|x_0, x_1)}_{p(x_3|x_0, x_1)} .$$

The resulting factor graph is shown in Figure 2.8a. In this illustration x_2 and factors $p(x_3|x_2)$ and $p(x_2|x_0, x_1)$ are gone. These factors are gathered in $p(x_3|x_0, x_1)$ in the novel factor graph and x_2 has been eliminated by the sum operation.

The factor node $p(x_3|x_0, x_1)$ can be opened again because its internal structure is known from the factor graph of $p(x_0, x_1, x_2, x_3)$. The factor graph of $p(x_0, x_1, x_3)$ with an opened node $p(x_3|x_0, x_1)$ is shown in Figure 2.8b. Please note that this factor graph is equivalent to the one shown in Figure 2.7. The



(a) Exemplary factor graph of a joint probability distribution $p(x_0, x_1, x_3)$ with a closed node $p(x_3|x_0, x_1)$.



(b) Exemplary factor graph of a joint probability distribution $p(x_0, x_1, x_3)$ with opened node $p(x_3|x_0, x_1)$.

Figure 2.8: Factor graphs of $p(x_0, x_1, x_3)$.

only difference is the dashed box labeled by $p(x_3|x_0, x_1)$ which shows the internal structure of

$$p(x_3|x_0, x_1) = \sum_{x_2 \in \mathcal{X}_2} p(x_3|x_2)p(x_2|x_0, x_1).$$

—*End of the example.*

The principles of opening and closing nodes in factor graphs of joint probability distributions are generic. The rule for closing a node simply is drawing a box around all factors to be gathered in the closed node. Afterwards one has to marginalize over all the variables inside this box to eliminate the respective variables. Opening a node describes the inverse process of considering the internal structure of a closed node. Opening nodes allows to switch from a higher level model to a more detailed view on the internal structure of a system described by a factor graph. Closing nodes, in contrast, allows to take a factor graph model to a higher level of abstraction which might be more convenient for a large scale analysis. This ability of hierarchical modelling will be adapted in the concept of Information Bottleneck graphs in Chapter 4.

2.4.3 The Sum-Product Algorithm

A generic algorithm exists which allows to efficiently calculate all marginal distributions from the factor graph of a joint probability distribution for factor graphs which do not have cycles. This algorithm is based on message passing between variable nodes and factor nodes in the factor graph. Moreover, updates are carried out at the respective nodes which only follow two simple and generic rules. In the following, first these generic rules are introduced. Afterwards, an example will be provided.

Consider an arbitrary cycle free factor graph of a joint probability distribution $p(x_0, x_1, \ldots, x_{N-1})$ which factorizes as

$$p(x_0, x_1, \ldots, x_{N-1}) = \prod_{l=0}^{L-1} f_l(X_l), \tag{2.65}$$

where X_l denotes the subset of all arguments $x_n \in X_l$ of factor node f_l, such that f_l is a function having all variables in the set X_l as arguments. In the sum-product algorithm, messages μ are passed on the factor graph of $p(x_0, x_1, \ldots, x_{N-1})$. These messages μ are functions of the variables involved. A message from variable node x_n to its connected factor node f_l is denoted $\mu_{x_n \to f_l}(x_n)$. Similarly, a message from factor node f_l to variable node x_n is denoted $\mu_{f_l \to x_n}(x_n)$. The messages passed are obtained by sequential updates at the variable and the factor nodes. Let $N(x_n)$ denote the set of neighboring factor nodes connected to a variable node x_n by an edge. With this notation the update rule of variable node x_n is

$$\mu_{x_n \to f_l}(x_n) = \prod_{g \in N(x_n) \setminus \{f_l\}} \mu_{g \to x_n}(x_n). \tag{2.66}$$

That is, the outgoing message of a variable node x_n to factor node f_l is given by the product of all incoming messages except the one which arrived at the variable node along the target edge connecting the variable node to factor node f_l. The update rule of factor node f_l is

$$\mu_{f_l \to x_n}(x_n) = \sum_{\sim\{x_n\}} \left(f_l(X_l) \prod_{x_m \in X_l \setminus \{x_n\}} \mu_{x_m \to f_l}(x_m) \right), \tag{2.67}$$

where $\sum_{\sim\{x_n\}}$ denotes the sum over all variables connected to the factor node, but x_n. As a result, the outgoing message of factor node f_l for variable node

x_n is generated by first taking the product of all incoming messages of the factor node except the one that reached the factor node along the target edge connected to variable node x_n. This product has to be multiplied by the local function $f_l(X_l)$ of the factor node. Finally, one has to sum the resulting product over all variables connected to the factor node, except the one connected to the factor node by the target edge. The sum-product algorithm is insensitive to scaling the messages exchanged with a constant factor, as its purpose is to determine a posteriori probability distributions which have to fulfil the law of total probability. Therefore, it is common to normalize the messages exchanged, such that each message sums up to 1.

The sum-product algorithm starts at the leaf nodes of a cycle free factor graph. In the initial step, all factor nodes appearing as leaf nodes simply pass their local functions along their only connected target edge. If message passing starts at a variable node x_n which is a leaf node, it passes a uniform distribution over the event space of X_n in the initial step of message passing. Typically, message passing is continued until two messages in both directions have been passed along all edges in the factor graph. The product of all messages arriving at a variable node x_n along any connected edge then is proportional to the marginal distribution $p(x_n)$.

The application of the sum-product algorithm shall be clarified in the following example.

Example 2.4.3.1. Consider a binary input binary output memoryless channel with two consecutive input bits B_0 and B_1 and their corresponding output bits Y_0 and Y_1. Both input bits $\mathsf{B}_0, \mathsf{B}_1$ and both output bits $\mathsf{Y}_0, \mathsf{Y}_1$ take values 0 and 1 with equal probability. The probability that a bit is received incorrectly is $\Pr(\mathsf{B}_k \neq \mathsf{Y}_k) = 0.1$. Based on the observation of realizations y_0 and y_1, one aims to generate a belief on a bit x which is given by the sum of b_0 and b_1, that is, $x = b_0 \oplus b_1$. Moreover, one aims to determine all other posterior distributions $p(b_0|y_0, y_1)$ and $p(b_1|y_0, y_1)$.

Figure 2.9a shows the factor graph of $p(x, b_0, b_1, y_0, y_1)$. The factor $p(x|b_0, b_1)$ in this factor graph reflects fact that $x = b_0 \oplus b_1$. Hence, it takes value 1 if $x = b_0 \oplus b_1$ and 0 else. This deterministic connection can be described using the Kronecker delta function, such that

$$p(x|b_0, b_1) = \delta(x \oplus b_0 \oplus b_1).$$

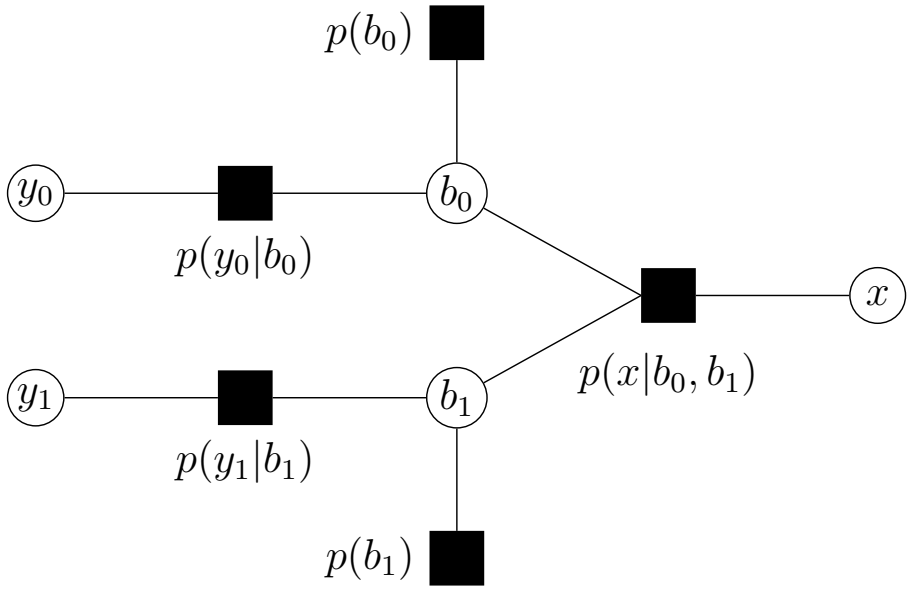
Let the received values be $(y_0, y_1) = (0, 1)$ and let us describe the messages μ passed in the sum product algorithm as vectors, such that

$$\mu_{f_l \to b_n}(b_n) = \begin{bmatrix} \mu_{f_l \to b_n}(0) \\ \mu_{f_l \to b_n}(1) \end{bmatrix}, \quad \mu_{b_n \to f_l}(b_n) = \begin{bmatrix} \mu_{b_n \to f_l}(0) \\ \mu_{b_n \to f_l}(1) \end{bmatrix}.$$

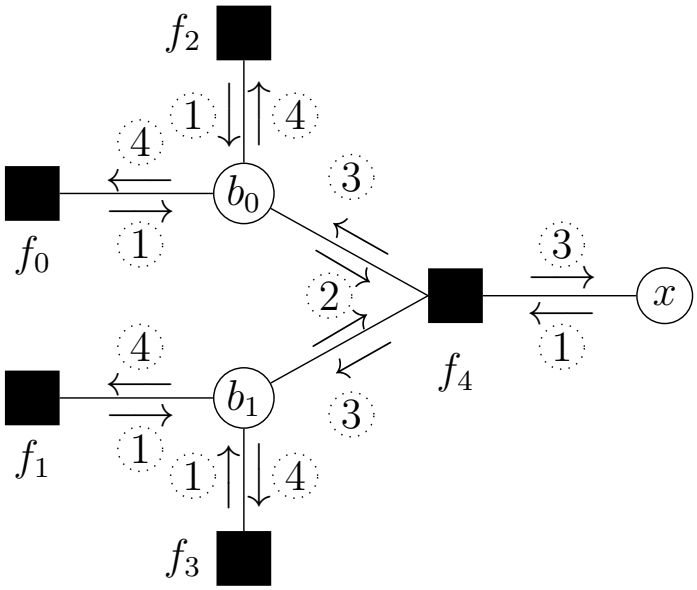
The consecutive steps of the sum-product algorithm on the factor graph of $p(x, b_0, b_1, y_0, y_1)$ to determine $p(x|y_0, y_1)$, $p(b_0|y_0, y_1)$ and $p(b_1|y_0, y_1)$ are enumerated and labeled using dotted circles in Figure 2.9b. Arrows indicate the messages passed in a certain step. First, it is important that since y_0 and y_1 are known, they become so called *evidence* which disappears from the factor graph.

The message passing process begins with step 1 at the leaf nodes of the resulting factor graph in Figure 2.9b. The messages $\mu_{f_0 \to b_0}(b_0)$, $\mu_{f_1 \to b_1}(b_1)$, $\mu_{f_3 \to b_1}(b_1)$, $\mu_{f_2 \to b_0}(b_0)$ and $\mu_{x \to f_4}(x)$ are passed in step 1. According to the rules explained, the local factors f_0, f_1, f_2 and f_3 pass their local functions and variable node x passes a uniform distribution over the two possible bit values $x \in \{0, 1\}$. Hence,

$$\begin{aligned}
\mu_{f_0 \to b_0}(b_0) &= \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}, \\
\mu_{f_1 \to b_1}(b_1) &= \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}, \\
\mu_{f_2 \to b_0}(b_0) &= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \\
\mu_{f_3 \to b_1}(b_1) &= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \\
\mu_{x \to f_4}(x) &= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}.
\end{aligned}$$



(a) Factor graph of $p(x, b_0, b_1, y_0, y_1)$. The deterministic connection $x = b_0 \oplus b_1$ is expressed by factor $p(x|b_0, b_1)$.



(b) Illustration of message passing on the factor graph of $p(x, b_0, b_1, y_0, y_1)$ for given evidence y_0, y_1. The numbers in the dotted circles count the consecutive steps of the message passing algorithm. Arrows indicate the messages passed in the respective steps.

Figure 2.9: Illustration of message passing in the sum-product algorithm.

In step 2 the variable nodes b_0 and b_1 are able to generate the messages $\mu_{b_0 \to f_4}(b_0)$ and $\mu_{b_1 \to f_4}(b_1)$. These messages are given by the products

$$\begin{aligned} \mu_{b_0 \to f_4}(b_0) &= \mu_{f_0 \to b_0}(b_0)\mu_{f_2 \to b_0}(b_0) = \begin{bmatrix} 0.45 \\ 0.05 \end{bmatrix} \propto \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix} \\ \mu_{b_1 \to f_4}(b_1) &= \mu_{f_1 \to b_1}(b_1)\mu_{f_3 \to b_1}(b_1) = \begin{bmatrix} 0.05 \\ 0.45 \end{bmatrix} \propto \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}. \end{aligned}$$

In the following step 3, the factor node f_4 can perform its updates because it has received messages along all its connected edges. According to the sum-product update rule for factor nodes its output messages are given by

$$\begin{aligned} \mu_{f_4 \to x}(x) &= \sum_{\sim\{x\}} \delta(x \oplus b_0 \oplus b_1)\mu_{b_0 \to f_4}(b_0)\mu_{b_1 \to f_4}(b_1), \\ \mu_{f_4 \to b_0}(b_0) &= \sum_{\sim\{b_0\}} \delta(x \oplus b_0 \oplus b_1)\mu_{b_1 \to f_4}(b_1)\mu_{x \to f_4}(x), \\ \mu_{f_4 \to b_1}(b_1) &= \sum_{\sim\{b_1\}} \delta(x \oplus b_0 \oplus b_1)\mu_{b_0 \to f_4}(b_0)\mu_{x \to f_4}(x). \end{aligned}$$

The message $\mu_{f_4 \to x}(x)$ can be resolved further as

$$\mu_{f_4 \to x}(x) = \sum_{\substack{(b_0,b_1): \\ b_0 \oplus b_1 = x}} \mu_{b_0 \to f_4}(b_0)\mu_{b_1 \to f_4}(b_1), \tag{2.68}$$

such that for $x = 0$ one has to sum over all combinations (b_0, b_1) with $b_0 \oplus b_1 = 0$ and, likewise for $x = 1$ over all combinations (b_0, b_1) with $b_0 \oplus b_1 = 1$. The reason is that the Kronecker delta inside the sum term is 0 if $x \neq b_0 \oplus b_1$ and 1 otherwise. Due to the fact that $\mu_{x \to f_4}(x) = 1/2 \, \forall x$, the other two messages are given by

$$\mu_{f_4 \to b_0}(b_0) = \frac{1}{2} \sum_{\substack{(x,b_1): \\ (x \oplus b_1) = b_0}} \mu_{b_1 \to f_4}(b_1), \tag{2.69}$$

$$\mu_{f_4 \to b_1}(b_1) = \frac{1}{2} \sum_{\substack{(x,b_0): \\ (x \oplus b_0) = b_1}} \mu_{b_0 \to f_4}(b_0). \tag{2.70}$$

This yields

$$\begin{aligned}
\mu_{f_4\to x}(x) &= \begin{bmatrix} 0.9\cdot 0.1 + 0.1\cdot 0.9 \\ 0.9\cdot 0.9 + 0.1\cdot 0.1 \end{bmatrix} = \begin{bmatrix} 0.18 \\ 0.82 \end{bmatrix} \\
\mu_{f_4\to b_0}(b_0) &= \begin{bmatrix} 0.5\cdot (0.1+0.9) \\ 0.5\cdot (0.9+0.1) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \\
\mu_{f_4\to b_1}(b_1) &= \begin{bmatrix} 0.5\cdot (0.9+0.1) \\ 0.5\cdot (0.1+0.9) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}.
\end{aligned}$$

In the final step 4, the messages $\mu_{b_0\to f_2}(b_0)$, $\mu_{b_1\to f_3}(b_1)$, $\mu_{b_0\to f_0}(b_0)$ and $\mu_{b_1\to f_1}(b_1)$ are calculated and passed. These messages are given by

$$\begin{aligned}
\mu_{b_0\to f_2}(b_0) &= \mu_{f_4\to b_0}(b_0)\mu_{f_0\to b_0}(b_0), \\
\mu_{b_1\to f_3}(b_1) &= \mu_{f_4\to b_1}(b_1)\mu_{f_1\to b_1}(b_1), \\
\mu_{b_0\to f_0}(b_0) &= \mu_{f_4\to b_0}(b_0)\mu_{f_2\to b_0}(b_0), \\
\mu_{b_1\to f_1}(b_1) &= \mu_{f_4\to b_1}(b_1)\mu_{f_3\to b_1}(b_1)
\end{aligned}$$

which delivers

$$\begin{aligned}
\mu_{b_0\to f_2}(b_0) &= \begin{bmatrix} 0.5\cdot 0.9 \\ 0.5\cdot 0.1 \end{bmatrix} \propto \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix} \\
\mu_{b_1\to f_3}(b_1) &= \begin{bmatrix} 0.5\cdot 0.1 \\ 0.5\cdot 0.9 \end{bmatrix} \propto \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix} \\
\mu_{b_0\to f_0}(b_0) &= \begin{bmatrix} 0.5\cdot 0.5 \\ 0.5\cdot 0.5 \end{bmatrix} \propto \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \\
\mu_{b_1\to f_1}(b_1) &= \begin{bmatrix} 0.5\cdot 0.5 \\ 0.5\cdot 0.5 \end{bmatrix} \propto \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}.
\end{aligned}$$

Finally, all posterior distributions $p(x|y_0, y_1), p(b_0|y_0, y_1)$ and $p(b_1|y_0, y_1)$ are obtained by multiplication of all incoming messages of the respective variable

nodes and an additional normalization step. In this example, the sum-product algorithms yields

$$\begin{aligned} p(x|y_0, y_1) &= \begin{bmatrix} 0.18 \\ 0.82 \end{bmatrix} \\ p(b_0|y_0, y_1) &= \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix} \\ p(b_1|y_0, y_1) &= \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}. \end{aligned}$$

—*End of the example.*

The sum-product algorithm is the origin of many signal processing algorithms, for example, the belief propagation decoding algorithm for decoding of LDPC codes presented in Section 2.3.3. The takeaway message of is this section is that the sum-product algorithm allows to efficiently calculate all marginal distributions of a given joint probability distribution by message passing on a factor graph according to generic rules which are applicable for every cycle free factor graph.

Finally, it is appropriate to add a note on the application of the sum-product algorithm on factor graphs with cycles which appear in many practical scenarios. Interestingly, a straightforward application of the sum-product algorithm yields useful results also on factor graphs with cycles in many applications, despite in theory, the algorithm only obtains exact marginal probability distributions on cycle free graphs [54]. In this case, the message passing process typically is executed repeatedly, resulting in an iterative message passing algorithm. The iterative decoding of LDPC codes is a famous example for the application of the sum-product algorithm on a graph with cycles.

Using Example 2.4.3.1, the fact that belief propagation decoding is an instance of the sum-product algorithm in the log-domain can quickly be checked.

Example 2.4.3.2. Obviously, an extrinsic LLR for x in Example 2.4.3.1 is given by

$$L^{\text{ext}}(x) = L(b_0) \boxplus L(b_1) \approx -1.5163.$$

Using the relations

$$\begin{aligned} \Pr(\mathsf{X}=0) &= \frac{\exp\left(L^{\text{ext}}(x)\right)}{1+\exp\left(L^{\text{ext}}(x)\right)} = \frac{\exp(-1.5163)}{1+\exp(-1.5163)} \approx 0.18 \\ \Pr(\mathsf{X}=1) &= \frac{1}{1+\exp\left(L^{\text{ext}}(x)\right)} = \frac{1}{1+\exp(-1.5163)} \approx 0.82, \end{aligned}$$

from [36], one directly obtains $\Pr(\mathsf{X}=0) = 0.18$ and $\Pr(\mathsf{X}=1) = 0.82$, just as it was indicated by $p(x|y_0, y_1)$ from Example 2.4.3.1. —*End of the example.*

Chapter 3

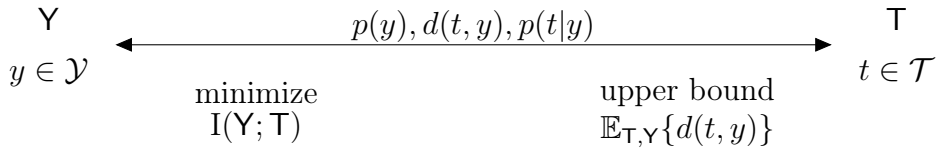
The Information Bottleneck Method

The Information Bottleneck method is an information theoretical framework which is related to the famous rate-distortion theory [57]. Rate-distortion theory considers lossy compression of source data. Its primary aim is to keep the distortion which results from a compression below a certain threshold. In [3] Tishby *et al.* argue that a major weakness of rate-distortion theory is its inherent need to define a distortion measure in advance. The choice of a proper distortion measure which shall be minimized under compression of the original source data is application specific and no general rule for choosing a good distortion measure can be provided. Motivated by this preliminary observation the fundamental idea presented in [3] was to consider a variable of interest which describes which features in the original source data are relevant and shall be preserved under compression. Doing so, one can design a lossy compression scheme which aims to preserve *relevant information* on this variable. In this chapter, the reader is first guided through the theoretical ideas of rate-distortion theory and the Information Bottleneck method. The relationship and the key differences between rate-distortion theory and the Information Bottleneck method are introduced and discussed. Afterwards, self consistent equations are presented which describe the mathematical structure of Information Bottleneck problems. Finally, several Information Bottleneck algorithms to design relevant information preserving compression mappings are provided.

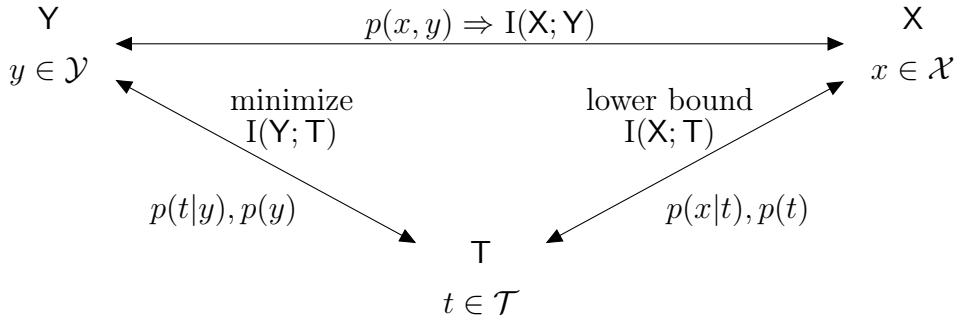
3.1 Rate-distortion and Information Bottleneck Theory

Rate-distortion theory deals with the problem of compressing an observed random variable Y to a more compact variable T with a certain amount of distortion. Typically, this distortion shall not exceed a certain limit, such that the compressed random variable still is a good approximation of the observation. This problem is compactly depicted in Figure 3.1a and discussed in the following.

The aim of rate-distortion theory is to map Y onto T such that T is a compressed variant of Y and random variables $\mathsf{Y} \to \mathsf{T}$ form a Markov chain. Formally, the connection between Y and T can be described by the conditional probability distribution $p(t|y)$. This conditional distribution can be considered



(a) Rate-distortion theory problem setup. The observed random variable Y and the compression variable T are considered. The aim is determination of an optimum compression mapping $p(t|y)$ such that $\mathrm{I}(\mathsf{Y};\mathsf{T})$ is minimized while the expected distortion $\mathbb{E}_{\mathsf{T},\mathsf{Y}}\{d(t,y)\}$ stays below a certain threshold D.



(b) Information Bottleneck problem setup. The observed random variable Y and relevant random variable X share mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$. The aim is determination of an optimum compression mapping $p(t|y)$ such that $\mathrm{I}(\mathsf{T};\mathsf{Y})$ is minimized while $\mathrm{I}(\mathsf{X};\mathsf{T})$ is preserved.

Figure 3.1: Visualization of the Information Bottleneck and the rate-distortion theory problem setups.

as a probabilistic mapping of realization y onto realization t. By determining $p(t|y)$ such that $\mathrm{I}(\mathsf{Y};\mathsf{T}) \rightarrow \min$, one can address the desired compression goal on the one hand. To get an intuition about the fact that minimization of $\mathrm{I}(\mathsf{Y};\mathsf{T})$ in fact results in a compression one can consider two extreme cases. First, consider a $p(t|y)$ which maps all $y \in \mathcal{Y}$ onto only one $t \in \mathcal{T}$ deterministically. Since in this case t is independent of y, it follows directly that $\mathrm{I}(\mathsf{Y};\mathsf{T})$ takes its smallest possible value $\mathrm{I}(\mathsf{Y};\mathsf{T}) = 0$. Now consider the case where $p(t|y)$ simply is an identity transform which does not imply any lossy compression as it just maps y onto t trivially. In this case, the mutual information $\mathrm{I}(\mathsf{Y};\mathsf{T})$ takes its largest possible value $\mathrm{H}(\mathsf{Y}) = \mathrm{H}(\mathsf{T})$.

On the other hand, to introduce a bound on the distortion, one first needs to quantify the distortion that mapping a realization y onto t will introduce. Therefore, a distortion function $d(t, y)$ has to be defined which assigns the distortion $d(t, y)$ to each possible pair (t, y). This function $d(t, y)$ has to be a non-negative, real valued function. The idea here is that $d(t, y)$ takes small values, if t is a good approximation for y and large values otherwise, such that it quantifies the distortion of mapping y onto t adequately. To perform a compression and to simultaneously impose an upper bound on the expected distortion, one ends up with a side constrained optimization problem for finding $p(t|y)$ which reads

$$\min_{p(t|y)} \mathrm{I}(\mathsf{Y};\mathsf{T}), \quad \text{such that } \mathbb{E}_{\mathsf{T},\mathsf{Y}}\{d(t,y)\} \leq D \tag{3.1}$$

for some expected distortion threshold $D > 0$. The side constrained optimization problem (3.1) can be solved using the method of Lagrangian multipliers. For the problem at hand, application of this method yields an iterative algorithm to determine $p(t|y)$ which is called the Blahut-Arimoto algorithm [57]. The Blahut-Arimoto algorithm is able to solve a rate-distortion problem for a given probability distribution $p(y)$, a given set of representatives $\mathcal{T}$ and a given distortion measure $d(t, y)$. Since rate-distortion theory is not the main topic of this thesis, this algorithm shall not be derived here. Compact descriptions of the Blahut-Arimoto algorithm are provided in [3, 9].

The problem setup of the Information Bottleneck method is related to rate-distortion theory, but some key differences exist. The method is compactly illustrated in Figure 3.1b and will be explained in the following.

Just as in rate-distortion theory, the general aim of the method is to construct a compression mapping $p(t|y)$ such that T is a compact representation of observation Y. Other than rate-distortion theory, however, the Information Bottleneck method does not only consider two random variables Y, T, but three random variables $\mathsf{X}, \mathsf{Y}, \mathsf{T}$ which form Markov chain $\mathsf{X} \to \mathsf{Y} \to \mathsf{T}$. The third random variable X is important for the design goal of $p(t|y)$ and is termed the *relevant random variable*. In order to explain the role of the relevant random variable X, it is first important to note that variables X and Y share the mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$. This mutual information is determined by their joint distribution $p(x, y)$ as it is implied by Equation (2.23). As it

is suggested by the nomenclature used, the random variable X defines which features in Y are considered to be *relevant*. Tishby *et al.* consequently describe this approach as the principle of defining *relevance through another variable* in [3]. The mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ quantifies the *relevant information* between X and Y. The key idea of the Information Bottleneck method is now to extract this relevant information while compressing Y to T which means that T shall be highly informative about X. These demands on the desired compression and the preservation of relevant information can be formalized as the side constrained optimization problem of finding $p(t|y)$ according to

$$\min_{p(t|y)} \mathrm{I}(\mathsf{Y};\mathsf{T}), \quad \text{such that } \mathrm{I}(\mathsf{X};\mathsf{T}) \geq D \tag{3.2}$$

for some $D > 0$. An upper bound for $\mathrm{I}(\mathsf{X};\mathsf{T})$ is given by $\mathrm{I}(\mathsf{X};\mathsf{Y})$ as processing of Y cannot generate any new information on X. Therefore, it is appropriate to interprete $\mathrm{I}(\mathsf{X};\mathsf{T})$ as the *preserved* relevant information under compression. It is important that the Information Bottleneck method only requires knowledge of the joint distribution $p(x, y)$ to determine the desired compression mapping $p(t|y)$. Algorithms to determine $p(t|y)$ are described in Section 3.2.

In summary, both mentioned approaches in Equations (3.1) and (3.2) aim to determine a compression mapping $p(t|y)$ under a side constraint. For the Information Bottleneck method, this side constraint is the preservation of relevant information, while rate-distortion theory aims to keep the expected distortion below a certain threshold. It is important that in rate-distortion theory a proper distortion function has to be defined in advance and that its choice influences $p(t|y)$. In contrast, the information constraint of preserving $\mathrm{I}(\mathsf{X};\mathsf{T})$ in the Information Bottleneck method does not require such a preliminary definition, but is symmetric in the sense that it describes the compression aim in the design of $p(t|y)$ and also its side constraint on the preservation of relevant information only by means of mutual information.

Different to rate-distortion theory which requires knowledge of the distribution $p(y)$, a set of representatives $\mathcal{T}$ and a proper distortion measure $d(t, y)$ to determine $p(t|y)$, the Information Bottleneck requires only knowledge of joint distribution $p(x, y)$ which connects the relevant variable X and the observation Y. It is especially important that the Information Bottleneck method does not require to find a set $\mathcal{T}$ of optimum representation values, as the mutual

information $\mathrm{I}(\mathsf{X};\mathsf{T})$ does not depend on the particular elements of $\mathcal{T}$, but only on the probabilities implied by $p(x,t)$. Interestingly, also the side constrained optimization problem of the Information Bottleneck method can be solved using the method of Lagrangian multipliers, as it will be explained in detail in the following section.

3.1.1 The Self Consistent Information Bottleneck Equations

This section introduces three self consistent equations which characterize the compression mapping $p(t|y)$ and the distributions $p(x|t)$ and $p(t)$ in the Information Bottleneck problem setup. These equations form the basis of all Information Bottleneck algorithms which will be used later to determine the desired compression mapping $p(t|y)$. The aim of this section is to describe their derivation in great detail because it is mathematically quite involved and the available literature on the topic does not provide such a detailed derivation. However, the major mathematical part of the derivation is provided in Appendix A.5 for better readability of this section.

The problem of finding a suitable conditional distribution $p(t|y)$ according to Equation (3.2) is the problem of finding extreme points of a *function* $\mathrm{I}(\mathsf{Y};\mathsf{T})$ of several variables under side constraints. This problem can be tackled using the Lagrange method. Following [3], a Lagrangian approach to determine $p(t|y)$ according to Equation (3.2) first requires to define the so called Lagrangian

$$\mathcal{L}(p(t|y)) = \mathrm{I}(\mathsf{Y};\mathsf{T}) - \beta \mathrm{I}(\mathsf{X};\mathsf{T}) - \sum_{y\in\mathcal{Y}} \lambda(y) \sum_{t\in\mathcal{T}} p(t|y). \tag{3.3}$$

which is a function of $p(t|y)$. The Lagrangian is mainly determined by the function to be minimized which is $\mathrm{I}(\mathsf{Y};\mathsf{T})$. In addition, the side constraints of the optimization problem are attached to the function to be minimized using the so called Lagrangian multipliers which are β and $\lambda(y)$ in this case. Each Lagrangian multiplier belongs to one side constraint of the minimization problem. First, the minimization of $\mathrm{I}(\mathsf{Y};\mathsf{T})$ shall be carried out while preserving $\mathrm{I}(\mathsf{X};\mathsf{T})$ which is coupled in using the Lagrangian multiplier $\beta \geq 0$. Second, $p(t|y)$ has to be a valid probability distribution which fulfills

$$\sum_{t\in\mathcal{T}} p(t|y) = 1 \quad \forall y \in \mathcal{Y}. \tag{3.4}$$

This side constraint is coupled in using the Lagrangian multipliers $\lambda(y)$ in Equation (3.3). Please note that there are $|\mathcal{Y}|$ such multipliers.

From here, extensive application of differential calculus and considering the Markov chain relation $\mathsf{X} \to \mathsf{Y} \to \mathsf{T}$ yields three self consistent equations which describe an implicit connection between $p(t|y)$, $p(t)$ and $p(x|t)$. These equations are

$$p(t|y) = \frac{p(t)}{Z(y,\beta)} \exp\left(-\beta D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}\right) \tag{3.5}$$

$$p(t) = \sum_{y\in\mathcal{Y}} p(t|y)p(y) \tag{3.6}$$

$$p(x|t) = \frac{1}{p(t)} \sum_{y\in\mathcal{Y}} p(t|y)p(x,y) \tag{3.7}$$

and are termed the self consistent Information Bottleneck equations [9]. In Equation (3.5), $Z(y,\beta)$ is a normalization function which ensures that $p(t|y)$ is a valid probability distribution for all $y \in \mathcal{Y}$. A complete derivation of Equations (3.5)-(3.7) together with some comments on the mathematical techniques required to derive them can be found in Appendix A.5. These equations enable algorithmic approaches to find $p(t|y)$ for fixed values of $\beta \geq 0$ which will be described in the Section 3.2.

3.1.2 The Trade-Off Parameter β

The parameter β which results from application of the Lagrange method plays the role of a trade-off parameter. The choice of $\beta \geq 0$ allows to trade preservation of relevant information for compression. For $\beta \to +\infty$, $p(t|y)$ aims to preserve a maximum amount of relevant information. For $\beta = 0$, the Lagrangian (3.3) becomes independent of $\mathrm{I}(\mathsf{X};\mathsf{T})$ and hence $p(t|y)$ only fulfills the compression goal of minimizing $\mathrm{I}(\mathsf{Y};\mathsf{T})$.

In the literature on the Information Bottleneck method the Lagrangian (3.3) is sometimes provided without the terms including $\lambda(y)$ and called Information Bottleneck functional [3,9]. The respective optimization problem to find $p(t|y)$ then reads

$$\min_{p(t|y)} \mathrm{I}(\mathsf{Y};\mathsf{T}) - \beta \mathrm{I}(\mathsf{X};\mathsf{T}), \tag{3.8}$$

but requires the additional restriction of $p(t|y)$ being a valid conditional probability distribution. Another very intuitive problem formulation is obtained

by assuming $\beta > 0$ and dividing Equation (3.8) by $-\beta$. Doing so results in the problem formulation

$$\max_{p(t|y)} \mathrm{I}(\mathsf{X};\mathsf{T}) - \beta^{-1}\mathrm{I}(\mathsf{Y};\mathsf{T}), \tag{3.9}$$

which directly indicates that the aim of the Information Botlleneck method for $\beta \to +\infty$ is to maximize the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$. If no other constraint was imposed, a trivial solution for $p(t|y)$ which results in $\mathrm{I}(\mathsf{X};\mathsf{T}) = \mathrm{I}(\mathsf{X};\mathsf{Y})$ would therefore always be an identity mapping. A constraint, however, also for $\beta \to +\infty$ can easily be imposed by upper bounding the cardinality $|\mathcal{T}|$ of the compression random variable T such that it is smaller than $|\mathcal{Y}|$. Hence, for $\beta \to +\infty$ the Information Bottleneck problem setup can be understood as the problem of finding $p(t|y)$ accoding to

$$\max_{p(t|y)} \mathrm{I}(\mathsf{X};\mathsf{T}) \quad \text{for a fixed } |\mathcal{T}| < |\mathcal{Y}|. \tag{3.10}$$

This case of a desired maximum amount of preserved relevant information under a restriction on the cardinality of the event space of the compression variable is of very high practical interest. The reason is that the cardinality of the event space of the compression variable T directly determines how many bits are required to store a realization $t \in \mathcal{T}$ in digital hardware.

3.1.3 Soft and Hard Clustering

The conditional probability distribution $p(t|y)$ describes a probabilistic mapping of all $y \in \mathcal{Y}$ onto $t \in \mathcal{T}$. An illustrative interpretation of this fact is understanding $p(t|y)$ as a clustering rule for the event space $\mathcal{Y}$ of the observation Y into $|\mathcal{T}|$ (not necessarily disjoint) subsets $\mathcal{Y}_t$. The subsets $\mathcal{Y}_t$ are often called *clusters* in this case and can be identified by their index $t \in \mathcal{T}$. In the following, $\mathcal{T}$ is considered to be a set of integers indices $\mathcal{T} = \{0, 1, \ldots, |\mathcal{T}|-1\}$, such that the clusters can simply be enumerated as $\mathcal{Y}_0, \mathcal{Y}_1, \ldots, \mathcal{Y}_{|\mathcal{T}|-1}$. The union of all clusters $\mathcal{Y}_t$ is the event space $\mathcal{Y}$ of the observation. However, the understanding of clustering as simple partitioning of a set into several subsets has to be extended to a *soft* partitioning in this context. For each element y the probabilities $\Pr(\mathsf{T} = t|\mathsf{Y} = y)$ implied by $p(t|y)$ are the probabilies that $y \in \mathcal{Y}$ is in a certain cluster $\mathcal{Y}_t$ with cluster index $t \in \mathcal{T}$. Hence, in general elements $y \in \mathcal{Y}$

can be in several clusters with certain probabilities. In fact, the case where $p(t|y)$ introduces a deterministic mapping, such that $p(t|y) \in \{0,1\}\,\forall(t,y)$ can be seen as a special case of a degenerated probabilistic clustering which assigns all the probability mass to only one cluster.

Example 3.1.3.1. Figure 3.2 illustrates the difference between a soft and a hard clustering characterized by two exemplary conditional distributions $p(t|y)$. In this example, the event space of the observation is $\mathcal{Y} = \{0, 1, \ldots, 7\}$ and this event space is clustered into two different clusters $\mathcal{Y}_0$ and $\mathcal{Y}_1$. The upper part of the illustration in Figure 3.2a shows an exemplary soft clustering. In the shown example, all elements $y \in \mathcal{Y}$ appear in both clusters $\mathcal{Y}_0$ and $\mathcal{Y}_1$. The opacity of the elements reflects the probability of the cluster membership which are provided numerically below the illustration. Figure 3.2b at the bottom, in contrast, shows an exemplary hard clustering. There, each element appears only in one cluster $\mathcal{Y}_0$ or $\mathcal{Y}_1$. Hence, all probabilites in $p(t|y)$ are either 0, meaning that an element y is not in cluster $\mathcal{Y}_t$, or 1, meaning that the respective element y is in cluster $\mathcal{Y}_t$ without any probabilistic uncertainty. — *End of the example.*

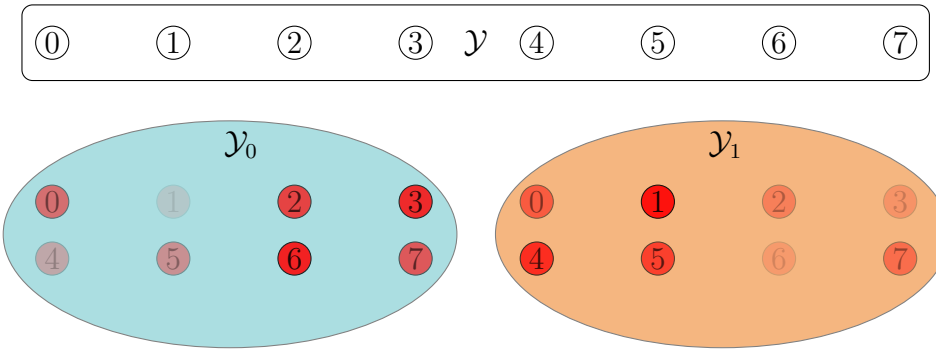
In fact, the special case of a deterministic clustering is of very high practical interest for the problems considered in this thesis because a desired maximum amount of preserved relevant information which corresponds to the choice $\beta \to +\infty$ is achieved with a deterministic clustering [58].

A deterministic mapping $p(t|y)$ can equivalently be interpreted as a function $f : \mathcal{Y} \to \mathcal{T}$ which maps each $y \in \mathcal{Y}$ directly onto the corresponding cluster index $t \in \mathcal{T}$, such that $t = f(y)$. A convenient mathematical description of a deterministic mapping $p(t|y)$ can be done using the Kronecker delta function

$$p(t|y) = \delta(t - f(y)) \Leftrightarrow t = f(y). \tag{3.11}$$

Any deterministic function $f(y)$ can be interpreted as a lookup table which tells which $t = f(y)$ appears for which y. Hence, the compression operation for a given deterministic mapping $p(t|y)$ can be implemented using a simple lookup table. This fact is illustrated in the following example.

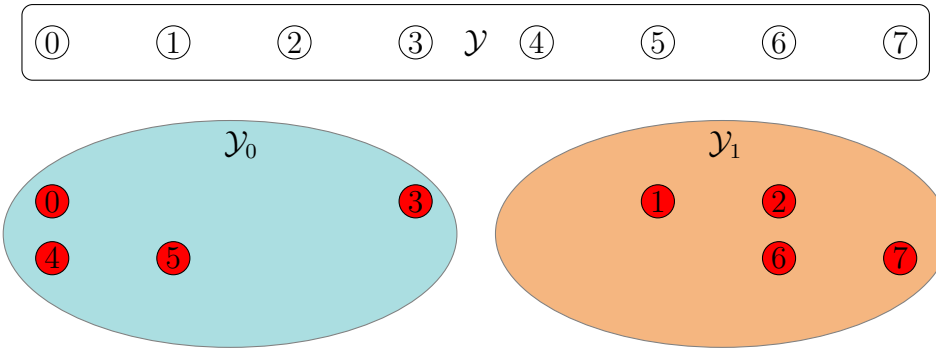
Example 3.1.3.2. For the deterministic mapping $p(t|y)$ illustrated in Figure 3.2b, an equivalent description is the lookup table from Table 3.1. The lookup



$\leftarrow$ all $y \in \mathcal{Y} \rightarrow$

all $t \in \mathcal{T}$	$p(t\|y) \searrow$	0	1	2	3	4	5	6	7
	0	0.50	0.10	0.70	0.81	0.25	0.35	0.85	0.60
	1	0.50	0.90	0.30	0.19	0.75	0.65	0.15	0.40

(a) Illustration of soft clustering. Elements from $\mathcal{Y}$ may appear in several clusters with a certain probability implied by $p(t|y) \in [0, 1]$.



$\leftarrow$ all $y \in \mathcal{Y} \rightarrow$

all $t \in \mathcal{T}$	$p(t\|y) \searrow$	0	1	2	3	4	5	6	7
	0	1.00	0.00	0.00	1.00	1.00	1.00	0.00	0.00
	1	0.00	1.00	1.00	0.00	0.00	0.00	1.00	1.00

(b) Illustration of hard clustering. All elements from $\mathcal{Y}$ only appear in one cluster. Therefore, $p(t|y) \in \{0, 1\}\, \forall (t, y) \in \mathcal{T} \times \mathcal{Y}$.

Figure 3.2: Illustration of soft and hard clustering.

Table 3.1: Exemplary lookup table description of a deterministic mapping.

input y	0	1	2	3	4	5	6	7
output t	0	1	1	0	0	0	1	1

table has one entry for each $y \in \mathcal{Y}$. The lookup table entry in column y simply is the cluster index t of the cluster $\mathcal{Y}_t$ which holds y with probability 1. — *End of the example.*

Since a desired maximum amount of preserved mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$ can be achieved with a deterministic $p(t|y)$, deterministic mappings $p(t|y)$ will play the major role in this thesis. The technique of determining and applying lookup tables from deterministic mappings $p(t|y)$ will be used extensively. Thereby, the focus will always lie on a maximum desired amount of preserved relevant information. As a result, in the remainder of the thesis $\beta \to +\infty$ is typically assumed.

3.1.4 Beliefs Associated with Cluster Indices

There exist many technical applications, where a quite intuitive link between a realization y and the realization x which caused this y according to the Markov chain relation $\mathsf{X} \to \mathsf{Y}$ exists. As an example, consider a temperature x and its measurement y which might be affected by some additive noise or some other imperfectness of the measurement device, for example, a bias. If set up properly, the measurement y should approximate the true x, meaning that the measurement approximates the true value of the measured quantity, that is, $x \approx y$.

A quite confusing fact about the probabilistic notion of compression which is crucial for the remainder of this thesis is that application of a compression mapping $p(t|y)$, that means transfering y to t, despite aiming to preserve relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ typically looses such a relation. This means, that an approximation relation that exists between x and y does not necessarily exist between x and t. The reason for this is that the mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$ is independent of the elements in the event spaces of X and T and, therefore, the event space $\mathcal{T}$ of the compression variable can just be any set of distinguishable objects. Above, $\mathcal{T}$ was chosen to be a set of indices. If

we stick with the example of a temperature measurement, it should be clear immediately that mapping a measured temperature from a certain interval, for example, $[-40, +40]$ degrees Celsius to an integer index can destroy the physical meaning of the measurement in the compressed representation.

However, the bridge between X and T that connects the compressed representation to its physical meaning is the conditional distribution $p(x|t)$ which provides a belief on the relevant variable X for any given $t \in \mathcal{T}$, just as $p(x|y)$ does for any given $y \in \mathcal{Y}$. In fact, despite t might not approximate y in absolute terms, as it has just been explained, $p(x|t)$ can still approximate $p(x|y)$ for some pair (t, y) quite well, anyway.

3.1.5 The Information Rate Function

The achievable trade-off between preservation of relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ and remaining compression information $\mathrm{I}(\mathsf{Y};\mathsf{T})$ can be described using the so called information rate function. The information rate function is defined as [59]

$$I(R') = \max_{p(t|y)} \mathrm{I}(\mathsf{X};\mathsf{T}), \text{ such that } \mathrm{I}(\mathsf{Y};\mathsf{T}) \leq R'. \tag{3.12}$$

It tells the maximum achievable relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ for a desired upper bound of $\mathrm{I}(\mathsf{Y};\mathsf{T})$. In this context, the compression information $\mathrm{I}(\mathsf{Y};\mathsf{T})$ is typically termed the *(compression) rate* and hence denoted by $R' = \mathrm{I}(\mathsf{Y};\mathsf{T})$. The curve of the information rate function is plotted in the so called relevance compression plane [9] which shows the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ over the remaining compression information $\mathrm{I}(\mathsf{Y};\mathsf{T})$. It separates this plane into an achievable and a not achievable region. Figure 3.3 shows qualitative examples. The achievable region for a given compression cardinality $|\mathcal{T}|$ lies below the graph of the respective information rate function for this cardinality. The not achievable region lies above it. Typically, larger compression cardinalities allow for higher preservation of relevant information, as it is illustrated. The mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ is an upper bound for $\mathrm{I}(\mathsf{X};\mathsf{T})$, as is also illustrated.

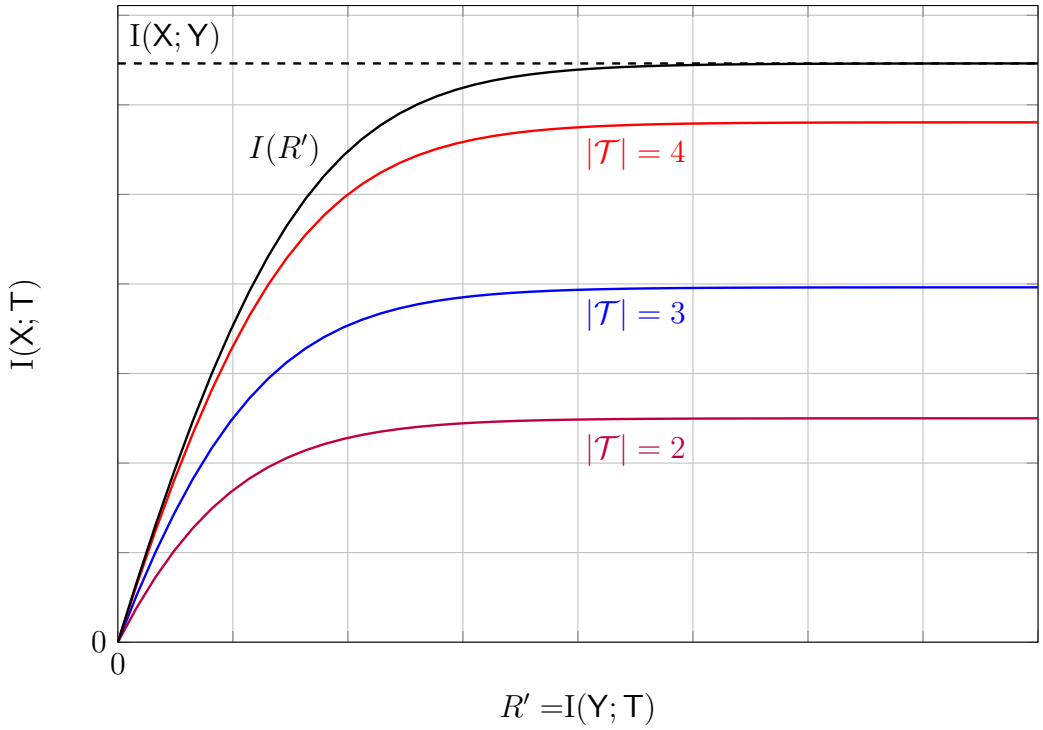


Figure 3.3: Qualitative graphs of exemplary information rate functions in the relevance compression plane.

3.2 Information Bottleneck Algorithms

An important question arising is how to determine conditional probability distributions $p(t|y)$ which result in the desired characteristic of minimizing $\mathrm{I}(\mathsf{Y};\mathsf{T})$ while maximizing $\mathrm{I}(\mathsf{X};\mathsf{T})$. The structure of the underlying optimization problem is known to depend on the trade-off parameter β. The case $\beta = 0$ is of no interest because it preserves no relevant information. For finite values of β, the objective function in Equation (3.8) is neither concave nor convex [24]. Therefore, finding a *global* optimum is cumbersome with standard optimization methods in general. For the very important case of $\beta \to +\infty$ and a given cardinality $|\mathcal{T}|$ the maximization of $\mathrm{I}(\mathsf{X};\mathsf{T})$ from Equation (3.10) is known to be the problem of maximizing a convex function over a closed convex set [24, 58]. This problem is also known as *concave* optimization [24, 58]. Solving such problems has non polynomial complexity in general [58].

Due to these facts, several heuristic algorithms have been introduced in the past which at least find locally optimal solutions for the considered problem.

Input : joint distribution $p(x, y)$, trade-off parameter β, cardinality of the compression variable $|\mathcal{T}|$

Output: mapping $p(t|y)$, $p(x|t)$ and $p(t)$

begin

$i \leftarrow 0$;

$p^{(i)}(t|y) \leftarrow$ random initial clustering;

$p^{(i)}(t) \leftarrow$ according to Equation (3.6);

$p^{(i)}(x|t) \leftarrow$ according to Equation (3.7);

while *not converged* **do**

$p^{(i+1)}(t|y) \leftarrow \frac{p^{(i)}(t)}{Z^{(i)}(y,\beta)} \exp\left(-\beta D_{\mathrm{KL}}\{p(x|y) \,|\, p^{(i)}(x|t)\}\right)$;

$p^{(i+1)}(t) \leftarrow \sum_{y \in \mathcal{Y}} p^{(i+1)}(t|y) p(y)$;

$p^{(i+1)}(x|t) \leftarrow \frac{1}{p^{(i+1)}(t)} \sum_{y \in \mathcal{Y}} p^{(i+1)}(t|y) p(x, y)$;

$i \leftarrow i + 1$;

end

end

return $p(t|y), p(x|t), p(t)$;

Algorithm 1: Algorithmic description of the Iterative Information Bottleneck algorithm.

These algorithms form the group of *Information Bottleneck algorithms*. In the following, three such algorithms are described in detail. These algorithms were adapted from [3,9,60]. A collection of more Information Bottleneck algorithms can be found compactly consolidated in [9].

3.2.1 The Iterative Information Bottleneck Algorithm

The iterative Information Bottleneck algorithm has been described in [3] together with the introduction of the Information Bottleneck method. It was inspired by the iterative processing of the Blahut-Arimoto algorithm and works quite similarly. The general idea is starting from a random initial clustering $p(t|y)$ and iterating between the self consistent Information Bottleneck Equations (3.5)–(3.7). The iterative Information Bottleneck algorithm is compactly described in Algorithm 1. In Algorithm 1, superscripts (i) were added to the

appearing probability distributions to distinguish the iterations of the algorithm, for example, $p^{(i)}(t|y)$.

The iterative Information Bottleneck algorithm inputs the joint probability distribution $p(x, y)$, the trade-off parameter β and the desired compression cardinality $|\mathcal{T}|$ of the compression variable T. In the first step of initialization, the algorithm randomly draws an initial soft mapping $p^{(0)}(t|y)$. It is important that this mapping has to be a valid probability distribution for all $y \in \mathcal{Y}$. Afterwards, starting from this initial mapping, Equation (3.6) and Equation (3.7) are evaluated. The rest of this very simple algorithm is just determining refined versions of $p(t|y)$, $p(t)$ and $p(x|t)$ by recursive evaluation of the self consistent Information Bottleneck equations. The algorithm processing has to be stopped, when a certain exit criterion is met. Typically, one can stop if the mapping $p^{(i+1)}(t|y)$ does not differ from its ancestor $p^{(i)}(t|y)$ from the preliminary iteration mentionably. A formal proof of the convergence to at least a local optimum $p(t|y)$ is provided in [3]. Then the algorithm outputs the final $p(t|y)$. As side products, the algorithms' processing delivers $p(x|t)$ and $p(t)$ through Equations (3.6) and (3.7) which can be used to determine the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$. Please note, that the algorithm can equivalently be considered to deliver $p(x, t) = p(x|t)p(t)$.

Due to the nature of the optimization problem, finding a globally optimum $p(t|y)$ is not guaranteed. Therefore, the iterative Information Bottleneck algorithm normally should be run several times with different initial clusterings and the best found result should be used.

3.2.2 The Sequential Information Bottleneck Algorithm

The sequential Information Bottleneck algorithm was proposed in [7] for unsupervised document classification and is also described in [9]. It is particularly interesting for several reasons:

- The algorithm only optimizes over deterministic mappings $p(t|y)$.
- Numerical stability in the case $\beta \longrightarrow +\infty$ is guaranteed.
- It is easily adaptable to the problem of finding relevant information preserving channel output quantizers which will be investigated in Section 5.1.

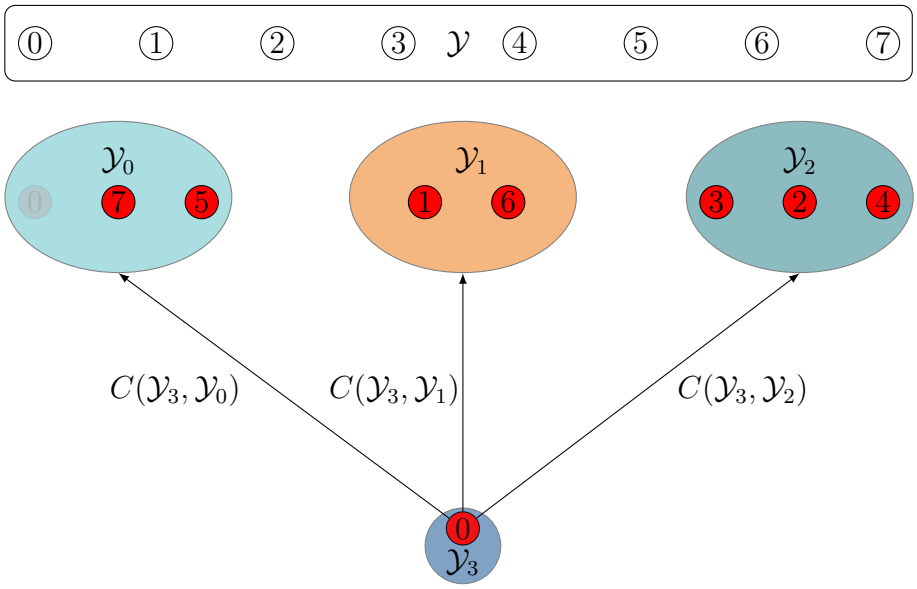


Figure 3.4: Processing of the sequential Information Bottleneck algorithm. The algorithm starts from a random initial deterministic clustering of $\mathcal{Y}$. It then takes out single elements y of their cluster and merges them back into the cluster minimizing the merger costs.

The processing of the sequential Information Bottleneck algorithm is illustrated in Figure 3.4 and described in the following. Moreover, a compact algorithmic description is provided in Algorithm 2. The sequential Information Bottleneck algorithm has the same input parameters as the iterative Information Bottleneck algorithm. It starts from a random initial hard clustering of $\mathcal{Y}$ into $|\mathcal{T}|$ clusters $\mathcal{Y}_0, \mathcal{Y}_1, \ldots, \mathcal{Y}_{|\mathcal{T}|-1}$.

The algorithm tries to improve this initial clustering sequentially by taking all the elements $y \in \mathcal{Y}$ out of their current cluster and putting them into other clusters. Once an element y is taken out of its cluster, it is put into a so called singleton cluster which only holds this one element. For a simple notation the singleton cluster is simply labeled $\mathcal{Y}_{|\mathcal{T}|}$.

At the beginning of the algorithm element $y = 0$ is taken out of its current cluster which is $\mathcal{Y}_0$ in the considered example in Figure 3.4. Note that the desired clustering of $\mathcal{Y}$ shall consist of $|\mathcal{T}|$ clusters, but including the singleton cluster there now exist $|\mathcal{T}| + 1$ clusters. In the shown example, $\mathcal{Y}_{|\mathcal{T}|} = \mathcal{Y}_3$. Of course, moving elements between the clusters modifies $p(t|y)$. Essentially, a new cluster $\mathcal{Y}_{|\mathcal{T}|}$ has been added and therefore T can take an additional value.

Input : joint distribution $p(x, y)$, trade-off parameter β, cardinality of the compression variable $|\mathcal{T}|$

Output: deterministic mapping $p(t|y)$, $p(x|t)$ and $p(t)$

```
begin
    done ← false;
    p(t|y) ← random initial deterministic clustering;
    p(t) ← according to Equation (3.6);
    p(x|t) ← according to Equation (3.7);
    while not done do
        done ← true;
        for y ∈ 𝒴 do
            Put y from original cluster to singleton cluster;
            Calculate merger costs C(𝒴_{|𝒯|}, 𝒴_t) ∀t ∈ 𝒯;
            Merge singleton cluster into 𝒴_t with minimum merger costs;
            if not merged into original cluster then
                done←false;
            end
        end
    end
end
return p(t|y), p(x|t), p(t);
```

Algorithm 2: Algorithmic description of the sequential Information Bottleneck algorithm.

As a consequence, $p(t)$ and $p(x|t)$ have to be updated according to Equations (3.6) and (3.7) after an element y has been moved into the singleton cluster.

After the update step, the sequential Information Bottleneck algorithm merges the singleton cluster into one of the original clusters $\mathcal{Y}_t$ to reduce the number of clusters to $|\mathcal{T}|$ again. All original clusters $\mathcal{Y}_t$, $t \in \{0, 1, \ldots, |\mathcal{T}| - 1\}$ are considered as target clusters. The merging possibilities are illustrated as arrows connecting the singleton cluster to the possible target clusters in Figure 3.4. The target cluster is determined by a minimum decision on so called merger costs $C(\mathcal{Y}_{|\mathcal{T}|}, \mathcal{Y}_t)$ of merging clusters $\mathcal{Y}_{|\mathcal{T}|}$ and $\mathcal{Y}_t$. These merger costs

need to be calculated for all possible target clusters which dominates the complexity of the algorithm.

The merger costs $C(\mathcal{Y}_{|\mathcal{T}|}, \mathcal{Y}_t)$ decide which target cluster is chosen to reduce the number of clusters to the desired number of $|\mathcal{T}|$ again. Therefore, their definition is essential. According to the maximization aim in Equation (3.9), the sequential Information Bottleneck algorithm merges the singleton cluster into the target cluster minimizing the loss of $\mathrm{I}(\mathsf{X};\mathsf{T}) - \beta^{-1}\mathrm{I}(\mathsf{Y};\mathsf{T})$ which results from the merging procedure. The merger costs $C(\mathcal{Y}_{|\mathcal{T}|}, \mathcal{Y}_t)$ simply correspond to the change of $\mathrm{I}(\mathsf{X};\mathsf{T}) - \beta^{-1}\mathrm{I}(\mathsf{Y};\mathsf{T})$ that results from merging the singleton cluster $\mathcal{Y}_{|\mathcal{T}|}$ into cluster $\mathcal{Y}_t$. Note that for $\beta \longrightarrow +\infty$ the coefficient of $\mathrm{I}(\mathsf{Y};\mathsf{T})$ in (3.9) vanishes and the merger costs simply are the respective changes of mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$ that result from merging $\mathcal{Y}_{|\mathcal{T}|}$ into a particular target cluster $\mathcal{Y}_t$.

The change of the target function in Equation (3.9) resulting from merging the singleton cluster $\mathcal{Y}_{|\mathcal{T}|} = \{y\}$ into one of the other clusters can be calculated in closed form. The merger costs are derived in [9]. In the notation of this thesis, they are given by

$$C(\mathcal{Y}_{|\mathcal{T}|}, \mathcal{Y}_t) = \psi(t,y)\, D_{\mathrm{JS}}^{(\pi_0,\pi_1)}\{p(x|\mathsf{Y}=y)\,|\,p(x|\mathsf{T}=t)\} - \psi(t,y)\beta^{-1} D_{\mathrm{JS}}^{(\pi_0,\pi_1)}\{p(y)\,|\,p(y|\mathsf{T}=t)\}, \quad (3.13)$$

where $D_{\mathrm{JS}}^{(\pi_0,\pi_1)}\{p(x)\,|\,q(x)\}$ is the Jensen-Shannon divergence with weights π_0, π_1 and

$$\psi(t,y) = \Pr(\mathsf{Y}=y) + \Pr(\mathsf{T}=t) \quad (3.14)$$

$$\pi_0 = \pi_0(t,y) = \Pr(\mathsf{Y}=y)/\psi(t,y) \quad (3.15)$$

$$\pi_1 = \pi_1(t,y) = \Pr(\mathsf{T}=t)/\psi(t,y). \quad (3.16)$$

The Jensen-Shannon divergence $D_{\mathrm{JS}}^{(\pi_0,\pi_1)}\{p(x)\,|\,q(x)\}$ is a convex combination of two Kullback-Leibler divergences. It is calculated as

$$D_{\mathrm{JS}}^{(\pi_0,\pi_1)}\{p(x)\,|\,q(x)\} = \pi_0\, D_{\mathrm{KL}}\{p(x)\,|\,\bar{p}(x)\} + \pi_1\, D_{\mathrm{KL}}\{q(x)\,|\,\bar{p}(x)\}, \quad (3.17)$$

where $\bar{p}(x) = \pi_0\, p(x) + \pi_1\, q(x)$ [7, 9].

The algorithm repeats the draw and merge steps for all $y \in \mathcal{Y}$ sequentially in the explained manner and stops when the obtained clustering does not change

any more. This convergence can be guaranteed according to the argumentation in [7]. Clearly, the algorithm is a greedy optimization algorithm. Just as the iterative Information Bottleneck algorithm, it is not guaranteed to find a globally optimum solution. This is why it has to be run with several different initial clusterings and the obtained $p(t|y)$ corresponding to the largest $\mathrm{I}(\mathsf{X};\mathsf{T}) - \beta^{-1}\mathrm{I}(\mathsf{Y};\mathsf{T})$ has to be chosen.

A modified variant of the sequential Information Bottleneck algorithm which is dedicated to the problem of finding scalar channel output quantizers will be developed in Section 5.1.1. This algorithm has drastically reduced computational costs in comparison to the sequential Information Bottleneck algorithm for this problem.

3.2.3 The KL-Means Algorithm

The KL-means algorithm is a modified variant of the K-means algorithm, which is a well known clustering algorithm from the machine learning field [61]. The general idea of the KL-means algorithm is replacing the similarity metric of the K-means algorithm, which is a metric involving the Euclidian distance, with the Kullback-Leibler divergence. Despite having been introduced in the machine learning field before, the KL-means algorithm has been reconsidered in [60] for the problem of finding quantizers for discrete memoryless channels with non-binary inputs, where a connection to the Information Bottleneck problem has not been explicitly mentioned. Anyway, the KL-means algorithm is applicable to arbitrary relevant information preserving clustering tasks and can essentially be interpreted as an Information Bottleneck algorithm with $\beta \to +\infty$. A formal description of the equivalence of the KL-means algorithm and other Information Bottleneck algorithms can be found in [22].

Just as the sequential Information Bottleneck algorithm from Section 3.2.2, the KL-means algorithm aims to find a hard clustering of the event space $\mathcal{Y}$ of an observation into $|\mathcal{T}|$ clusters $\mathcal{Y}_t$ for the Markov chain $\mathsf{X} \to \mathsf{Y} \to \mathsf{T}$, such that the mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$ is maximized under a constraint for the cardinality of $\mathcal{T}$. The algorithms' processing is summarized in Algorithm 3 and explained in the following.

The KL-means algorithm inputs the joint distribution $p(x, y)$, the desired cardinality of the compression variable $|\mathcal{T}|$ and a maximum number of allowed

Input : joint distribution $p(x, y)$, cardinality of the compression variable $|\mathcal{T}|$, maximum number of iterations $i_{\max}^{\mathrm{KL}}$
Output: deterministic mapping $p(t|y)$, $p(x|t)$ and $p(t)$

begin
  done $\leftarrow$ false;
  $i \leftarrow 0$;
  $p(x|y) \leftarrow p(x, y)/p(y)\, \forall y \in \mathcal{Y}$;
  Randomly choose $|\mathcal{T}|$ different $p(x|t)$ from $p(x|y)$ as cluster means;
  while *not done* **do**
    Group all $y \in \mathcal{Y}$ to cluster with closest mean $p(x|t)$:
    $\mathcal{Y}_t \leftarrow \left\{ y | D_{\mathrm{KL}} \left\{ p(x|y) \,|\, p(x|t) \right\} \rightarrow \min_{t \in \mathcal{T}} \right\}$;
    Update $p(t) \leftarrow$ according to Equation (3.6);
    Update the means $p(x|t)$ according to Equation (3.7);
    $i \leftarrow i + 1$;
    if *clustering did not change or* $i == i_{\max}^{\mathrm{KL}}$ **then**
      done$\leftarrow$true;
    end
  end
end
return $p(t|y), p(x|t), p(t)$;

Algorithm 3: Algorithmic description of the KL-means algorithm.

iterations $i_{\max}^{\mathrm{KL}}$. The further description of the algorithm requires to explain some wording used in the context of the K-means algorithm.

Each cluster $\mathcal{Y}_t$ is assigned a so called *mean*. A cluster mean in the KL-means algorithm corresponds to the conditional probability distribution $p(x|t)$ for a certain $t \in \mathcal{T}$ in the introduced notation of the Information Bottleneck method. The general idea is that the so called mean $p(x|t)$ is representative for the meaning of cluster $\mathcal{Y}_t$ with respect to the relevant variable X. Due to this, the KL-means algorithm tries to group all $y \in \mathcal{Y}$ which imply similar beliefs $p(x|y)$ on the relevant variable into one cluster with mean $p(x|t)$. As stated earlier, the measure of similarity used in the KL-means algorithm is the Kullback-Leibler divergence $D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}$ which shall be iteratively

minimized for the decision of the cluster assignments of all $y \in \mathcal{Y}$. Thus, in the KL-means algorithm $D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}$ can be considered as costs of putting y into a cluster $\mathcal{Y}_t$. Please note that these costs evaluate the similarity of $p(x|y)$ and $p(x|t)$ with the exact same measure which emerged from the derivation of the self consistent Information Bottleneck equations. This fact is not a coincidence. The authors of [60] showed that finding a $p(t|y)$ such that the expectation

$$\mathbb{E}_{\mathsf{Y},\mathsf{T}}\{D_{\mathrm{KL}}\{p(x|\mathsf{Y}=y) \,|\, p(x|\mathsf{T}=t)\}\} \to \min \tag{3.18}$$

is equivalent to maximizing the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$. This obvervation was the motivation to use the Kullback-Leibler divergence as a similarity measure in the K-means algorithm to obtain the KL-means algorithm in [60].

To begin its processing, the KL-means algorithm first determines the conditional distribution $p(x|y)$. Afterwards it randomly picks a total of $|\mathcal{T}|$ posterior distributions $p(x|\mathsf{Y}=y')$ and sets them as initial cluster means $p(x|t)$. The algorithm then calculates the Kullback-Leibler divergences $D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}$ for all remaining $y \in \mathcal{Y}$ and all $t \in \mathcal{T}$. Then each $y \in \mathcal{Y}$ is put into the cluster $\mathcal{Y}_t$ with the closest mean $p(x|t)$ under this Kullback-Leibler divergence. Afterwards, the cluster means have to be updated in order to again correspond to the meaning $p(x|t)$ with respect to the relevant random variable X reflected by the cluster $\mathcal{Y}_t$. This update is carried out using the Equations (3.6) and (3.7) which already appeared in the Information Bottleneck setup.

The KL-means algorithm then works iteratively and tries to identify if there exists a cluster $\mathcal{Y}_t$ with a more appropriate mean, that is, a smaller $D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}$ for each element $y \in \mathcal{Y}$ in each iteration. Therefore, the algorithm in each iteration loops over all $y \in \mathcal{Y}$ and calculates $|\mathcal{T}|$ Kullback-Leibler divergences $D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}$, $t \in \mathcal{T}$ for each $y \in \mathcal{Y}$. Once all these divergence measures are obtained the cluster with the closest mean is chosen for each $y \in \mathcal{Y}$.

Then the means $p(x|t)$ are updated again according to the new constellation of the clusters. This processing is repeated until the clustering does not change any more, or the allowed maximum number of iterations $i_{\max}^{\mathrm{KL}}$ has been performed. When it has terminated, it outputs the determinstic mapping $p(t|y)$ and just as the other introduced Information Bottleneck algorithms

delivers $p(x|t)$ and $p(t)$ as side products. Since the KL-means algorithm also is a greedy optimization algorithm which does not necessarily find a globally optimum clustering, it should be initialized with several distinct initial means and the result corresponding to the largest $\mathrm{I}(\mathsf{X};\mathsf{T})$ should be chosen.

It is worth mentioning that the processing of the KL-means algorithm offers an advantage over the one of the sequential Information Bottleneck algorithm. Despite both algorithms handle all $y \in \mathcal{Y}$ in each iteration and both algorithms also evaluate $|\mathcal{T}|$ cost functions for putting an element y to a certain cluster $\mathcal{Y}_t$, a key difference is that in the KL-means algorithm all $D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}$, $t \in \mathcal{T}$ for all $y \in \mathcal{Y}$ can be calculated independently because $p(x|t)$ is fixed for this step. In contrast, in the sequential Information Bottleneck algorithm, $p(x|t)$ is updated after each merge of an element y. Therefore, the calculation of the Kullback-Leibler divergences can be parallelized in the KL-means algorithm. This makes the KL-means algorithm especially appealing for problems with a very large cardinality $|\mathcal{Y}|$ and a huge number of clusters $|\mathcal{T}|$.

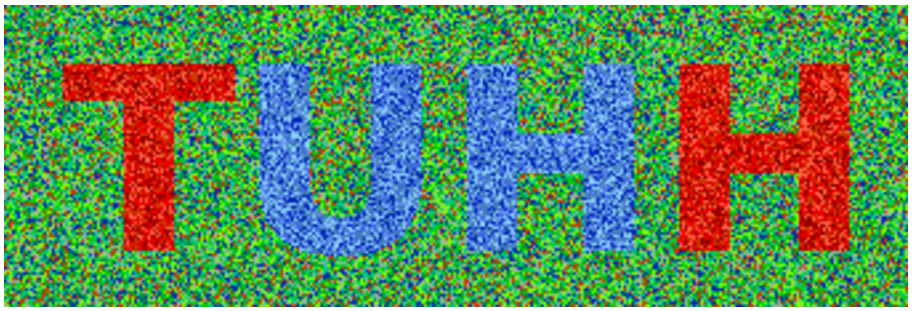
3.3 An Illustrative Example

Following all theoretical derivations and explanations included in this chapter, this section shall provide an easy to understand toy example for the application and the working principle of the Information Bottleneck method.

Consider the picture shown in Figure 3.5a which holds the letters TUHH. A significant property of this picture is that it only contains the three basic colors red, green and blue and no different shades of these colours. Suppose that the original picture from Figure 3.5a is shown on a faulty display, which results in the picture from Figure 3.5b instead of the original picture at a certain time instance. As it can be seen, the device introduces a lot of noise into the picture, such that the colors of the individual pixels seen on the display do not match their true colors in the original image.

It is noteworthy, that one is still able to read the letters TUHH in the noisy picture. Moreover, a human is able to guess which base colors the letters in the distorted picture have, despite they include shades of these colors which do not appear in the original image and a huge amount of pixels is shown faulty. So, obviously, the *information* on the colors of the letters and the

(a) Original image.



(b) Observed image at a particular time instance.

Figure 3.5: Original image and a noisy observation of the picture which suffers from pixel noise.

background is conserved in the noisy observation, but the noisy picture also contains some *irrelevant* features. A question arising is how one is able to extract only the relevant features from the observation. This example will show, that the Information Bottleneck method is a perfect tool for this task.

The first step for application of the Information Bottleneck method is defining which variable is relevant. In the scope of the thesis, this choice will sometimes be cumbersome, but in this example, the *true* color of the pixels as they appear in the original image is an intuitive choice. Therefore, we have the observed pixel colors $y \in \mathcal{Y}$ and the true pixel colors $x \in \mathcal{X}$ and we interprete both as realizations of random variables X and Y for each pixel. Please note that while the original image only contains the base colors red, green and blue, $\mathcal{X} = \{r, g, b\}$, $\mathcal{Y}$ contains much more distinct colors, $\mathcal{Y} = \{r_0, r_1, ..., g_0, g_1, ..., b_0, b_1, ...\}$ which are distinct shades of these base colors, such that $|\mathcal{Y}| >> |\mathcal{X}|$.

To apply the Information Bottleneck method, the first step is obtaining the joint probability distribution $p(x, y)$. Unfortunately, there is no general rule which allows for determination of this joint probability distribution for an arbitrary problem. One either needs a mathematical model which describes this distribution, or one needs to estimate it from realizations of X and Y. Also mixtures of both techniques are possible, such that some parameters of a model need to be estimated from data.

The probabilistic transition model $p(y|x)$ which led to the noisy images from the original image is illustrated in Figure 3.6. This model describes how the input pixels $x \in \{r, g, b\}$ shown on the left are transformed into the displayed pixels $y \in \mathcal{Y}$ on the right. For example, if a red input pixel appears, it is transformed into a shade of red r_0, r_1, r_2, r_3 or r_4 and all these transitions have a probability of 20%. Likewise, any blue input pixel b is transformed into a certain shade of blue. This essentially explains, why the red and the blue letters still appear as noisy red and blue letters in the noisy images. Something, however, is special about the green input pixels. Green input pixels are either transformed into shades of green g_0, g_1, g_2, g_3, g_4, such that at least their base color is preserved, or they are displayed entirely wrong as red pixels r_4 or blue pixels b_0.

In order to get $p(x, y)$ from the given transition model $p(y|x)$, one needs the distribution $p(x)$ of the relevant random variable X. This distribution describes the probabilites of red, green and blue pixels in the original image. It is illustrated on the left of Figure 3.6. A total of 13% of the pixels in the original image are red, 72% are green and 15% are blue. Knowing the transition model $p(y|x)$ and $p(x)$ one can finally determine $p(x, y) = p(y|x)p(x)$ for all pairs (x, y). For example, $\Pr(\mathsf{X} = g, \mathsf{Y} = b_0) = 0.125 \cdot 0.72 = 0.09$.

Knowing $p(x, y)$ one has all the ingredients for application of an Information Bottleneck algorithm. As it is illustrated in Figure 3.7, this joint distribution can be fed to one of the Information Bottleneck algorithms from Section 3.2. As shown in Figure 3.7, the only thing one is left with is choosing the desired output cardinality $|\mathcal{T}|$ of the compression variable T. In addition, some Information Bottleneck algorithms, for example, the iterative Information Bottleneck algorithm and the sequential Information Bottleneck algorithm allow to choose a finite value of β. From these inputs, the Information Bottleneck

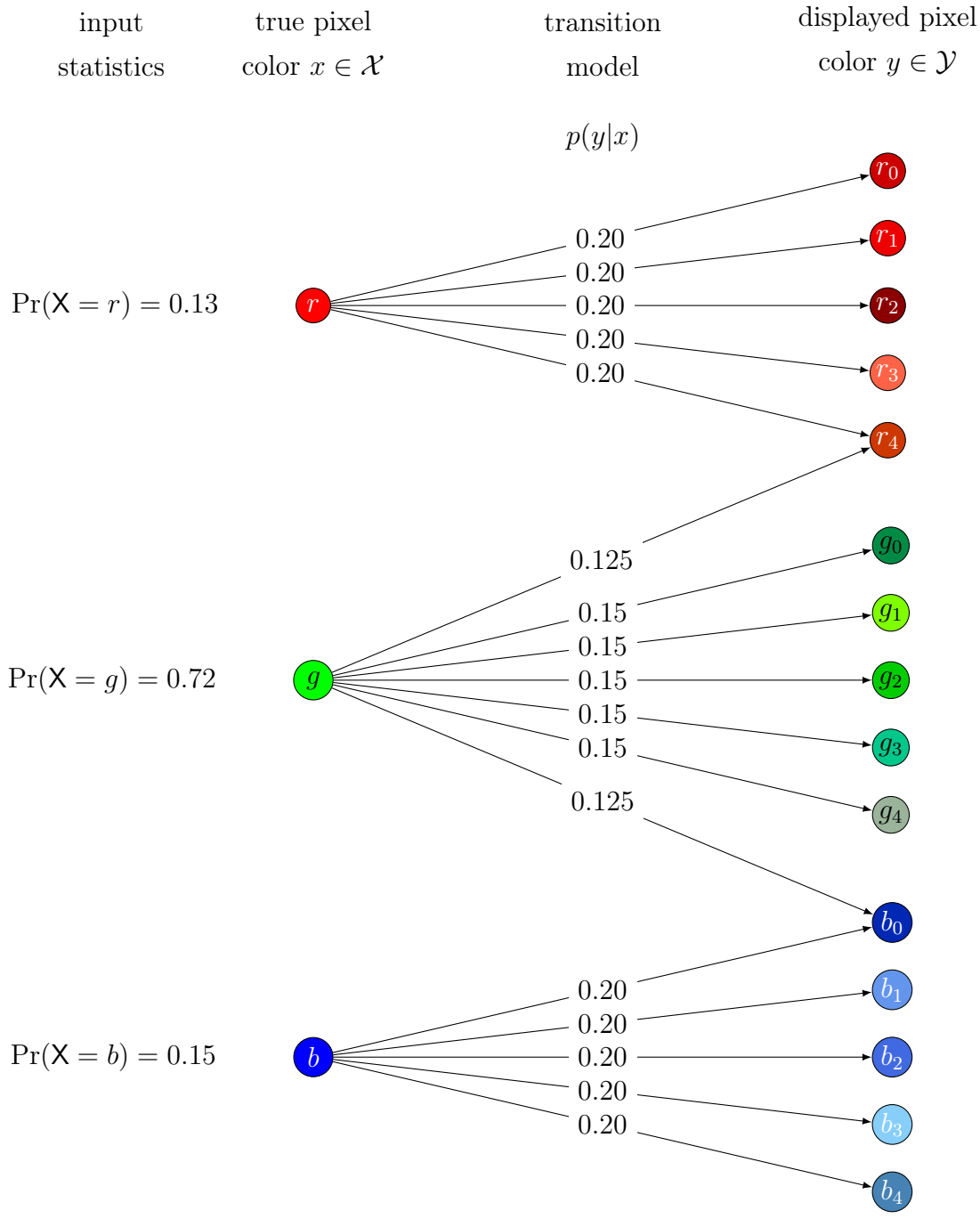


Figure 3.6: Illustration of the pixel color transition model and the statistics of an exemplary input picture.

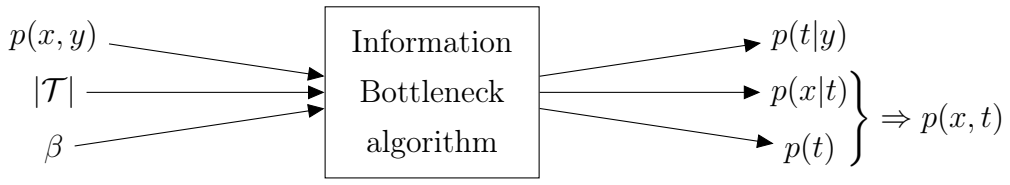


Figure 3.7: Inputs and outputs of an Information Bottleneck algorithm. The algorithm inputs $p(x, y)$, $\beta \geq 0$ and $|\mathcal{T}|$. The algorithm delivers the compression mapping $p(t|y)$. Distributions $p(x|t)$ and $p(t)$ follow from Equations (3.6), (3.7).

method delivers the compression mapping $p(t|y)$, the conditional distribution $p(x|t)$ and the distribution $p(t)$.

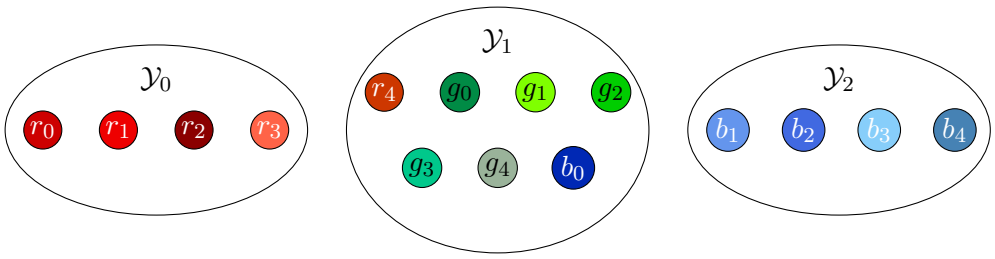
Figure 3.8 illustrates these outputs for the problem considered. For exemplary purposes, the sequential Information Bottleneck algorithm from Section 3.2.2 with an output cardinality $|\mathcal{T}| = 3$ and $\beta \to +\infty$ has been applied to obtain these results, but comparable results can also be obtained using the other introduced algorithms with the same parameters. Figure 3.8a visualizes the conditional distribution $p(t|y)$ obtained by the algorithm in its numerical representation and also the obtained hard partitioning of the event space $\mathcal{Y}$ into the clusters $\mathcal{Y}_0, \mathcal{Y}_1, \mathcal{Y}_2$. An eye catching fact about this illustration is that the Information Bottleneck algorithm clustered the red shades r_0, r_1, r_2, r_4 and the blue shades b_1, b_2, b_3, b_4 into seperate clusters which only hold shades of red or shades of blue, respectively. Moreover, all shades of green appear in cluster $\mathcal{Y}_1$ together with red shade r_4 and blue shade b_0.

The second output of the Information Botteneck algorithm is the conditional distribution $p(x|t)$. This distribution is plotted for all $t \in \mathcal{T}$ in Figure 3.8b. The event space $\mathcal{T}$ of the compression variable was delibaretely chosen to be a set of indices because the choice of the elements in $\mathcal{T}$ does not affect $\mathrm{I}(\mathsf{X};\mathsf{T})$.

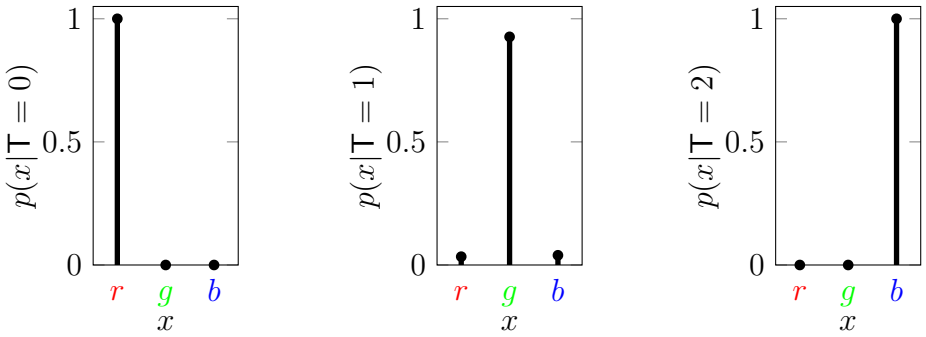
In order to understand how information is being preserved, one has to recall that $p(x|t)$ which also is a result from the Information Bottleneck algorithm provides a belief on the true color x for a given $t \in \mathcal{T}$. Analyzing $p(x|t)$, it immediately makes sense that the red shades r_0, r_1, r_2, r_3 lie in cluster $\mathcal{Y}_0$ which corresponds to the shown belief $p(x|\mathsf{T} = 0)$ because the belief on the true color X of the pixel is that a pixel appearing in one of these shades in the noisy image is red with probability 1. Reconsidering the underlying transition model

$\leftarrow$ all $y \in \mathcal{Y} \rightarrow$

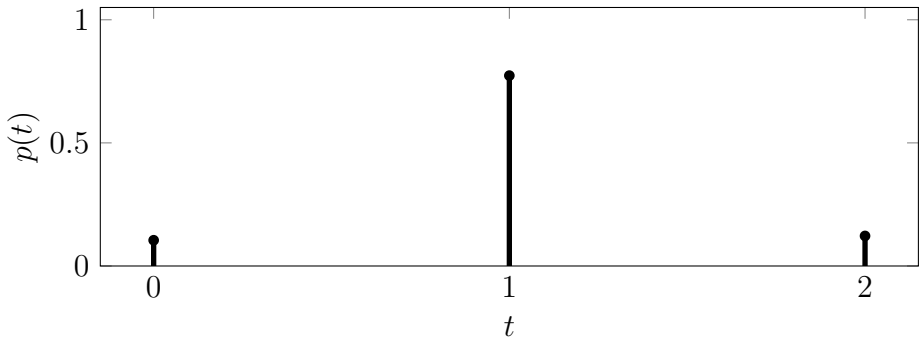
all $t \in \mathcal{T}$	$p(t\|y) \searrow$	r_0	r_1	r_2	r_3	r_4	g_0	g_1	g_2	g_3	g_4	b_0	b_1	b_2	b_3	b_4
	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1



(a) Output $p(t|y)$ of the sequential Information Bottleneck algorithm. The upper part of the illustration shows the numerical representation of $p(t|y)$ and the lower part shows the clustered event space of the observation.



(b) Output $p(x|t)$ of the sequential Information Bottleneck algorithm.



(c) Output $p(t)$ of the sequential Information Bottleneck algorithm.

Figure 3.8: Outputs of the sequential Information Bottleneck algorithm for $\beta \to +\infty, |\mathcal{T}| = 3$.

$p(y|x)$, this makes perfect sense because all pixels appearing in the shades in cluster $\mathcal{Y}_0$ in the noisy image *must* result from a red pixel in the original image. An identical explanation holds for the shades in cluster $\mathcal{Y}_2$ which must result from a blue pixel in the original image and, therefore, $p(x|\mathsf{T} = 2)$ expresses this absolute certainty with a probability 1 for $x = b$.

Cluster $\mathcal{Y}_1$, however, is special because it holds all shades of green and also r_4 and b_0. Recalling the transition model $p(y|x)$ from Figure 3.6 tells that r_4 and b_0 are the red and the blue shades that might be displayed in the wrong base color green. Hence, other than for the shades in $\mathcal{Y}_0$ and $\mathcal{Y}_2$, observing a shade from $\mathcal{Y}_1$ does not provide absolute certainty about the true pixel color $x \in \mathcal{X}$. Anyway, $p(x|\mathsf{T} = 1)$ provides a belief on the true color which shows that most probably (with a probabity of approximately 93%) the corresponding pixel is green, but also some small probabilities for red and blue exist which again perfectly fits to the underlying transition model $p(y|x)$.

The third output of the Information Bottleneck algorithm is $p(t)$ which is plotted in Figure 3.8c. This distribution tells the probabilitities that a certain cluster appears. Knowledge of $p(x, y)$, $p(x|t)$ and $p(t)$ allows to compare the mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ and $\mathrm{I}(\mathsf{X};\mathsf{T})$ to get an objective measure of how much relevant information is preserved when applying clustering $p(t|y)$ to the pixels from a noisy image. For this example, one obtains $\mathrm{I}(\mathsf{X};\mathsf{Y}) \approx 0.955$ bit and $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx 0.793$ bit, that is, 83% of the relevant information on the original pixels can be preserved when mapping the noisy pixels $y \in \mathcal{Y}$ to $t \in \{0, 1, 2\}$ according to the clustering shown in Figure 3.8a.

A particularly nice property of the three base colors r, g, b is that they can be mixed into a new color c by a linear superposition of their color codes in the rgb format. In the rgb format, each base color is represented as a number between 0 and 255 reflecting the intensity of the respective component. Thus, it is possible to assign a representative color c_t to each of the clusters $\mathcal{Y}_t$ in Figure 3.8a which corresponds to the expectation over the r, g, b components for a given t, that is,

$$c_t = \sum_{x \in \{r,g,b\}} x \cdot p(x|t). \tag{3.19}$$

Being given the conditional distribution $p(x|t)$ shown in Figure 3.8b, it is easy to see that for cluster $\mathcal{Y}_0$, this representative color is $c_0 = r$, for $\mathcal{Y}_1$ it is a linear

Table 3.2: Representative cluster colors c_t implied by $p(x|t)$ in rgb code.

cluster	r	g	b
$\mathcal{Y}_0$	255	0	0
$\mathcal{Y}_1$	9	236	10
$\mathcal{Y}_2$	0	0	255

Figure 3.9: Compressed image for $|\mathcal{T}| = 3$. The colors chosen for the pixels $y \in \mathcal{Y}$ mapped onto $t \in \mathcal{T}$ by $p(t|y)$ correspond to c_t from Table 3.2.

combination of r, g, b and for cluster $\mathcal{Y}_2$, it is $c_2 = b$. The corresponding rgb codes are provided in Table 3.2.

Please note that cluster $\mathcal{Y}_1$ is *not* represented by the base color green which has rgb code $(r, g, b) = (0, 255, 0)$, but instead a darker shade of green which suffers from some influence of red and blue. Drawing each pixel from the noisy picture in Figure 3.5b which is mapped onto an index $t \in \{0, 1, 2\}$ by the compression mapping $p(t|y)$ from the Information Bottleneck method in the respective representative color of the cluster yields the image shown in Figure 3.9. As it can clearly be seen, for the vast majority of the pixels, the compressed representation preserves the relevant information on the true base color of the pixels. Moreover, the noise influence is strongly weakened in the compressed representation. At the end of the next chapter we will revisit this example and it will be shown how the Information Bottleneck method can be used to obtain an even better reconstruction of the original image.

So far, this example has illustrated how the Information Bottleneck method can be applied to extract the relevant features of an observed random variable while compressing it to a more compact random variable with a smaller event space. We have seen, that the clustering of the event space of Y obtained by

an Information Bottleneck algorithm has to be considered together with the conditional probability distribution $p(x|t)$ to reobtain a belief on $x \in \mathcal{X}$ for a given $t \in \mathcal{T}$. In fact, $p(x|t)$ can be considered as the *meaning* of the appearance of a certain cluster index t with respect to the relevant x and, therefore, plays a crucial role.

In this example, it was inherently assumed that one aims for extraction of the maximum possible amount of relevant information for a given cardinality $|\mathcal{T}|$ and, therefore, $\beta \to +\infty$. The case $\beta \to +\infty$ in fact is the most interesting case for this thesis because typically our aim is building systems which preserve the maximum possible amount of relevant information for a given output cardinality $|\mathcal{T}|$. The output cardinality $|\mathcal{T}|$ directly determines how many bits are required to store $t \in \mathcal{T}$ in hardware. Therefore, throughout this thesis, we will typically assume that $\beta \to +\infty$ and aim for the smallest possible $|\mathcal{T}|$ which allows us to preserve a significant amount of relevant information.

A particularly useful feature of the Information Bottleneck method is that the general approach for its application is always identical and follows the methodology shown in Figure 3.7. This illustration can be understood as a recipe. Once one has defined which relevant feature X shall be preserved when compressing Y to T and one has access to the joint probability distribution $p(x, y)$, one is ready for application of the method. This generality enables a huge variety of applications in communication systems which will be explored in the following chapters.

3.4 Aspects of the Multivariate Information Bottleneck Method

So far, no assumption on the dimensionality of the involved random variables X and Y in the Information Bottleneck method has been made. An impressive generalization of the Information Bottleneck method exists which is termed the multivariate Information Bottleneck method [62]. It allows *all* involved random variables X, Y and T to be multivariate variables. Hence, it is able to handle several observations, several compression variables and several relevant variables and complex information preservation relations between all these variables. The multivariate Information Bottleneck method allows for a huge

variety of relations concerning the compression and the information preservation between the distinct variables involved. Moreover, it introduces a huge variety of different multivariate Information Bottleneck algorithms. In [62] the authors use so called Bayesian networks and the concept of multi-information to describe the information relations between observed, relevant and compression variables in such multivariate settings. The multivariate Information Bottleneck requires the selection of a Bayesian input network which defines the compression relations between the random variables involved. It, furthermore, requires to define a Bayesian output network to describe their intended information preservation. This approach is tremendously generic.

A simple case of particular interest for this thesis is the case where the observed random variable Y is multivariate, that is, $\mathsf{Y} = (\mathsf{Y}_0, \mathsf{Y}_1, \ldots, \mathsf{Y}_{N-1})$, but the relevant and the compression variable are one dimensional. While this is strictly speaking a case of a relaxed multivariate Information Bottleneck problem, it turns out that the Information Bottleneck method and the processing of the Information Bottleneck algorithms are actually not different in this case at all. The only change in this case is that the elements y from the event space of the observation Y are tuples. Therefore, the compression mapping $p(t|y)$ becomes conditioned on all scalar variables in such tuples, for example, $p(t|y_0, y_1, \ldots, y_{N-1})$. The method, however, works identically. Anyway, the minor change can have a huge impact on the cardinality of the event space of the observed random variable. Assume that each component Y_n from the multivariate observation has event space $\mathcal{Y}_n$. Each component in a realization $y = (y_0, y_1, \ldots, y_{N-1})$ then can take $|\mathcal{Y}_n|$ different values. Therefore, there exists a total of $|\mathcal{Y}_0| \cdot |\mathcal{Y}_1| \cdot \ldots \cdot |\mathcal{Y}_{N-1}|$ possible tuples y.

Consider, for example, the case where all Y_n can take 16 different values and $N = 10$. In this case there are $16^{10} \approx 1.1 \cdot 10^{12}$ different realizations y. Recall that the number of different realizations y is the number of entries in a lookup table implementation of a deterministic compression mapping $p(t|y)$. Thus, storing such a compression mapping has an enormous space complexity. Moreover, the cardinality of the event space of Y has a huge impact on the algorithmic complexity of the Information Bottleneck algorithms described in Section 3.2. The sequential Information Botteneck algorithm from Section

3.2.2 has to process all elements in $\mathcal{Y}$ sequentially and, therefore, has linear time complexity in $|\mathcal{Y}|$.

From the above, one can draw the conclusion that application of the Information Bottleneck method is feasible for multivariate Y. However, challenges arise for multivariate Y because multivariate random variables might have very large event spaces.

Chapter 4

Information Bottleneck Graphs

Communication engineers traditionally use graphical models to visualize complicated signal processing problems and to break them down into smaller subproblems handled by concatenated blocks in signal processing chains. Factor graphs described in Section 2.4 are one very powerful tool for this process. This section introduces Information Bottleneck graphs. Information Bottleneck graphs are extended factor graphs which have been developed in the scope of this thesis and were first presented in [63]. They have also been used by other authors in [20, 21].

The fundamental idea of Information Bottleneck graphs is to describe the information relation between the variables involved in the Information Bottleneck method compactly in a factor graph. Information Bottleneck graphs are very helpful when the number of variables involved in an Information Bottleneck problem becomes large and when several Information Bottleneck compression mappings shall be applied in a concatenated scheme. This is required to build complex signal processing schemes which typically consist of subsequent blocks processing outputs from other blocks in a signal processing chain.

Information Bottleneck graphs enable a very intuitive and information centric approach to the design of such complex structures. As extended factor graphs, they moreover inherit the possibility of hierarchical modelling from factor graphs. In the following, Information Bottleneck graphs will first be introduced as a graph based model for arbitrary compression mappings $p(t|y)$ designed with the Information Bottleneck method. Due to the simplicity of the proposed concept, this introduction of Information Bottleneck graphs can stay very short. A simple Information Bottleneck graph will then be used to clarify their usage and usefulness. Moreover, the focus will be laid on hierarchical modelling using Information Bottleneck graphs by opening and closing factor nodes.

Towards the end of the chapter another illustrative example for the application of Information Bottleneck graphs will be provided. This example shows how Information Bottleneck graphs can be used to design a flow of relevant information. More applications of Information Bottleneck graphs focussing on communication systems are deferred to Chapter 5, where they will be used extensively. Finally, some concluding remarks on Information Bottleneck graphs and their usage end this chapter.

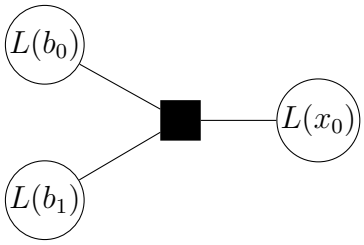
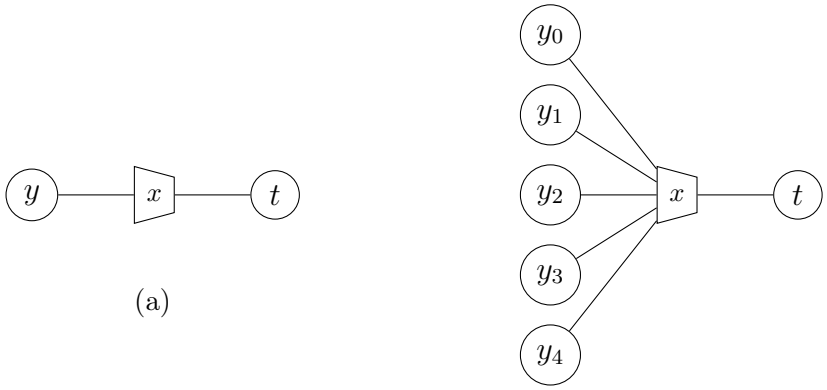


Figure 4.1: Factor graph model of a box-plus operation. The shown node is factor $p(L(x_0)|L(b_0), L(b_1))$ which implies the deterministic input $\rightarrow$ output connection from Equation (2.52).

4.1 A Factor Graph Model for Information Bottleneck Compression Mappings

A natural view on signal processing devices is an input $\rightarrow$ output perspective. A signal processing device has one or several inputs and calculates outputs which are processed by subsequent signal processing entities. Factor graphs from Section 2.4 are a very powerful framework for modeling of signal processing chains. Consider, for example, the elementary box-plus operation from Equation (2.52). Figure 4.1 shows a factor graph of a factor $p(L(x_0)|L(b_0), L(b_1))$ which performs the box-plus operation $L(x_0) = L(b_0) \boxplus L(b_1)$. The deterministic rule to calculate $L(x_0)$ from $L(b_0)$ and $L(b_1)$ given by Equation (2.52) is described by the factor $p(L(x_0)|L(b_0), L(b_1)) = \delta(L(x_0) - L(b_0) \boxplus L(b_1))$ in the factor graph.

Just as the box-plus operation from this example, compression mappings designed with the Information Bottleneck method process one or several inputs and deliver an output. Therefore, Information Bottleneck compression mappings can also be seen as signal processing entitities which are designed to preserve relevant information on a variable of interest under compression. For a desired maximum amount of preserved relevant information which corresponds to the case where $\beta \rightarrow +\infty$, these signal processing entities can be implemented in lookup tables, as it has been explained in Section 3.1.3. The aim of Information Bottleneck graphs is now to extend factor graphs by a visualization of the flow of relevant information between the variables involved which results from application of the Information Bottleneck method. For this purpose the following notation is introduced:



(a)

(b)

Figure 4.2: Exemplary Information Bottleneck graphs of $p(t|y)$ (a) and $p(t|y_0, y_1, y_2, y_3, y_4)$ (b). Information Bottleneck graphs use trapezoid nodes labeled by the relevant variable for compression mappings designed with the Information Bottleneck method. The compression variable is connected to the shortest side of a trapezoid node.

1. Factor nodes representing compression mappings which have been designed with an Information Bottleneck algorithm use a trapezoid symbol.

2. These factor nodes are labeled by their respective chosen relevant random variable x.

3. The compression random variable is always connected to the shortest side of the trapezoid.

4. All other variables connected to a trapezoid node represent the (possibly multivariate) observation.

Examples are given in Figure 4.2. Figure 4.2a shows an Information Bottleneck graph of $p(t|y)$ and Figure 4.2b exemplarily shows a compression mapping $p(t|y_0, y_1, y_2, y_3, y_4)$ with five inputs. The additional information in comparison to a regular factor graph is that the illustrations include a very compact description of the intended information and compression relations between the variables involved. The introduced notation inherently describes that the depicted compression mappings are directed components which aim to maximize the relevant information between their output t and the relevant x, that is,

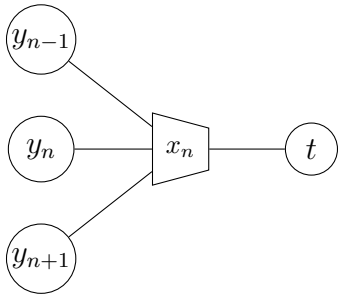
$I(\mathsf{X};\mathsf{T}) \rightarrow \max$. Moreover, it describes the bottleneck situation by visualizing that a realization of Y has to be pressed through a bottleneck with respect to the preservation of relevant information on X and that the output of this operation is a realization of the compression random variable T. Therefore, the introduced notation describes the information relations of the variables involved in the Information Bottleneck framework adequately.

Please note that also the joint distribution required for designing the shown node can be recognized from the Information Bottleneck graph which for any node is given by the joint distribution of the variable labeling the node and all its connected observations. Compression mappings appearing in Information Bottleneck graphs will be called *Information Bottleneck nodes* from now on.

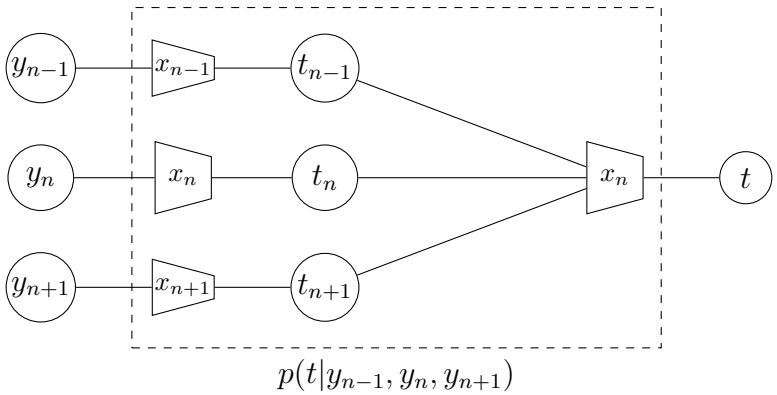
Before one can start to use Information Bottleneck graphs, a very important statement on optimality has to be made. As it has already been mentioned in Chapter 3, solving the Information Bottleneck problem for $\beta \rightarrow +\infty$ is a problem of convex maximization and the available algorithms can normally only find locally optimum solutions. As a consequence, *instead of* perhaps naturally defining that a compression mapping $p(t|y)$ described in an Information Bottleneck graph is a global optimum compression mapping which maximizes $I(\mathsf{X};\mathsf{T})$ under a constraint for the cardinality of T globally, it has only been assumed so far, that the respective compression mapping has been designed using an Information Bottleneck algorithm in order to preserve a desired huge amount of relevant information. Formally, it is only required for the concept of an Information Bottleneck graph that the preserved relevant information $I(\mathsf{X};\mathsf{T}) > 0$, such that the output random variable is informative about the respective relevant random variable which labels the node. This will be important for the ability of hierarchical modelling by opening and closing nodes in Information Bottleneck graphs which will be investigated soon. However, one should always inherently assume that one aims to achieve a desired maximum of preserved relevant information for a given output cardinality $|\mathcal{T}|$.

4.2 Usage of Information Bottleneck Graphs

Information Bottleneck graphs can be used to easily model and design complex systems by considering a *flow of relevant information* between concatenated In-



(a) Information Bottleneck graph of $p(t|y_{n-1}, y_n, y_{n+1})$.



(b) Information Bottleneck graph of $p(t|y_{n-1}, y_n, y_{n+1})$ with opened node and intermediate variables t_{n-1}, t_n, t_{n+1}.

Figure 4.3: Several Information Bottleneck graphs for compression of three pixels y_{n-1}, y_n, y_{n+1} without and with opened nodes.

formation Bottleneck nodes. The usefulness of Information Bottleneck graphs in this context can be illustrated best using an example. Therefore, we reuse the illustrative example on the Information Bottleneck method from Section 3.3.

In this example, a noisy pixel y has been compressed using a deterministic compression mapping $p(t|y)$ which was designed to preserve information on the original pixel color x. Please note that the Information Bottleneck graph of $p(t|y)$ for this setup is the one shown in Figure 4.2a. So far, in Section 3.3 compression of only one pixel at a time has been considered.

A quite natural extension of this approach is also trying to utilize the spatial correlation of neighbored pixels in the image for the extraction of relevant information. For simplicity, consider only three vertically neighbored pixels

(y_{n-1}, y_n, y_{n+1}) in a noisy image, where index n is the vertical position in the image. Based on the observation of these noisy three pixels, one aims to extract the information on the pixel x_n in the middle of these pixels with the Information Bottleneck method.

A straightforward approach for this is depicted in Figure 4.3a in an Information Bottleneck graph. The figure illustrates, that the pixels (y_{n-1}, y_n, y_{n+1}) shall be compressed together and that the output t shall be highly informative about the true pixel color x_n of the pixel in the middle. This very simple example already illustrates that Information Bottleneck graphs can describe the information relation between the variables involved very compactly and adequately.

The Information Bottleneck node $p(t|y_{n-1}, y_n, y_{n+1})$ shown in Figure 4.3a has to be designed by feeding joint distribution $p(x_n, y_{n-1}, y_n, y_{n+1})$ to an Information Bottleneck algorithm. This joint distribution can be calculated as

$$p(x_n, y_{n-1}, y_n, y_{n+1}) = \sum_{x_{n-1}\in\mathcal{X}_{n-1}} \sum_{x_{n+1}\in\mathcal{X}_{n+1}} p(x_{n-1}, x_n, x_{n+1}, y_{n-1}, y_n, y_{n+1}) =$$
$$\sum_{x_{n-1}\in\mathcal{X}_{n-1}} \sum_{x_{n+1}\in\mathcal{X}_{n+1}} p(y_{n-1}|x_{n-1})p(y_n|x_n)p(y_{n+1}|x_{n+1})p(x_{n-1}, x_n, x_{n+1}), \quad (4.1)$$

where the joint distribution $p(x_{n-1}, x_n, x_{n+1})$ describes the spatial dependencies of three vertically neighbored pixels in the original image. While this approach is straightforward, it suffers from the disadvantage that the lookup table implementation of $p(t|y_{n-1}, y_n, y_{n+1})$ has exponential complexity in the number of inputs. In this example, the lookup table requires $15^3 = 3375$ entries because according to the transition model shown in Figure 3.6, each pixel y_m, $m \in \{n-1, n, n+1\}$ can take 15 different colors.

However, going back to the Information Bottleneck example from Section 3.3 one can consider an alternative compression approach which is depicted in Figure 4.3b. The idea is to also include the Information Bottleneck compression mapping $p(t|y)$ designed in Section 3.3 in the Information Bottleneck graph. This compression mapping can be applied to every pixel y_m, $m \in \{n-1, n, n+1\}$ to compress each pixel to a compressed t_m which is informative about the true color x_m of this pixel. Like this, the mapping gets used for all three input pixels $m \in \{n-1, n, n+1\}$, resulting in the Information Bottleneck nodes $p(t_m|y_m)$, $m \in \{n-1, n, n+1\}$ shown on the left of Figure 4.3b.

Afterwards, one can feed the compressed t_{n-1}, t_n, t_{n+1} to an Information Bottleneck node $p(t|t_{n-1}, t_n, t_{n+1})$ which is designed to preserve the information on x_n by making use of the spatial correlation of the pixels as it is also illustrated in Figure 4.3b. An advantage of this approach is that implementation of the Information Bottleneck node $p(t|t_{n-1}, t_n, t_{n+1})$ offers the possibility to lower the required number of lookup table entries because the cardinality of the input variables of table $p(t|t_{n-1}, t_n, t_{n+1})$ is now $|\mathcal{T}_m|$ instead of $|\mathcal{Y}_m|$. For example, with $|\mathcal{T}_m| = 5\,\forall m \in \{n-1, n, n+1\}$, this Information Bottleneck node only requires $5^3 = 125$ lookup table entries. The additional lookup tables $p(t_m|y_m)$ which have been adapted from Section 3.3 require $|\mathcal{Y}_m| = 15$ entries. Moreover, all $p(t_m|y_m)$, $m \in \{n-1, n, n+1\}$ are identical. Hence the total number of lookup table entries to implement the Information Bottleneck graph from Figure 4.3b with $|\mathcal{T}_m| = 5$ is $15 + 125 = 140 << 3375$ which is a drastical reduction in comparison to the straightforward approach from Figure 4.3a.

Essentially, to come from the Information Bottleneck graph in Figure 4.3a to the one in Figure 4.3b, novel intermediate variables t_{n-1}, t_n, t_{n+1} and additional Information Bottleneck nodes have been introduced. Considering how the joint distribution $p(x_n, t_{n-1}, t_n, t_{n+1})$ which is required to design the Information Bottleneck node $p(t|t_{n-1}, t_n, t_{n+1})$ is calculated reveals an interesting fact. This distribution is given by

$$p(x_n, t_{n-1}, t_n, t_{n+1}) = \sum_{x_{n-1} \in \mathcal{X}_{n-1}} \sum_{x_{n+1} \in \mathcal{X}_{n+1}} p(x_{n-1}, x_n, x_{n+1}, t_{n-1}, t_n, t_{n+1}) =$$
$$\sum_{x_{n-1} \in \mathcal{X}_{n-1}} \sum_{x_{n+1} \in \mathcal{X}_{n+1}} p(t_{n-1}|x_{n-1}) p(t_n|x_n) p(t_{n+1}|x_{n+1}) p(x_{n-1}, x_n, x_{n+1}), \quad (4.2)$$

where the distributions $p(t_m|x_m)$, $m \in \{n-1, n, n+1\}$ are given by

$$p(t_m|x_m) = \frac{1}{p(x_m)} p(x_m|t_m) p(t_m). \quad (4.3)$$

Interestingly, the joint distribution $p(x_n, t_{n-1}, t_n, t_{n+1})$ required to design the Information Bottleneck node $p(t|t_{n-1}, t_n, t_{n+1})$ depends on the outputs $p(x_m|t_m)$ and $p(t_m)$ of the Information Bottleneck algorithm used to design the Information Bottleneck nodes $p(t_m|y_m)$ in Figure 4.3b. Therefore, to design the concatenated compression scheme depicted in Figure 4.3b, one first has to design the nodes $p(t_m|y_m)$ as it has been explained in Section 3.3. The side products $p(x_m|t_m)$ and $p(t_m)$ of the Information Bottleneck algorithm applied

to design $p(t_m|y_m)$ are afterwards required to design the next concatenated Information Bottleneck node $p(t|t_{n-1}, t_n, t_{n+1})$.

This can be interpreted to be similar to the process of density evolution analyzed in Section 2.3.4, where probability density functions were passed between processing units. More precisely, the described process describes the evolution of joint probability distributions used to design the concatenated Information Bottleneck nodes by making use of the output distributions of the applied Information Bottleneck algorithms.

4.3 Opening and Closing Nodes in Information Bottleneck Graphs

Information Bottleneck graphs are extended factor graphs of probability distributions. As a consequence, they inherit the valuable ability to open and to close nodes explained in Section 2.4.2 from these factor graphs. The transition from the Information Bottleneck graph in Figure 4.3a to the one in Figure 4.3b essentially is identical to the process of considering a possible internal structure of Information Bottleneck node $p(t|y_{n-1}, y_n, y_{n+1})$. Therefore, the dashed box in Figure 4.3b is labeled $p(t|y_{n-1}, y_n, y_{n+1})$ and this box depicts the respective *opened* Information Bottleneck node. The notation used is also formally correct according to the rules for opening and closing nodes in factor graphs which have been introduced in Section 2.4.2. Taking the product of all factors in Figure 4.3b yields

$$p(t_{n-1}|y_{n-1})p(t_n|y_n)p(t_{n+1}|y_{n+1})p(t|t_{n-1}, t_n, t_{n+1}) = \\ p(t, t_{n-1}, t_n, t_{n+1}|y_{n-1}, y_n, y_{n+1}). \quad (4.4)$$

Following Section 2.4.2 one has to sum over all variables inside an opened node to close it again. Doing so results in

$$\sum_{t_{n-1}\in\mathcal{T}_{n-1}} \sum_{t_n\in\mathcal{T}_n} \sum_{t_{n+1}\in\mathcal{T}_{n+1}} p(t, t_{n-1}, t_n, t_{n+1}|y_{n-1}, y_n, y_{n+1}) = \\ p(t|y_{n-1}, y_n, y_{n+1}) \quad (4.5)$$

and in fact yields the original Information Bottleneck node $p(t|y_{n-1}, y_n, y_{n+1})$ depicted in Figure 4.3a. It is intuitive that when closing the Information

Bottleneck node, the respective label of the closed Information Bottleneck node has to be taken from the Information Bottleneck node which is connected to the output t in the underlying opened node. After the node has been closed, the internal structure of the node is not visible anymore. Since a closed node internally consists of several concatenated lossy compression mappings, one cannot conclude optimality in the sense that $\mathrm{I}(\mathsf{X};\mathsf{T}) \to \max$ for every close node labeled with x. However, one will typically aim for a huge amount of this preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$. The loosest mathematical requirement which describes the intended information relation is that $\mathrm{I}(\mathsf{X};\mathsf{T}) > 0$. This is the reason why this was the only requirement introduced on the preservation of relevant information in an Information Bottleneck graph. Of course, this raises the question, when a designed opened node structure can be considered to be well designed.

The ultimate benchmark for any factor node $p(t|y)$ labeled x is a globally optimum compression mapping $p(t|y)$ which maximizes $\mathrm{I}(\mathsf{X};\mathsf{T})$ for a given cardinality of T. Unfortunately, it is unknown how this node can be found in general due to the convex maximization nature of the underlying optimization problem. Moreover, for multivariate Y, a straightforward implementation of the closed node is often prohibitive due to the huge number of possible input realizations. Anyway, it turns out that it is possible to design quite meaningful *flows of relevant information* using Information Bottleneck nodes which are implementable in practice and preserve a significant amount of relevant information.

Please note that already in Figure 4.3b *not* all nodes are labeled with x_n, but the closed node should deliver an output which is informative about this quantity. In contrast, following the nodes in Figure 4.3b from the left to the right reveals that in the first stages $p(t_m|y_m)$, $m \in \{n-1, n, n+1\}$ relevant information on the true color x_m of the individual pixels shall be preserved. This was motivated by reusing the compression mappings designed in Section 3.3.

In the next stage given by $p(t|t_{n-1}, t_n, t_{n+1})$ this information is then aggregated to information on the pixel color x_n which is of interest by making use of the spatial correlation of the pixels. This can be considered to be a logical *flow of relevant information* from the left to the right in the depicted graph.

However, also other meaningful flows are thinkable. One could, for example, label all nodes $p(t_m|y_m)$, $m \in \{n-1, n, n+1\}$ with x_n and design these nodes with an Information Bottleneck algorithm accordingly to already make use of the spatial correlation of the pixels in the first step of compression. The required input distribution $p(x_n, y_m)$ for any $m \in \{n-1, n, n+1\}$ could be obtained by a proper marginalization of $p(x_n, y_{n-1}, y_n, y_{n+1})$ from Equation (4.1). Anyway, if the node $p(t|y_{n-1}, y_n, y_{n+1})$ designed like this was closed, it would look identical to the node shown in Figure 4.3a. Thus, by closing the node, despite loosing some information on the internal structure of the node, we get a compact model which still describes the intended information relations between the variables involved. Hence, when designing an opened node structure in an Information Bottleneck graph, it is part of the task to consider a meaningful *flow of relevant information*, such that in the end, the output of the node guarantees $\mathrm{I}(\mathsf{X};\mathsf{T}) > 0$. However, despite one will typically aim for it, one cannot conclude that $\mathrm{I}(\mathsf{X};\mathsf{T}) \to \max$ for every closed node.

4.4 Aspects of Designing a Flow of Relevant Information

The above considerations enable to summarize and conclude several aspects for the design of Information Bottleneck nodes. First of all, the general approach to design an Information Bottleneck node $p(t|y)$ including also multivariate observations $\mathsf{Y} = (\mathsf{Y}_0, \mathsf{Y}_1, \ldots, \mathsf{Y}_{N-1})$ can be split into the following subsequent steps.

1. Identify all available observations to be processed by the node and determine the relevant random variable of interest.

2. Draw a closed Information Bottleneck node which aims to extract the relevant information from all the available observations.

3. If straightforward design and implementation of the closed node is too complex, open it and try to find a decomposition which is feasible by considering a meaningful flow of relevant information in the direction of the output variable.

4. Design all appearing Information Bottleneck nodes locally using an Information Bottleneck algorithm. Concatenated nodes require processing the output distributions $p(x|t)$ and $p(t)$ of the ancestor Information Bottleneck algorithms.

Moreover, it has been discussed that the choice of the cardinalities of the compression variables influences the number of total entries required to implement an opened Information Bottleneck node in a lookup table. From this perspective, it seems reasonable to choose the cardinalities of the compression variables in a concatenated scheme as small as possible because this will result in small lookup tables.

However, the cardinality of the compression variable also strongly interacts with the amount of preserved relevant information. Choosing it to small, typically looses preserved relevant information. Therefore, the choice of appropriate cardinalities of the compression variables leaves the designer of an Information Bottleneck graph with many degrees of freedom. Essentially, each and every single involved Information Bottleneck node in an opened node structure could use a different output cardinality.

Often even more degrees of freedom are given by the actual choice of the structure of an opened Information Bottleneck node which also has the potential to influence the amount of preservable relevant information. On the one hand, it is intuitive that concatenating lossy compression mappings tends to loose some relevant information due to the inherent bottleneck behaviour of all nodes involved. On the other hand, we have seen that it is even possible to choose different relevant random variables for concatenated Information Bottleneck nodes in an opened node structure and, therefore, it is highly nontrivial to forecast how a certain decomposition with a given choice of local relevant variables will affect the preserved relevant information of the closed node. These facts make it very hard to make a statement on the best choice of a decomposition. Despite this fact, it will be shown in many practical examples in the remainder of the thesis that using the proposed methodology of considering and carefully designing a meaningful flow of relevant information in the direction of a node output can yield very useful Information Bottleneck nodes for signal processing. In the end, they can describe entire receiver chains, as it will be shown in Chapter 5.

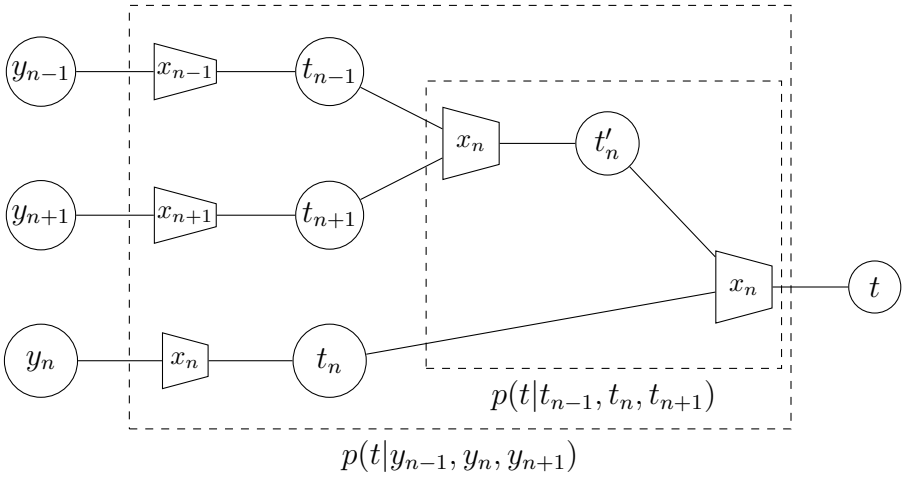


Figure 4.4: Information Bottleneck graph of $p(t|y_{n-1}, y_n, y_{n+1})$ with two opened nodes and intermediate variables $t_{n-1}, t_n, t_{n+1}, t'_n$.

4.5 An Illustrative Example

One can consider several ways to further open the node $p(t|t_{n-1}, t_n, t_{n+1})$ from Figure 4.3b. One example to do so is shown in Figure 4.4. There the node $p(t|t_{n-1}, t_n, t_{n+1})$ is broken up into the nodes $p(t'_n|t_{n-1}, t_{n+1})$ and $p(t|t'_n, t_n)$ internally. In this section it shall be illustrated how the opened node structure shown in Figure 4.4 is designed and how it generates a flow of relevant information. For this purpose, it will be explained how to obtain all joint distributions required to design the depicted Information Bottleneck nodes. Afterwards, it is visualized how the concatenated Information Bottleneck nodes extract relevant information by analyzing the compressed images at the output of each depicted Information Bottleneck node.

The first step for construction of the depicted Information Bottleneck nodes is obtaining the joint distributions connecting the relevant and the observed random variables for each node. These joint distributions are given by

$$p(x_m, y_m) = p(y_m|x_m)p(x_m),\ m \in \{n-1, n, n+1\}, \tag{4.6}$$

$$p(x_n, t'_n, t_n) = p(t'_n|x_n)p(t_n|x_n)p(x_n) = p(t'_n|x_n)p(x_n, t_n) \tag{4.7}$$

and

$$p(x_n, t_{n-1}, t_{n+1}) = \sum_{x_{n-1} \in \mathcal{X}_{n-1}} \sum_{x_{n+1} \in \mathcal{X}_{n+1}} p(t_{n-1}|x_{n-1}) p(t_{n+1}|x_{n+1}) p(x_{n-1}, x_n, x_{n+1}). \quad (4.8)$$

Please note that these joint distributions have some inherent dependencies. The joint distributions required to design the concatenated nodes depend on the respective output distributions of the Information Bottleneck algorithms applied to design their respective preceding nodes. Therefore, one has to design the nodes subsequently from the left to the right in Figure 4.4.

To start doing so, one moreover has to choose the cardinalities of the compression variables. In this example, all Information Bottleneck nodes shall be designed with an identical output cardinality $|\mathcal{T}| = 5$ for simplicity. The chosen configuration results in a total of 65 lookup table entries. Since $|\mathcal{Y}_m| = 15 \, \forall m \in \{n-1, n, n+1\}$, 15 table entries are required for each $p(t_m|y_m)$, $m \in \{n-1, n, n+1\}$. However, these tables are identical due to their identical design distributions. The Information Bottleneck nodes $p(t'_n|t_{n-1}, t_{n+1})$ and $p(t|t'_n, t_n)$ both require 25 entries because both of their inputs can take 5 different values.

Finally, Information Bottleneck algorithms to design the Information Bottleneck nodes have to be selected. Here the KL-means Information Bottleneck algorithm from Section 3.2.3 is applied for all Information Bottleneck nodes. It delivers the compression mappings $p(t_m|y_m)$, $m \in \{n-1, n, n+1\}$, $p(t'_n|t_{n-1}, t_{n+1})$ and $p(t|t'_n, t_n)$ which can now be applied to compress a noisy input image.

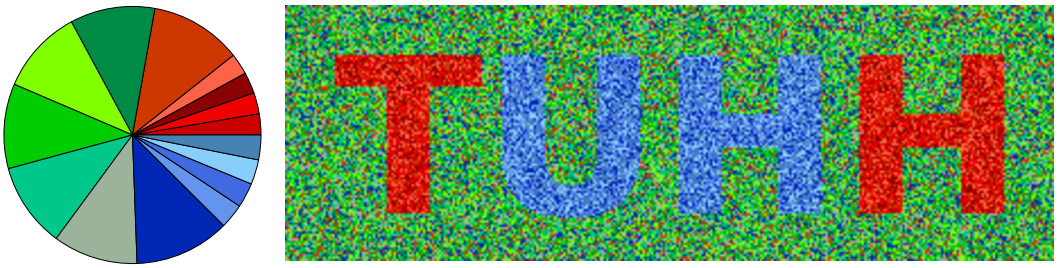
The processing of the noisy input image with the designed Information Bottleneck node is performed as follows. Starting at the top left corner of the image in pixel row $n = 1$, the node processes three vertically neighbored input pixels $y_m, m \in \{n-1, n, n+1\}$ at a time. From these pixels, first t_{n-1}, t_{n+1} and t_n are determined. Afterwards, t'_n and t are determined. The realizations of $t_{n-1}, t_n, t_{n+1}, t'_n$ and t are stored together with the position of their corresponding input pixels (i.e., the row index n and the respective column index which is skipped in the notation for convenience). This can easily be done by storing the respective realizations in matrices of the size of the original image. Then the three pixels from the next pixel column in row $n = 1$ are processed in

the same manner. When the right border of the image is reached, one moves on to the next pixel row $n+1$ and performs the same steps until the entire image has been processed.

The side products $p(x_n|t_n)$, $p(x_n|t'_n)$ and $p(x_n|t)$ of the Information Bottleneck algorithms applied to design the nodes $p(t_n|x_n), p(t'_n|t_{n-1}, t_{n+1})$ and $p(t|t'_n, t_n)$ then allow to to visualize the output images of each node in the concatenated scheme using rgb colors. Figure 4.5 illustrates the development of the compressed images for the mentioned concatenated Information Bottleneck nodes. All images in this figure have been amended by a pie diagram of their included colors which is depicted on the left of each respective figure. Figure 4.5a shows the noisy input image of the Information Bottleneck graph. The corresponding pie diagram clearly shows that all 15 color shades present on the right of the transition model from Figure 3.6 appear in the noisy input picture. For the compressed images depicted in Figures 4.5b, 4.5c and 4.5d, the colors not being present in the original image are separated in the pie diagrams to provide an intuitive measure about the remaining distortion in these pictures. In the following, the evolution of the compressed images at the outputs of the Information Bottleneck nodes from Figure 4.4 will be discussed.

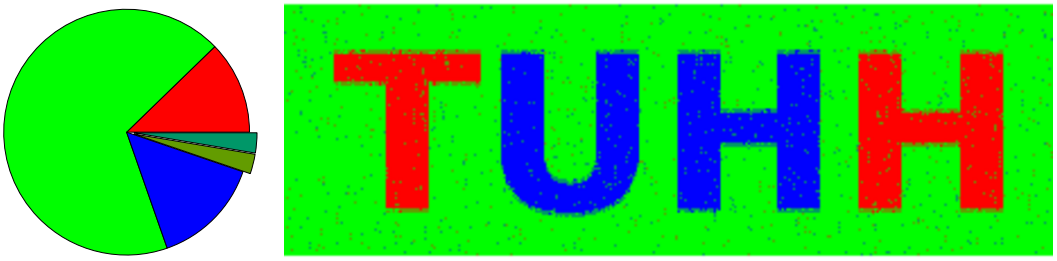
The compressed image after pixelwise application of the compression mappings $p(t_m|y_m)\, m \in \{n-1, n, n+1\}$ for all appearing pixels, is shown in Figure 4.5b for $m = n$ exemplarily. This figure compares to the compressed images from Section 3.3 with the only difference, that the compression cardinality was chosen larger than in Section 3.3 ($|\mathcal{T}| = 5$ instead of $|\mathcal{T}| = 3$). It reveals that application of the first compression mapping preserves some relevant information on the original color of each particular pixel. However, it also shows that a huge amount of pixels located inside the letters still suffers from a strong influence of false green pixels. Moreover, the background also still suffers from noise, such that a huge amount of the pixels in the background appears in a darker green shade instead of the true base color green. This influence is also clearly evident in the corresponding pie diagram.

Figure 4.5c shows the compressed image at the output of Information Bottleneck node $p(t'_n|t_{n-1}, t_{n+1})$. A comparison to Figure 4.5b shows that the influence of false dark green pixels is strongly reduced in this image by making use of the spatial correlation of the pixels. This effect is visible especially

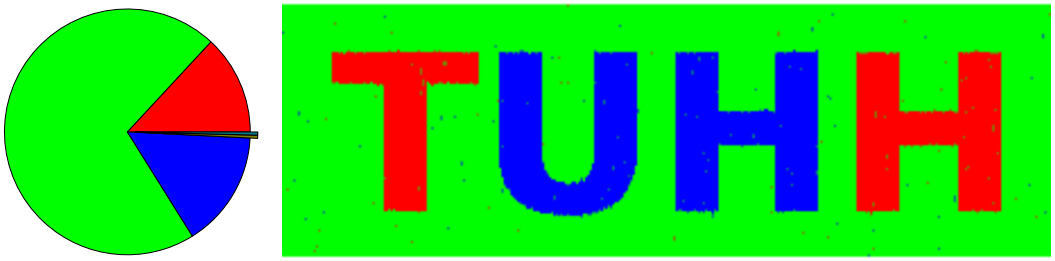


(a) Color statistics and noisy input image of the Information Bottleneck graph.

(b) Color statistics and compressed image after application of $p(t_n|y_n)$.



(c) Color statistics and compressed image after application of $p(t'_n|t_{n-1}, t_{n+1})$.



(d) Color statistics and compressed image after application of $p(t|t_n, t'_n)$.

Figure 4.5: Illustration of the development of the compressed images in the Information Bottleneck graph from Figure 4.4. All output cardinalities of the applied Information Bottleneck algorithms were set to $|\mathcal{T}| = 5$.

inside the letters. However, the horizontal edges of the letters still suffer from mentionable noise which appears in dark green shades. Anyway, comparing the pie diagrams between Figures 4.5b and Figure 4.5c shows that the colors not present in the original image have been suppressed further by Information Bottleneck node $p(t'|t_{n-1}, t_{n+1})$.

Finally, Figure 4.5d shows the compressed image at the output of the Information Bottleneck node $p(t|t'_n, t_n)$. This node aims to combine the information on x_n in t_n with the information aggregated in t'_n and delivers the output t of the opened Information Bottleneck node. The depicted compressed image reveals that the Information Bottleneck graph is able to almost completely denoise the input image by subsequently extracting and combining the relevant information on the true pixel colors in the concatenated Information Bottleneck nodes. Almost all pixels are displayed in their true base color, as it can be seen clearly in the corresponding pie diagram and the image.

The fact that the compressed images at the outputs of the concatenated Information Bottleneck nodes evolve and subsequently become closer to the original image illustrates the process of aggregating and combining relevant information in the nodes appearing in the Information Bottleneck graph and, therefore, visualizes how relevant information flows through the considered Information Bottleneck graph.

Above it has been discussed that using the chosen cardinalities, the opened node structure requires to store 65 lookup table entries. Straightforwardly designing a single node $p(t|y_{n-1}, y_n, y_{n+1})$ by feeding $p(x_n, y_{n-1}, y_n, y_{n+1})$ to an Information Bottleneck algorithm requires $15^3 = 3375$ lookup table entries to store the resulting lookup table and is still feasible with the KL-means algorithm. An interesting question is if such a node performs better in terms of the preserved relevant information $\mathrm{I}(\mathsf{X}_n; \mathsf{T})$. In this example, the original mutual information $\mathrm{I}(\mathsf{X}_n; \mathsf{Y}_{n-1}, \mathsf{Y}_n, \mathsf{Y}_{n+1}) \approx 1.141$ bits. Directly designing the Information Bottleneck node $p(t|y_{n-1}, y_n, y_{n+1})$ with the KL-means algorithm and an output cardinality $|\mathcal{T}| = 5$ yields a preserved relevant information of $\mathrm{I}(\mathsf{X}_n; \mathsf{T}) \approx 1.1378$ bits, that is, 99.7% of the relevant information can be preserved. It turns out that *almost* exactly the same amount of preserved relevant information can be achieved with the opened node structure from Figure 4.4.

The difference in the preserved mutual information of the straightforwardly designed single node and the opened node is smaller than 10^{-4} bit.

It is important, however, that this fact is not an automatism, but has been achieved by a careful choice of the parameters of the involved relevant random variables and the output cardinalities of the Information Bottleneck algorithms applied. Decreasing the output cardinality of node $p(t'_n|t_{n-1}, t_{n+1})$ to $|\mathcal{T}'_n| = 2$, for example, reduces the preserved mutual information of the opened node structure to approximately 1.119 bit and hence, only 98% of the original relevant information $\mathrm{I}(\mathsf{X}_n; \mathsf{Y}_{n-1}, \mathsf{Y}_n, \mathsf{Y}_{n+1})$.

Finally, it is worth mentioning that despite not loosing a significant amount of relevant information in this particular case, the reduction of the lookup table size from 3375 to 65 entries has to come at some cost. A price to pay for obtaining an Information Bottleneck node $p(t|y_{n-1}, y_n, y_{n+1})$ with fewer lookup table entries in an opened node structure is that the Information Bottleneck nodes appearing in the opened node process intermediate results from their neighbors and as a result the lookups have to be performed sequentially. Therefore, this approach typically causes more latency than a direct implementation in a single lookup table.

4.6 Concluding Remarks on Information Bottleneck Graphs

In this chapter, Information Bottleneck graphs were introduced with the motivation to compactly illustrate the information relation between the variables involved in the Information Bottleneck framework in a factor graph. Despite the concept of Information Bottleneck graphs is very simple, it was shown that they can be used to place Information Bottleneck nodes in schematical diagrams of signal processing chains intuitively. In doing so, one essentially designs a flow of relevant information through a factor graph and is able to design complex systems on a high abstraction level.

The ability to open nodes then allows to dive into local nodes which process several inputs and to design their local flow of relevant information. Information Bottleneck graphs enable to design a digital signal processing chain from an information perspective. This is entirely different to conventional

signal processing approaches which typically focus on a minimum desired distortion. The major engineering task for using Information Bottleneck graphs is considering a meaningful flow of relevant information through concatenated Information Bottleneck nodes. Moreover, one has to determine all required joint distributions for their design with an Information Bottleneck algorithm. Once this is done, one is left with the choice of cardinalities of all appearing compression variables. This choice strongly influences the ability to preserve relevant information and hence has to be made carefully.

The investigated Information Bottleneck graphs in this chapter have shown many concatenated Information Bottleneck nodes which can be implemented as lookup tables. Each lookup table processes inputs and delivers an output. Some of these outputs are passed to the input of concatenated tables until finally the output t of the Information Bottleneck node is obtained. This procedure essentially is a message passing process comparable to the one discussed in Section 2.4.3 on factor graphs of probability distributions. However, obviously some key differences exist:

1. The messages passed are elements from the event spaces of the appearing compression variables instead of real valued functions of the variables involved.

2. All factor nodes appearing are Information Bottleneck nodes which preserve relevant information on some relevant variable.

3. All operations that have to be performed in the Information Bottleneck graph are lookup operations.

An emerging question is how the resulting message passing algorithm is related to the sum-product algorithm. Information Bottleneck graphs will be used to model and analyze all systems designed with the Information Bottleneck method in the remainder of the thesis. This will help to further understand the discrete message passing process on Information Bottleneck graphs. The discrete message passing on Information Bottleneck graphs will be revisited in Section 7.4 after many other practical applications of Information Bottleneck graphs have been studied.

Chapter 5

Applications of the Information Bottleneck Method in Communications

This chapter describes the application of the Information Bottleneck method to several problems of receiver-sided signal processing which have been investigated in the scope of this thesis.

The starting point of this investigation is the design of scalar channel output quantizers which perform the analog-to-digital conversion of the received signal from the transmission channel. Channel output quantizers are constructed and investigated for several channel models and modulation schemes in Section 5.1. The focal design aim of these quantizers is the maximum possible preservation of relevant information for a given output bit width of the quantizer.

Afterwards, in Section 5.2, a framework to construct quantized LDPC decoders for regular LDPC codes using the Information Bottleneck method is described. This framework requires to pair a density evolution technique with the construction of lookup table based node operations with the Information Bottleneck method. The bit error rate performance of the resulting decoders is investigated for BPSK modulated transmission over AWGN channels and compared to state-of-the-art decoding algorithms using extensive bit error rate simulations. The study of decoders for regular LDPC codes is continued in Section 5.3 for transmission systems with quadrature amplitude modulation (QAM). An intermediate step in the construction of LDPC decoders with the Information Bottleneck method is introduced which is vividly termed *message alignment* in this context. Also for regular LDPC codes with QAM modulation and message alignment, bit error rates under AWGN are compared to state-of-the-art decoding algorithms to measure the performance of the resulting LDPC decoders constructed with the Information Bottleneck method.

Following the investigation of decoders for regular LDPC codes, it is discussed in Section 5.4 that the construction of Information Bottleneck decoders for irregular LDPC codes includes some challenges which also require to use message alignment in the decoder construction and the decoding. Also for irregular LDPC codes, extensive comparisons to state-of-the-art decoding algorithms under AWGN are presented.

The focus which lies mostly on channel output quantization and LDPC decoding so far is widened starting from Section 5.5, where an entire coherent signal processing chain for detection of an LDPC encoded data transmission over a frequency flat block fading channel is constructed with the Information

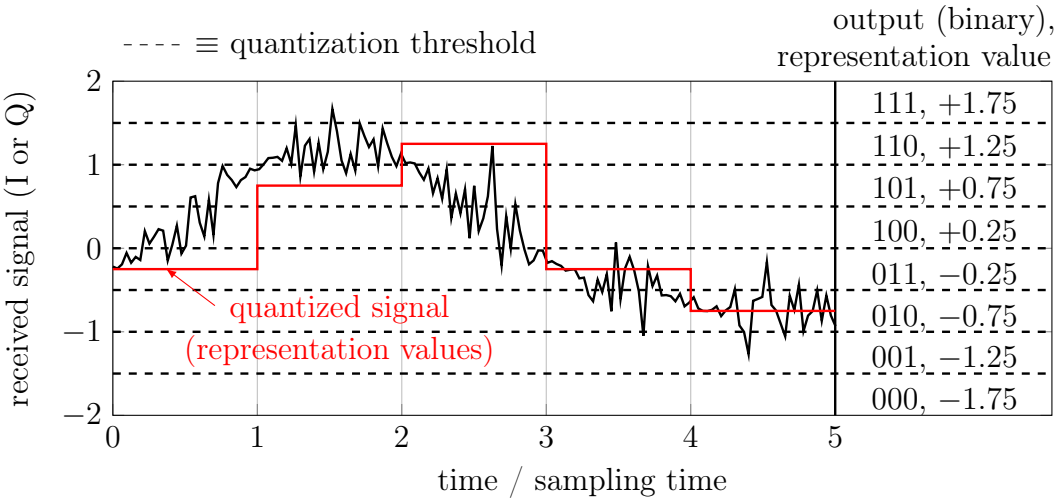


Figure 5.1: Illustration of receiver-sided 3 bit analog-to-digital conversion. The analog-to-digital converter delivers a threshold decision represented as quantization index. A representation value is assigned to each quantization region of the analog-to-digital converter which is used in the further signal processing.

Bottleneck method. The receiver structure to be presented solves all detection, channel estimation and decoding tasks completely based on lookup tables which are constructed with the Information Bottleneck method. Finally, in Section 5.6, this receiver chain is supplemented with a decision feedback aided channel estimation. Also for the Information Bottleneck based receiver chains, the bit error rate performances are compared to state-of-the-art receivers.

5.1 Information Bottleneck Quantizer Design

In digital communication receivers the received signal is sampled, quantized and processed for detection and channel decoding. The terms quantization and analog-to-digital conversion are typically used synonymously in this context. Higher precision of the analog-to-digital converter typically yields better performance, but is also more costly in terms of complexity and energy consumption. Figure 5.1 illustrates the conventional quantization process performed by an analog-to-digital converter. First, threshold decisions are performed on the continuous input signal to categorize it into quantization regions. The input

signal is then approximated using representation values which lie in the center of a quantization region in the shown example. These representation values are used in the applied algorithms, for example, to calculate LLRs for soft channel decoding. The quantization regions and the representation values are often chosen to minimize the resulting distortion with respect to the continuous input signal in the sense of the mean squared error in conventional signal processing approaches. This can be achieved by designing the thresholds and the corresponding representation values with the Lloyd-Max algorithm [64].

The Information Bottleneck method clusters the event space of an observed random variable. This clustering process is very similar to the problem of designing a quantizer because clustering the event space of an observation groups several elements together and maps them onto the corresponding compression variable. A quantizer essentially does the same.

The Information Bottleneck method can be used to determine the quantization thresholds shown in Figure 5.1 in an information-optimum manner to obtain quantizers which aim to preserve a maximum amount of relevant information for a given output bit width. The major difference to the described conventional quantization process in state-of-the-art receivers is that the considered quantizers do not necessarily have to deliver representation values from the domain of the input variable. Instead quantizers are considered which just output the index of the quantization region a received sample falls into. From an information theoretical perspective, it is not necessary to assign representation values from the domain of the input variable to the quantization regions, as the preserved relevant information of a quantizer does not depend on its representation values.

The focus of this section lies on scalar quantizers for symmetric and memoryless channels. The idea to design quantizers that aim to preserve relevant information is not entirely new. Some important preliminary works on this topic are [13, 14, 16, 58–60]. In [13, 16], Zeitler *et al.* proposed to use the Information Bottleneck method to design rate efficient quantizers for LLRs appearing at relays in network coded scenarios and to design quantizers for AWGN channels with memory. In [14, 59] Winkelbauer *et al.* investigated the achievable trade-off between preserved relevant information and compression information at the output of a quantizer for some purely Gaussian channels.

In [58], Kurkoski and Yagi presented an algorithm which is able to find *globally* optimum quantizers for binary input discrete memoryless channels and proved its optimality. In [60], the KL-means algorithm from Section 3.2.3 was used to find relevant information preserving quantizers for discrete memoryless channels with higher order input alphabets.

In this section, first a novel Information Bottleneck algorithm for channel output quantization will be introduced. This algorithm has been developed in the scope of this thesis and appeared in [65, 66]. It is a modified variant of the sequential Information Bottleneck algorithm described in Section 3.2.2 which is dedicated to the problem of designing channel output quantizers. The algorithm will then be applied to design quantizers for the AWGN channel under several input modulation alphabets. Afterwards, frequency flat fading channels will be investigated. Where it is possible, the resulting quantizers will be compared with the ones from the aforementioned literature and Lloyd-Max quantizers [64]. The designed quantizers will be used in the remainder of the thesis.

5.1.1 A Dedicated Sequential Information Bottleneck Algorithm for Channel Output Quantization

A symmetric discrete input continuous output memoryless channel with input $x \in \mathcal{X}$ and output $\tilde{y} \in \tilde{\mathcal{Y}}$ is considered. This channel is characterized by $p(\tilde{y}|x)$. Please recall that in the chosen notation a tilde indicates a continuous quantity and the lack of a tilde indicates a discrete quantity, accordingly. We aim to design a relevant information preserving quantizer $p(t|\tilde{y})$ in a way which preserves a desired maximum relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ between the quantized output $t \in \mathcal{T}$ and the transmitted modulation symbol $x \in \mathcal{X}$.

In this notation, the relation between the quantizer design problem and the Information Bottleneck problem setup with $\beta \to +\infty$ is obvious. The relevant random variable is given by the transmitted modulation symbol and the observation is the channel output. For this problem, this is an intuitive choice because a quantizer maximizing $\mathrm{I}(\mathsf{X};\mathsf{T})$ will inherently maximize the communication rate of the quantized channel [58].

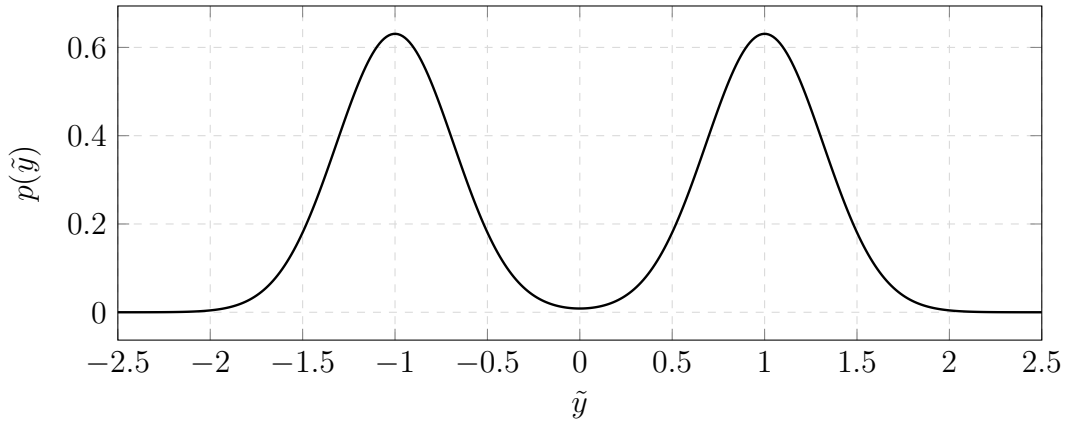
An obstacle in designing the quantizer $p(t|\tilde{y})$ is, however, that the observation is continuous. In order to apply an Information Bottleneck algorithm from Section 3.2, the received $\tilde{y}$ formally first has to be quantized to a discrete y in a very fine resolution. This process is termed *discretization* and illustrated in Figures 5.2a and 5.2b for an AWGN channel and an exemplary BPSK modulation scheme. Figure 5.2a shows the probability density function of the continuous received value $\tilde{y}$. Application of a fine granular uniform quantizer approximates the continuous received $\tilde{y}$ with a discretized y as shown in Figure 5.2b. In Figure 5.2b, 100 equidistant samples were chosen to approximate the continuous $\tilde{y}$ for exemplary purposes. If a very large number of samples is used for this discretization, the difference between the probability density function from Figure 5.2a and the probability distribution from Figure 5.2b becomes negligible, as it can concluded from the figure. If the cardinality $|\mathcal{Y}|$ is chosen large enough to guarantee that $y \approx \tilde{y}$, the resultant quantizer can in practice also be applied directly to the continuous $\tilde{y}$. Therefore, by designing the quantizer $p(t|y)$ for a very large cardinality $|\mathcal{Y}|$, one can effectively obtain $p(t|\tilde{y})$.

In order to design $p(t|y)$ with the Information Bottleneck method, one needs the joint distribution $p(x, y)$. For the considered channel and modulation scheme it can be calculated using the relation

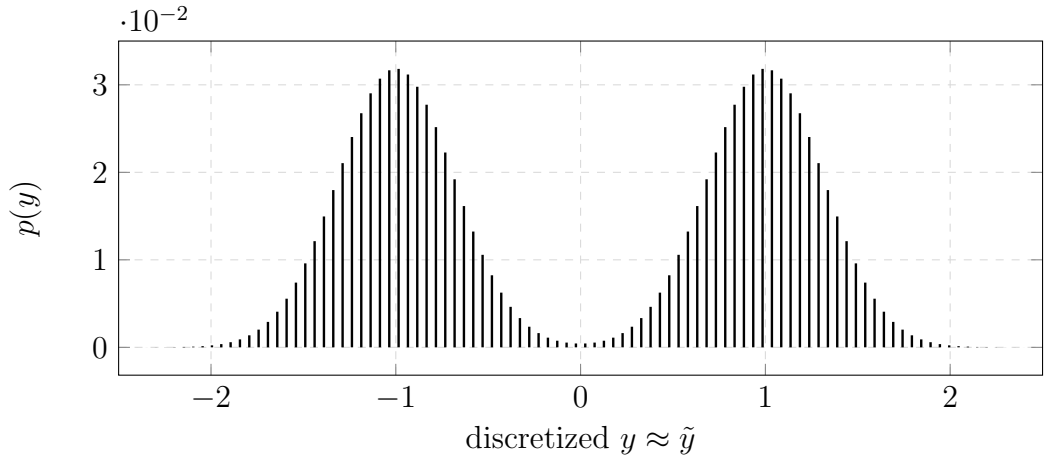
$$p(x,y) = p(y|x)p(x) \propto p(x)\frac{1}{\sqrt{2\pi}\sigma_{\tilde{n}}} \exp\left(-\frac{(y-x)^2}{2\sigma_{\tilde{n}}^2}\right), \tag{5.1}$$

where $x \in \{-1, +1\}$ and $\sigma_{\tilde{n}}^2$ is the noise variance of the AWGN. Feeding this joint distribution to an Information Bottleneck algorithm from Section 3.2 with parameter $\beta \to +\infty$ and some fixed cardinality $|\mathcal{T}|$ of the compression variable is possible and straightforwardly delivers the desired quantizer $p(t|y)$.

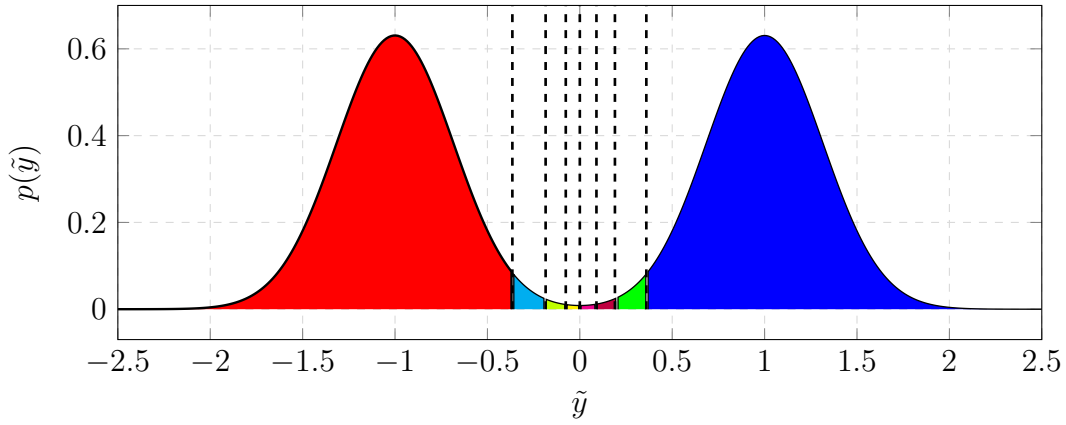
The resulting quantization regions for equally likely symbols x obtained by the sequential Information Bottleneck algorithm for an output cardinality $|\mathcal{T}| = 8$ and $\sigma_{\tilde{n}}^2 = 0.1$ are shown in Figure 5.2c. It is noteworthy, that a straightforward application of the sequential Information Bottleneck algorithm yielded continuous quantization regions separable by thresholds. Also the illustration from Figure 5.1 assumed that the quantization regions of the analog-to-digital converter are continuous intervals. Such continuous quantiza-



(a) Probability density function of the received values $\tilde{y}$ under BPSK modulation.



(b) Probability distribution of a finely quantized y under BPSK modulation.



(c) Resulting quantization regions and decision thresholds of the quantizer.

Figure 5.2: Illustration of quantizer design for the continuous channel output of the AWGN channel with $\sigma_{\tilde{n}}^2 = 0.1$ under BPSK modulation.

tion regions separated by thresholds are very desirable because they enable to implement the resulting quantizer using only elementary threshold decisions.

A modified variant of the sequential Information Bottleneck algorithm has been developed in the scope of this thesis which only optimizes over quantizers with symmetric and continuous quantization regions. This algorithm is described next. The proposed modified sequential Information Bottleneck algorithm has reduced computational costs in comparison to the original sequential Information Bottleneck algorithm from Section 3.2.2. The processing of the algorithm is consolidated in Algorithm 4 and visualized in Figure 5.3.

As it is shown in Figure 5.2c, the quantizer $p(t|y)$ is fully characterized by a set of $|\mathcal{T}| - 1$ quantization thresholds separating neighbored quantization regions. The idea of the proposed modified sequential Information Bottleneck algorithm is making use of the fact that only the quantization thresholds have to be optimized when restricting the quantizer to have continuous quantization regions. Therefore, realizations y can only move to their neighboring clusters during the sequential optimization process of the proposed Information Bottleneck algorithm.

The modified algorithm starts from a random symmetric initial clustering which separates the discretized event space $\mathcal{Y}$ into $|\mathcal{T}|$ clusters $\mathcal{Y}_0, \mathcal{Y}_1, \ldots, \mathcal{Y}_{|\mathcal{T}|-1}$ as shown in Figure 5.3a. Most importantly, this initialization consists of clusters with contiguous elements $y \in \mathcal{Y}$. As a consequence, the clusters are separated by quantization thresholds. The algorithm then loops over the quantization thresholds which separate negative realizations y.

For each cluster $\mathcal{Y}_k$ bounded by a such a quantization threshold, it picks the element y which is directly next on the left to a quantization threshold and puts it into a singleton cluster. It then calculates the merger costs $C(\{y\}, \mathcal{Y}_k)$ of putting the element back to this original cluster. The algorithm moreover calculates $C(\{y\}, \mathcal{Y}_{k+1})$ of putting the element to the neighboring cluster on the right, as illustrated in Figure 5.3b. These merger costs for $\beta \to +\infty$ and cluster $\mathcal{Y}_k$ are given by

$$C(\{y\}, \mathcal{Y}_k) = \psi(k, y)\, D_{\mathrm{JS}}^{(\pi_0, \pi_1)}\{p(x|\mathsf{Y} = y) \,|\, p(x|\mathsf{T} = k)\}, \tag{5.2}$$

Input : joint distribution $p(x, y)$, cardinality of the compression variable $|\mathcal{T}|$

Output: quantizer $p(t|y)$, $p(x|t)$ and $p(t)$

```
begin
    done←false;
    Initialize random symmetric quantizer;
    while not done do
        for quantization thresholds separating negative y do
            while not merged back to original cluster do
                Put y left of current threshold into singleton cluster;
                Put −y into another singleton cluster;
                Calculate merger costs of putting y back or to the right;
                Merge into the cluster with minimum merger costs;
                Merge −y keeping symmetry;
            end
            while not merged back to original cluster do
                Put y right of current threshold into singleton cluster;
                Put −y into another singleton cluster;
                Calculate merger costs of putting y back or to the left;
                Merge into the cluster with minimum merger costs;
                Merge −y keeping symmetry;
            end
        end
        if thresholds did not change then
            done←true;
        end
    end
end
return p(t|y), p(x|t), p(t);
```

Algorithm 4: Algorithmic description of the modified sequential Information Bottleneck algorithm for quantizer design.

(a) Start from random initial symmetric quantizer.

(b) Calculate two merger costs.

(c) Situation after merging.

(d) Try to merge from the right to the left.

Figure 5.3: Illustration of the modified sequential Information Bottleneck algorithm for quantization of symmetric channels.

where

$$\psi(k,y) = \Pr(\mathsf{Y}=y) + \Pr(\mathsf{T}=k) \tag{5.3}$$

$$\pi_0 = \pi_0(k,y) = \Pr(\mathsf{Y}=y)/\psi(k,y) \tag{5.4}$$

$$\pi_1 = \pi_1(k,y) = \Pr(\mathsf{T}=k)/\psi(k,y). \tag{5.5}$$

The respective merger costs for cluster $\mathcal{Y}_{k+1}$ are obtained equivalently.

The target cluster with the minimum merger costs is chosen. The algorithm inherently keeps the clusters holding negative y symmetrical to the ones holding positive y to preserve the channel symmetry under quantization. This is achieved by moving the (positive) element $-y$ accordingly. In the shown case in Figure 5.3b the merger costs of putting y to the next cluster on the right

are smaller and thus the element moves to the next cluster on the right. The situation after merging is shown in Figure 5.3c.

Trying to merge elements next on the left of the current quantization threshold to their neighboring cluster on the right is repeated in the first inner while loop in Algorithm 4 until an element is merged back into its original cluster $\mathcal{Y}_k$ for the first time. The algorithm then moves on to cluster $\mathcal{Y}_{k+1}$ and equivalently tries to sequentially merge the elements next on the right to the current threshold into cluster $\mathcal{Y}_k$ in the second inner while loop as illustrated in Figure 5.3d. This step first seems counterintuitive, as it is equivalent to the merger cost calculation which has just been performed. In fact, the second inner while loop terminates after the first merger cost calculation, if elements have moved from the left to the right in the first inner while loop. However, if no elements have moved from the left to the right in the first while loop, it is possible that elements move from the right to the left in the second while loop. The entire process is repeated until the resulting quantization thresholds do not change any more in the outer while loop of the algorithm.

The proposed algorithm has reduced computational costs in comparison to the sequential Information Bottleneck algorithm from Section 3.2.2. An obvious reason is that it only considers two merging possibilities in each sequential optimization step, whereas the original sequential Information Bottleneck algorithm considers all clusters in each step of sequential optimization as possible target clusters. The complexity of the proposed algorithm is dominated by the complexity of the merger cost calculation which is linear in the cardinality $|\mathcal{X}|$. Even in the worst case, where almost all y move in a run of the while loop in Algorithm 4, less than $|\mathcal{Y}|$ merger costs have to be calculated. This results in linear complexity in $|\mathcal{Y}|$ for each loop iteration.

Unfortunately, the number of iterations of the outer while loop and the number of elements that move in a particular run of this while loop cannot be predicted in general. These numbers clearly depend on the initial clustering. However, the same holds for the original sequential Information Bottleneck algorithm described in Section 3.2.2. In the following sections, Algorithm 4 will be applied to design quantizers for several transmission channels and modulation schemes. The resulting quantizers will be compared to other quantization

approaches in terms of their resulting quantization regions and their preserved relevant information.

5.1.2 Quantizers for Additive White Gaussian Noise Channels

In this section, quantizers for the AWGN channel will be designed. For this purpose the Information Bottleneck notation $\mathsf{X} \to \mathsf{Y} \to \mathsf{T}$ will be adapted to the corresponding communication problems. The received signal $\tilde{r}_k$ for the transmitted modulation symbol $s_k \in \mathcal{S}$ at time instance k under AWGN is given by

$$\tilde{r}_k = s_k + \tilde{n}_k, \tag{5.6}$$

where $\tilde{n}_k$ is a realization of a Gaussian random process with mean zero and variance $\sigma_{\tilde{n}}^2$. A relevant information preserving quantizer shall deliver $r_k \in \mathcal{R}$ from a continuous received $\tilde{r}_k \in \tilde{\mathcal{R}}$ in a way which preserves a desired maximum relevant information $\mathrm{I}(\mathsf{S}_k; \mathsf{R}_k)$ between the quantized output $r_k \in \mathcal{R}$ and the transmitted modulation symbol $s_k \in \mathcal{S}$ for a given cardinality $|\mathcal{R}|$. A description of a quantizer which is more common in communications than the conditional probability distribution $p(r_k|\tilde{r}_k)$ is a function $f_\mathrm{Q} : \tilde{\mathcal{R}} \to \mathcal{R}$ which maps a received $\tilde{r}_k$ onto a quantized r_k according to $r_k = f_\mathrm{Q}(\tilde{r}_k)$. However, both descriptions are equivalent. Their connection again can be formalized using the Kronecker delta function as $p(r_k|\tilde{r}_k) = \delta(r_k - f_\mathrm{Q}(\tilde{r}_k))$. So far, no assumption on the modulation alphabet $\mathcal{S}$ has been made. In the following, BPSK and QAM will be considered.

Quantizers for Additive White Gaussian Noise Channels with Binary Phase Shift Keying Modulation

In the previous section, it has already been discussed how the desired quantizer for BPSK with transmitted symbols $s_k \in \{-1, +1\}$ can be constructed using an Information Bottleneck algorithm. There, the continuous channel output was discretized using a very fine granular uniform quantizer. This in fact turned the continuous output AWGN channel into a binary input discrete output memoryless channel.

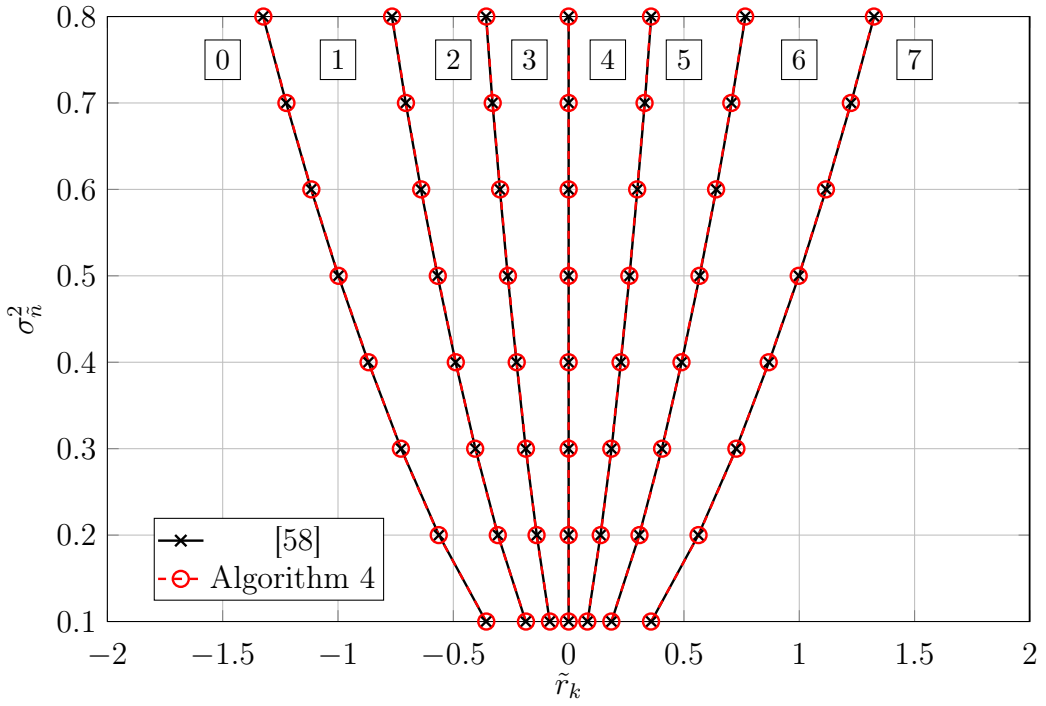


Figure 5.4: Comparison of the quantization regions from Algorithm 4 and the globally optimum algorithm from [58] for $|\mathcal{R}| = 8$ and various channel noise variances $\sigma_{\tilde{n}}^2$.

For the problem of quantizing a binary input discrete memoryless channel an algorithm exists which is guaranteed to find a globally optimum quantizer which maximizes $\mathrm{I}(\mathsf{S}_k; \mathsf{R}_k)$ for a given output cardinality $|\mathcal{R}|$. This algorithm is described in [58] and uses a dynamic programming approach. It has cubic complexity in the number of levels used to discretize the received signal.

Figure 5.4 shows a comparison of the resulting quantization regions of the algorithm from [58] and the proposed Information Bottleneck algorithm Algorithm 4 as a function of the channel noise variance $\sigma_{\tilde{n}}^2$. As it can be seen, their resulting quantization regions are equivalent. Therefore, Algorithm 4 performs equivalently to the globally optimum algorithm from [58] for the considered problem. Algorithm 4, however, has runtime benefits in comparison to the algorithm from [58] because the latter has cubic complexity in the number of channel outputs while Algorithm 4 has linear complexity in this quantity.

In [58] it was shown that if the discretized channel outputs can be sorted according to their channel LLRs, there exists a globally optimum quantizer which

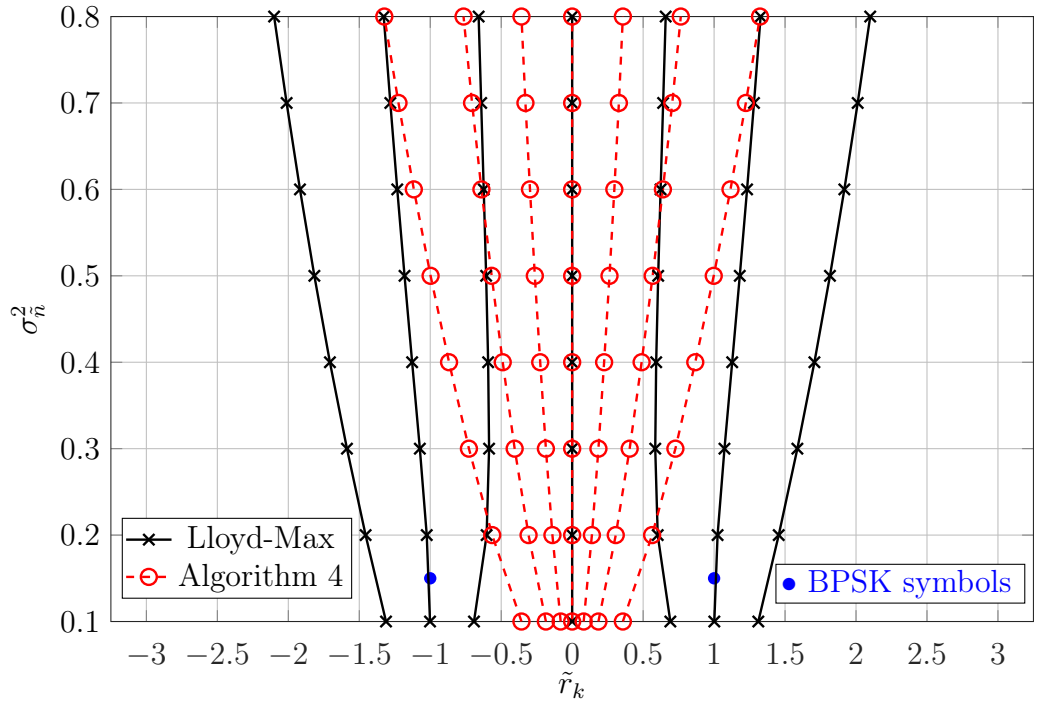
has continuous quantization regions. For BPSK under AWGN the real valued channel outputs $\tilde{r}_k$ are inherently sorted by their channel LLR. Algorithm 4 restricts the search for a proper quantizer to quantizers with contiguous quantization regions. As it can be seen in Figure 5.4, this results in the fact that Algorithm 4 finds quantizers which are equivalent to globally optimum ones.

It is important that an obtained quantizer is not considered to output a representant from the domain of the input signal here, but instead a cluster index. This is illustrated at the top of Figure 5.4 where the output indices for the respective quantization regions are mentioned. Anyway, it is easy to assign channel LLRs to these integers because Algorithm 4 delivers $p(s_k|r_k)$ and $p(r_k)$ which correspond to $p(x|t)$ and $p(t)$ in the Information Bottleneck notation. This allows to calculate a channel LLR from a quantized index $r_k \in \mathcal{R}$ which is given by

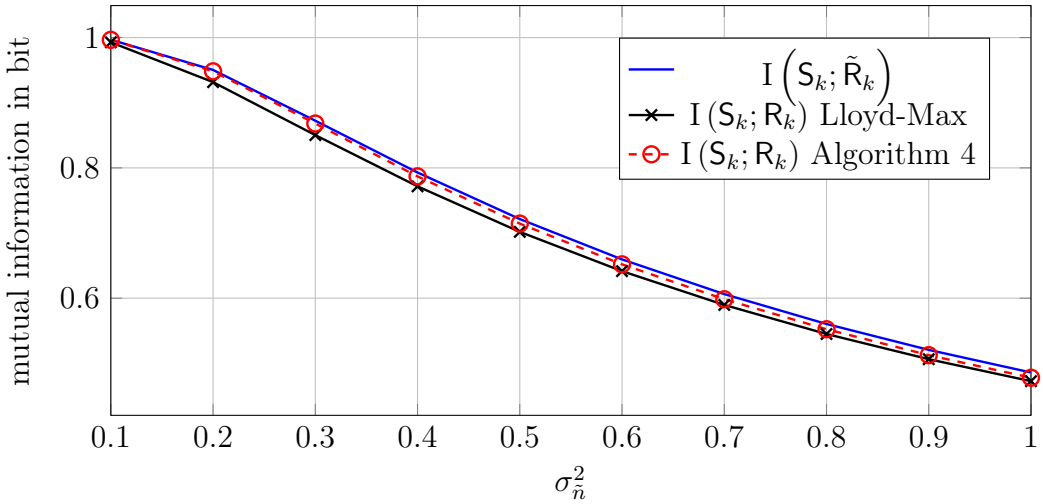
$$L^{\mathrm{ch}}(s_k) = \frac{p(r_k|\mathsf{S}_k = +1)}{p(r_k|\mathsf{S}_k = -1)}. \tag{5.7}$$

This step is required, for example, if a conventional soft decoding algorithm for a channel code shall be applied after quantization. It is noteworthy, that this eliminates the need for any algorithmic search for optimum representation values of the resulting quantization regions as one can obtain the LLRs directly from Equation (5.7).

A well known algorithm to design quantizers which aim for a minimum resulting distortion is the Lloyd-Max algorithm originally presented in [64]. An interesting question is how a Lloyd-Max quantizer differs from a quantizer constructed using Algorithm 4 in the considered scenario. Figure 5.5a compares the quantization regions obtained using the Lloyd-Max algorithm and the ones from Algorithm 4 for various channel noise variances $\sigma_{\tilde{n}}^2$ and $|\mathcal{R}| = 8$ quantization regions. The figure reveals that both quantization approaches differ significantly. For the considered BPSK, the received samples concentrate around the two constellation points $s_k \in \{-1, +1\}$. Since the Lloyd-Max algorithm aims for a minimum distortion according to the mean squared error, it focusses the decision thresholds, where the received samples are expected, that is, around the symbols ± 1. This effect is especially visible at the bottom of Figure 5.5a, where the noise variance is small. The Information Bottleneck algorithm in contrast aims to preserve the relevant information on the modulation symbol $s_k \in \{-1, +1\}$. It focusses the quantization thresholds around



(a) Comparison of the quantization regions for BPSK under AWGN obtained using the Lloyd-Max algorithm and Algorithm 4 for $|\mathcal{R}| = 8$ and various channel noise variances $\sigma^2_{\tilde{n}}$.



(b) Comparison of the preserved relevant information of the quantizers for BPSK under AWGN obtained using the Lloyd-Max algorithm and Algorithm 4 for $|\mathcal{R}| = 8$ and various channel noise variances $\sigma^2_{\tilde{n}}$.

Figure 5.5: Comparison of quantizers for BPSK under AWGN obtained using the Lloyd-Max algorithm and Algorithm 4.

the decision threshold $\tilde{r}_k = 0$, such that it can be resolved how close a received $\tilde{r}_k$ lies to this decision threshold from the quantizer output r_k.

Finally, the relevant information being preserved by both quantization algorithms is investigated. Figure 5.5b shows the mutual information $\mathrm{I}\left(\mathsf{S}_k; \tilde{\mathsf{R}}_k\right)$ which is the original mutual information before quantization as a reference. For this setup, this original mutual information $\mathrm{I}\left(\mathsf{S}_k; \tilde{\mathsf{R}}_k\right)$ before quantization can be calculated in closed form. Measured in bit it is given by [67]

$$\mathrm{I}\left(\mathsf{S}_k; \tilde{\mathsf{R}}_k\right) = \frac{1}{\log 2}\left(\frac{1}{\sigma_{\tilde{n}}^2} - \int_{-\infty}^{+\infty} \frac{\exp\left(-\frac{z^2}{2}\right)}{\sqrt{2\pi}} \log\cosh\left(\frac{1}{\sigma_{\tilde{n}}^2} - \frac{1}{\sigma_{\tilde{n}}} z\right) \mathrm{d}\, z\right). \quad (5.8)$$

The figure also shows the mutual information $\mathrm{I}(\mathsf{S}; \mathsf{R})$ after quantization with the Lloyd-Max algorithm and Algorithm 4 for various channel noise variances $\sigma_{\tilde{n}}^2$. The Information Bottleneck algorithm Algorithm 4 finds quantizers which preserve greater amounts of relevant information than the Lloyd-Max algorithm. This is visible especially for small $\sigma_{\tilde{n}}^2$ on the left of Figure 5.5b. The figure also shows that for increasing values of $\sigma_{\tilde{n}}^2$ the advantage of the Information Bottleneck quantizer decreases.

As evaluating a gain of a certain quantization algorithm solely from its preserved relevant information is rather abstract, the practical advantages of the proposed Information Bottleneck quantizers shall be illustrated in the following example.

Example 5.1.2.1. A practical end-to-end measure which can be used to evaluate the quality of a channel output quantizer is the bit error rate of a coded transmission scheme with a receiver that uses a soft decision decoding algorithm. Figure 5.6 shows the bit error rates of LDPC encoded transmissions using BPSK modulation over an AWGN channel with quantized channel outputs for various channel output quantizers constructed either with the Lloyd-Max algorithm or Algorithm 4. The applied code was taken from [68], where it is listed under the identifier 8000.4000.3.483. This code is termed *regular code a)* in this thesis. It has code rate $R = 0.5$ and node degrees $(d_\mathrm{v}, d_\mathrm{c}) = (3, 6)$. The decoding algorithm used for all shown curves is belief propagation decoding with double precision and a maximum of $i_\mathrm{max} = 50$ decoding iterations.

The blue curve on the very left of Figure 5.6 shows the bit error rate of the belief propagation decoder without a channel output quantizer. The other

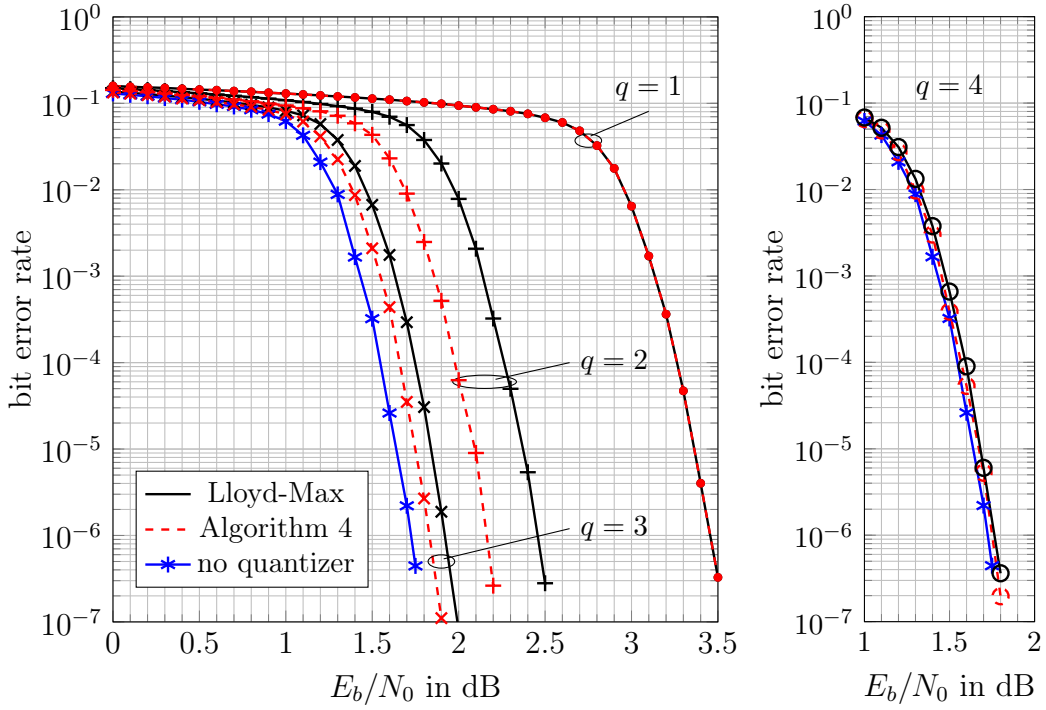


Figure 5.6: Bit error rate of an LDPC encoded transmission with BPSK modulation under AWGN. Double precision belief propagation decoding of *regular code a)* with a codeword length of $8,000$ bits is applied to q bit quantized output and to continuous output channels. The quantizers were constructed using Algorithm 4 or the Lloyd-Max algorithm.

curves labeled by the respective quantizer design algorithm Lloyd-Max or Algorithm 4 refer to a situation, where the continuous channel output was first quantized using a q bit quantizer with $|\mathcal{R}| = 2^q$ quantization regions. LLRs were calculated from the outputs of the respective quantizers according to Equation (5.7) in double precision and fed to the belief propagation decoder.

As it can be observed in the left part of the figure, already $q = 3$ bit quantization, that is, using $2^3 = 8$ quantizer output levels typically looses only 0.1 dB over E_b/N_0 in the waterfall region of the code in comparison to the non-quantized case, if Algorithm 4 is used for quantizer design. The Lloyd-Max quantizer induces an additional loss of approximately 0.1 dB in this case. Decreasing the number of quantization regions to $2^2 = 4$ shows that the loss of the Lloyd-Max algorithm in comparison to the Information Bottleneck approach increases. Using $|\mathcal{R}| = 2^4 = 16$ quantization regions

almost completely eliminates the loss resulting from quantization with the proposed Information Bottleneck quantizers, as it is illustrated in the right part of Figure 5.6. Also for $q = 4$, the Information Bottleneck quantizers offer slightly better performance over E_b/N_0 than the corresponding Lloyd-Max quantizers, but in this case the shown curves almost match.—*End of the example.*

Quantizers for Additive White Gaussian Noise Channels with Quadrature Amplitude Modulation

When high spectral efficiency shall be achieved, higher order modulation schemes which allow to assign several bits to one modulation symbol have to be used. A famous linear modulation scheme which is being applied in numerous communication standards, for example, the IEEE 802.11 wireless LAN standard [42] is the $|\mathcal{S}|$-QAM scheme, where $|\mathcal{S}|$ is the number of output levels of the modulator. $|\mathcal{S}|$-QAM essentially is built out of two independent amplitude shift keying (ASK) modulation patterns with $\sqrt{|\mathcal{S}|}$ output levels each. These ASK modulations are applied in the inphase and the quadrature path of the transmitter synchronously, resulting in a total of $|\mathcal{S}|$ complex modulation symbols.

As a result, $|\mathcal{S}|$-QAM allows to assign $\log_2 |\mathcal{S}|$ bits to one modulation symbol. Figure 5.7 shows an exemplary symbol alphabet of a 16-QAM scheme with complex modulation symbols $s_k = s_k^{\mathrm{re}} + j s_k^{\mathrm{im}}$. The number of modulation symbols allows for a huge number of assignments of bits to the distinct modulation symbols s_k. The assignment of bits to modulation symbols is termed the *bit labeling* of a modulation pattern. However, it is irrelevant for the design of a quantizer which aims for preserving the maximum amount of relevant information on the modulation symbols s_k. The received signal under $|\mathcal{S}|$-QAM modulation takes complex values. The total noise variance $\sigma_{\tilde{n}}^2$ splits up in the inphase and the quadrature component equally, such that both components are disturbed by independent Gaussian processes with variance $\sigma_{\tilde{n}}^2/2$.

From an implementation perspective it is desirable to quantize the real and the imaginary part of the received signal $\tilde{r}_k = \tilde{r}_k^{\mathrm{re}} + j\tilde{r}_k^{\mathrm{im}}$ independently because doing so, the analog-to-digital-converter can be implemented using only simple threshold decisions on the respective quadrature components. Hence, quantiz-

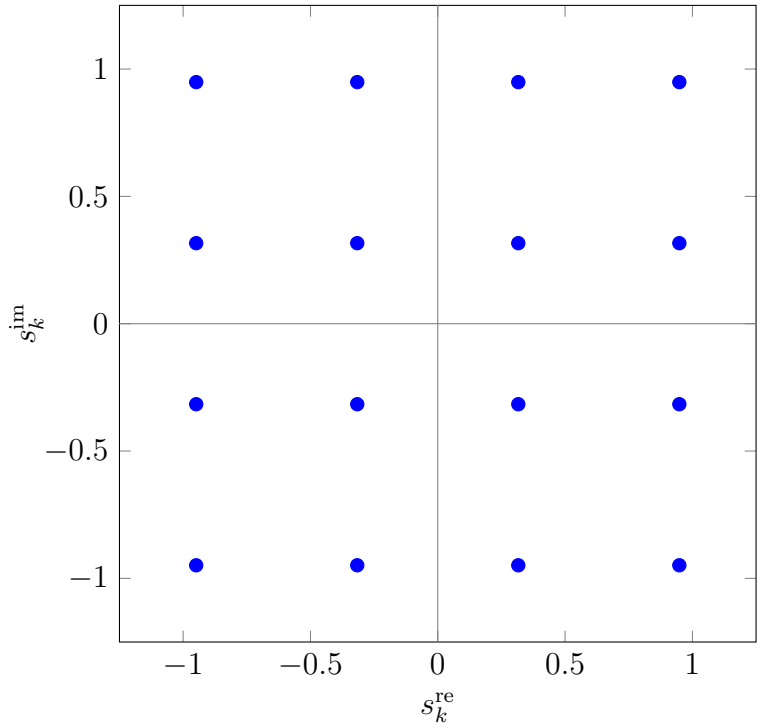


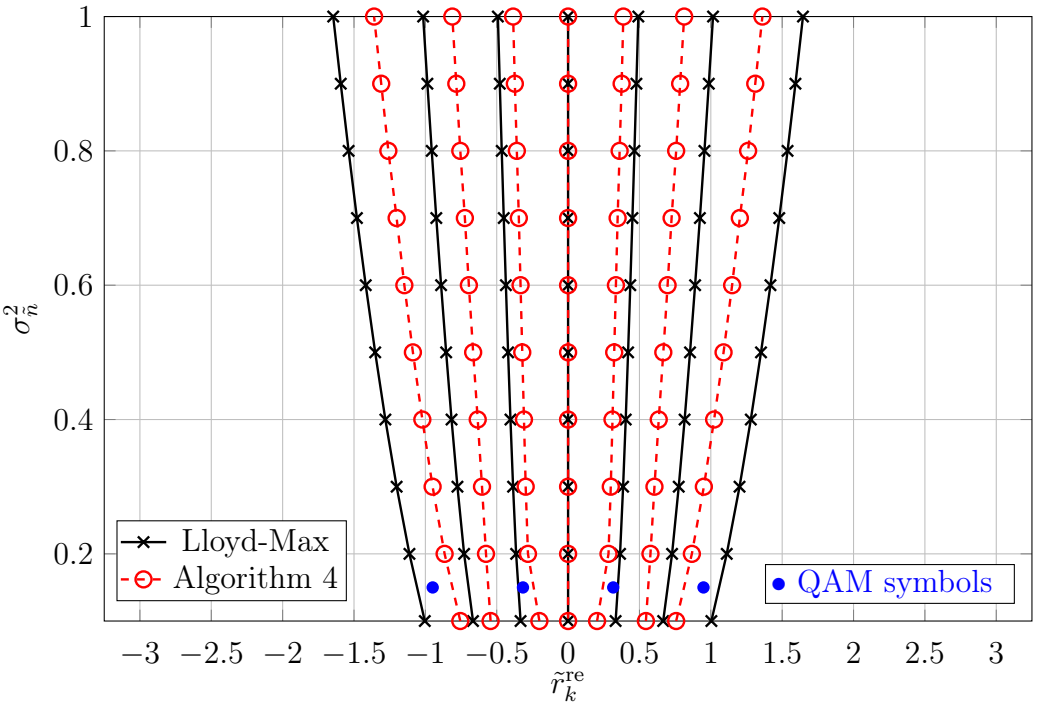
Figure 5.7: Illustration of a 16-QAM modulation alphabet.

ers aiming to maximize $\mathrm{I}\left(\mathsf{S}_k^{\mathrm{re}}; \mathsf{R}_k^{\mathrm{re}}\right)$ for the inphase and equivalently $\mathrm{I}\left(\mathsf{S}_k^{\mathrm{im}}; \mathsf{R}_k^{\mathrm{im}}\right)$ for the quadrature component of the received signal are constructed using Algorithm 4.

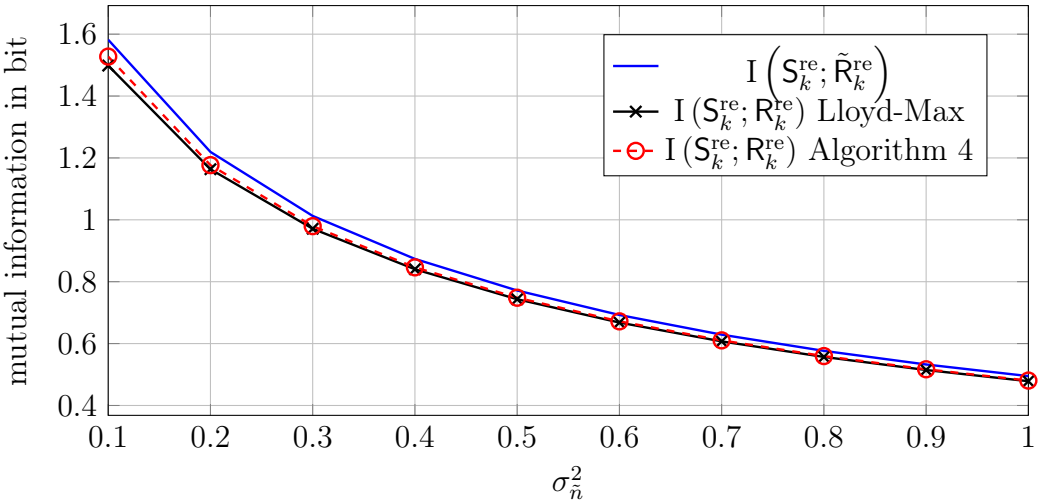
Since the inphase and the quadrature component of the received signal have identical statistical properties, it is sufficient to consider design of the quantizer for the real signal component. This quantizer can be designed with the Information Bottleneck method using the joint distribution

$$p(s_k^{\mathrm{re}}, \tilde{r}_k^{\mathrm{re}}) = \frac{1}{\sqrt{\pi |\mathcal{S}|} \sigma_{\tilde{n}}} \exp\left(-\frac{(\tilde{r}_k^{\mathrm{re}} - s_k^{\mathrm{re}})^2}{\sigma_{\tilde{n}}^2} \right). \tag{5.9}$$

Figure 5.8a shows the resulting quantization regions obtained using Algorithm 4 and again compares them to quantization regions obtained using the Lloyd-Max algorithm for $|\mathcal{R}^{\mathrm{re}}| = 8$ output levels and various channel noise variances $\sigma_{\tilde{n}}^2$. Similar to the previously investigated BPSK case, the obtained quantization regions of both algorithms differ significantly. Moreover, the quantizers constructed using Algorithm 4 preserve slightly larger amounts of relevant information as it can be seen in Figure 5.8b for $\sigma_{\tilde{n}}^2 \leq 0.2$. However, the differences between both quantization algorithms in terms of their preser-



(a) Comparison of the quantization regions for 16-QAM under AWGN obtained using the Lloyd-Max algorithm and Algorithm 4 for $|\mathcal{R}^{\mathrm{re}}| = 8$ and various channel noise variances $\sigma_{\tilde{n}}^2$.



(b) Comparison of the preserved amount of relevant information of the quantizers for 16-QAM under AWGN obtained using the Lloyd-Max algorithm and Algorithm 4 for $|\mathcal{R}^{\mathrm{re}}| = 8$ and various channel noise variances $\sigma_{\tilde{n}}^2$.

Figure 5.8: Comparison of quantizers for 16-QAM under AWGN obtained using the Lloyd-Max algorithm and Algorithm 4.

vation of relevant information vanish with increasing channel noise variance $\sigma_{\tilde{n}}^2$, similar as it has been observed under BPSK.

5.1.3 Quantizers for Frequency Flat Fading Channels

Another frequently used channel model is the frequency flat fading channel. For this channel model the received signal depends on a time variant complex channel coefficient $\tilde{h}_k$ which weights the transmitted signal at time instance k. In addition, Gaussian noise represents the receiver noise, such that the complex received signal $\tilde{r}_k$ at sampling instance k reads

$$\tilde{r}_k = \tilde{h}_k s_k + \tilde{n}_k = \underbrace{\tilde{h}_k^{\mathrm{re}} s_k^{\mathrm{re}} - \tilde{h}_k^{\mathrm{im}} s_k^{\mathrm{im}} + \tilde{n}_k^{\mathrm{re}}}_{\tilde{r}_k^{\mathrm{re}}} + j \underbrace{\left(\tilde{h}_k^{\mathrm{im}} s_k^{\mathrm{re}} + \tilde{h}_k^{\mathrm{re}} s_k^{\mathrm{im}} + \tilde{n}_k^{\mathrm{im}} \right)}_{\tilde{r}_k^{\mathrm{im}}}. \quad (5.10)$$

Just as in the preceding section, the channel noise comes from an uncorrelated zero mean complex Gaussian process which splits the overall noise power $\sigma_{\tilde{n}}^2$ equally into the real and the imaginary signal component. Also the channel coefficients $\tilde{h}_k$ are realizations from a complex zero mean Gaussian random process with variance $\sigma_{\tilde{h}}^2$.

The channel model depends on the correlation properties of the channel coefficients $\tilde{h}_k$. In this thesis, a frequency flat block fading channel will be considered. For this channel, the channel coefficients $\tilde{h}_k$ are constant for a block of length B. After B symbols have passed the channel, a new channel coefficient $\tilde{h}_k$ is drawn independently which weights the following B transmitted symbols.

The channel coefficients $\tilde{h}_k$ are unknown to the receiver and hence need to be estimated if coherent detection shall be applied. A first question is, therefore, what information a relevant information preserving quantizer should actually aim to preserve in the first place. Analyzing Equation (5.10) tells that there are the useful parts of the received signal reading

$$\tilde{x}_k^{\mathrm{re}} = \tilde{h}_k^{\mathrm{re}} s_k^{\mathrm{re}} - \tilde{h}_k^{\mathrm{im}} s_k^{\mathrm{im}} \quad (5.11)$$

$$\tilde{x}_k^{\mathrm{im}} = \tilde{h}_k^{\mathrm{im}} s_k^{\mathrm{re}} + \tilde{h}_k^{\mathrm{re}} s_k^{\mathrm{im}}, \quad (5.12)$$

for the real and the imaginary part of the received signal, respectively. For the considered channel model, quantizing the received signal $\tilde{r}_k$ in the inphase and

the quadrature path, such that information on $\tilde{\mathsf{X}}_k^{\mathrm{re}}$ and $\tilde{\mathsf{X}}_k^{\mathrm{im}}$ is being preserved, respectively, is an intuitive choice. In the following, this case is investigated for BPSK modulation.

Quantizers for Frequency Flat Fading Channels with Binary Phase Shift Keying Modulation

For BPSK, the relevant quantities in the inphase and the quadrature path simplify to

$$\tilde{x}_k^{\mathrm{re}} = \tilde{h}_k^{\mathrm{re}} s_k^{\mathrm{re}} \tag{5.13}$$

$$\tilde{x}_k^{\mathrm{im}} = \tilde{h}_k^{\mathrm{im}} s_k^{\mathrm{re}} \tag{5.14}$$

because $s_k^{\mathrm{im}} = 0$ for all $s_k \in \{-1, +1\}$. As a result, for the channel model under consideration, $\tilde{x}_k^{\mathrm{re}}$ and $\tilde{x}_k^{\mathrm{im}}$ have identical statistical properties. Hence, only $\tilde{x}_k^{\mathrm{re}}$ will be considered in the following. Recall that $\tilde{h}_k^{\mathrm{re}}$ is drawn from a zero mean Gaussian process with variance $\sigma_{\tilde{h}}^2/2$. A direct consequence is that also $\tilde{x}_k^{\mathrm{re}}$ follows a Gaussian distribution with the same parameters because multiplication by $s_k^{\mathrm{re}} \in \{-1, +1\}$ does not change the distribution of $\tilde{h}_k^{\mathrm{re}}$ if the mean of $\tilde{h}_k^{\mathrm{re}}$ is zero. As a consequence, the relevant random variable $\tilde{\mathsf{X}}_k^{\mathrm{re}}$ is Gaussian with mean zero and variance $\sigma_{\tilde{h}}^2/2$. Reviewing Equation (5.10) moreover shows that $\tilde{r}_k^{\mathrm{re}}$ is given by

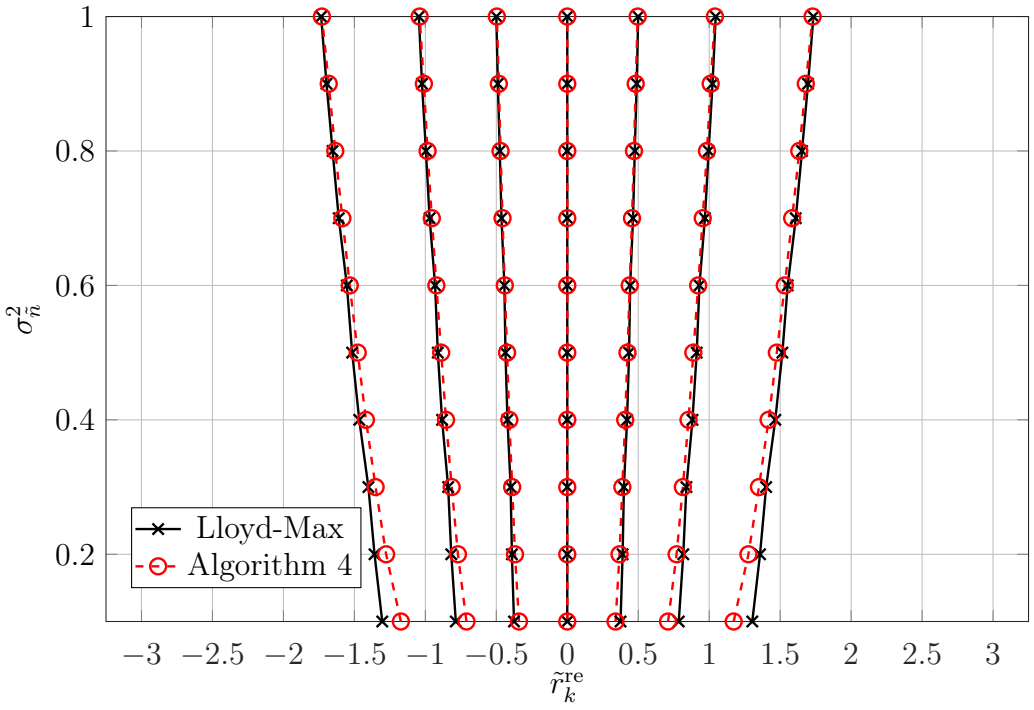
$$\tilde{r}_k^{\mathrm{re}} = \tilde{x}_k^{\mathrm{re}} + \tilde{n}_k^{\mathrm{re}}. \tag{5.15}$$

Thus, the observed $\tilde{r}_k^{\mathrm{re}}$ is a sum of two independent Gaussian random variables with mean zero which is well known to also be Gaussian with mean zero and variance equal to the sum of both summands' variances. Hence, for the considered problem, the observation and the relevant random variable are jointly Gaussian. Their joint probability distribution can be obtained as

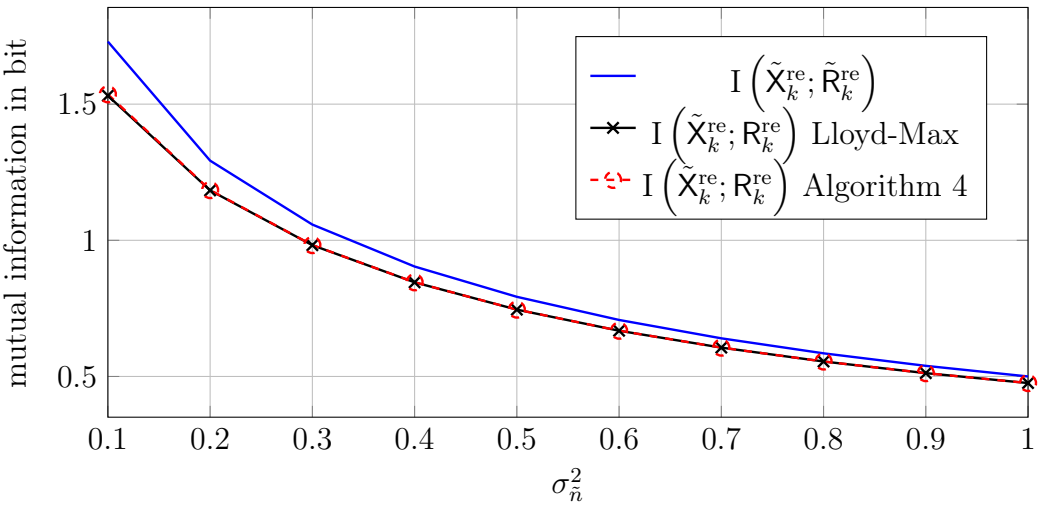
$$p(\tilde{x}_k^{\mathrm{re}}, \tilde{r}_k^{\mathrm{re}}) = \frac{1}{\pi \sigma_{\tilde{n}} \sigma_{\tilde{h}}} \exp\left(-\frac{(\tilde{r}_k^{\mathrm{re}} - \tilde{x}_k^{\mathrm{re}})^2}{\sigma_{\tilde{n}}^2} - \frac{(\tilde{x}_k^{\mathrm{re}})^2}{\sigma_{\tilde{h}}^2} \right). \tag{5.16}$$

The mutual information $\mathrm{I}\left(\tilde{\mathsf{X}}_k^{\mathrm{re}}; \tilde{\mathsf{R}}_k^{\mathrm{re}}\right)$ in bit can be calculated in closed form as [67]

$$\mathrm{I}\left(\tilde{\mathsf{X}}_k^{\mathrm{re}}; \tilde{\mathsf{R}}_k^{\mathrm{re}}\right) = \frac{1}{2} \log_2 \left(1 + \frac{\sigma_{\tilde{h}}^2}{\sigma_{\tilde{n}}^2} \right). \tag{5.17}$$



(a) Comparison of the quantization regions for BPSK under flat fading obtained using the Lloyd-Max algorithm [64] and Algorithm 4 for $|\mathcal{R}^{\mathrm{re}}| = 8$ and various channel noise variances $\sigma_{\tilde{n}}^2$ and $\sigma_{\tilde{h}}^2 = 1$.



(b) Comparison of the preserved amount of relevant information of the quantizers for BPSK under flat fading obtained using the Lloyd-Max algorithm [64] and Algorithm 4 for $|\mathcal{R}^{\mathrm{re}}| = 8$ and various channel noise variances $\sigma_{\tilde{n}}^2$ and $\sigma_{\tilde{h}}^2 = 1$.

Figure 5.9: Comparison of quantizers for BPSK under flat fading obtained using the Lloyd-Max algorithm and Algorithm 4.

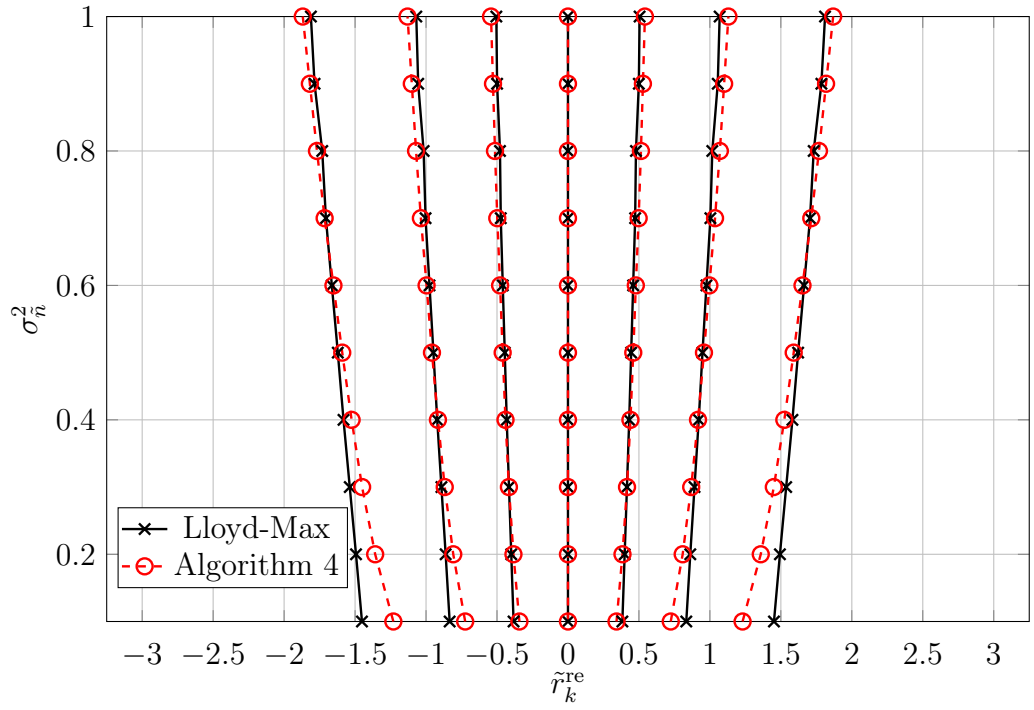
It shall be noted that optimum compression of the channel output in this scenario is a use case of the so called Gaussian Information Bottleneck [69] which is studied in greater detail in [14, 59]. The Gaussian Information Bottleneck [69], however, does not yield a scalar quantizer, but rather considers a compression implemented by a linear transformation of the observation. It is, therefore, not studied in more detail in this thesis.

Figure 5.9 compares the quantizers resulting from application of the Lloyd-Max algorithm and Algorithm 4 for $|\mathcal{R}^{\mathrm{re}}| = 8$. As it can be seen in Figure 5.9a, the quantization regions of both quantizers differ only for small values of $\sigma_{\tilde{n}}^2$ at the bottom of the illustration. However, the quantization thresholds become identical as the noise variance increases. Figure 5.9b reveals that both quantization algorithms yield practically identical amounts of preserved relevant information. However, it shall be emphasized they are not exactly identical, but the ones from Algorithm 4 are very slightly superior. The largest measured difference between $\mathrm{I}\left(\tilde{\mathsf{X}}_k^{\mathrm{re}}; \mathsf{R}_k^{\mathrm{re}}\right)$ obtained with Algorithm 4 and $\mathrm{I}\left(\tilde{\mathsf{X}}_k^{\mathrm{re}}; \mathsf{R}_k^{\mathrm{re}}\right)$ obtained with the Lloyd-Max algorithm in Figure 5.9b, appears for $\sigma_{\tilde{n}}^2 = 0.1$ and is less than $5 \cdot 10^{-3}$ bit and hence negligible in practice. For larger $\sigma_{\tilde{n}}^2$, the curves converge, just as the corresponding quantization regions do.

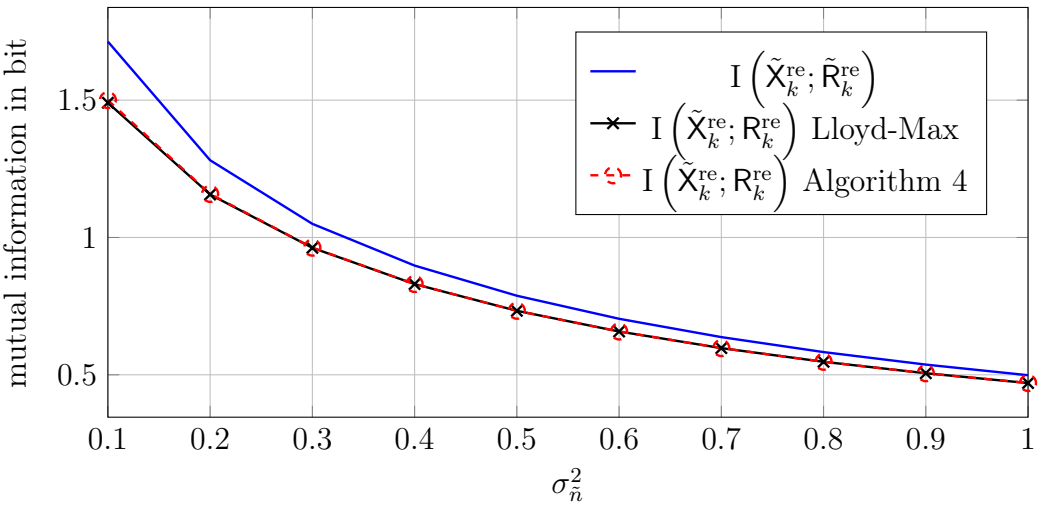
Quantizers for Frequency Flat Fading Channels with Quadrature Amplitude Modulation

Unlike for BPSK, the relevant quantities $\tilde{x}_k^{\mathrm{re}}$ and $\tilde{x}_k^{\mathrm{im}}$ according to Equations (5.11) and (5.12) cannot be simplified for QAM. Anyway, since the QAM modulation alphabet is symmetric to the origin of the complex plane, it is again noted that $\tilde{x}_k^{\mathrm{re}}$ and $\tilde{x}_k^{\mathrm{im}}$ just as $\tilde{r}_k^{\mathrm{re}}$ and $\tilde{r}_k^{\mathrm{im}}$ have identical statistical properties. Therefore, again only quantizer design for the real signal component $\tilde{r}_k^{\mathrm{re}}$ is considered in this section. In order to design a quantizer preserving relevant information on $\tilde{x}_k^{\mathrm{re}}$ with the Information Bottleneck method, one needs the joint distribution

$$p(\tilde{x}_k^{\mathrm{re}}, \tilde{r}_k^{\mathrm{re}}) = p(\tilde{r}_k^{\mathrm{re}} | \tilde{x}_k^{\mathrm{re}}) p(\tilde{x}_k^{\mathrm{re}}). \tag{5.18}$$



(a) Comparison of the quantization regions for 16-QAM under flat fading obtained using the Lloyd-Max algorithm and Algorithm 4 for $|\mathcal{R}^{\mathrm{re}}| = 8$ and various channel noise variances $\sigma^2_{\tilde{n}}$ and $\sigma^2_{\tilde{h}} = 1$.



(b) Comparison of the preserved amount of relevant information of the quantizers for 16-QAM under flat fading obtained using the Lloyd-Max algorithm and Algorithm 4 for $|\mathcal{R}^{\mathrm{re}}| = 8$ and various channel noise variances $\sigma^2_{\tilde{n}}$ and $\sigma^2_{\tilde{h}} = 1$.

Figure 5.10: Comparison of quantizers for 16-QAM under flat fading obtained using the Lloyd-Max algorithm and Algorithm 4.

It is easy to see that $p(\tilde{r}_k^{\mathrm{re}}|\tilde{x}_k^{\mathrm{re}})$ again is a Gaussian distribution with mean $\tilde{x}_k^{\mathrm{re}}$ and variance $\sigma_{\tilde{n}}^2/2$. Other than for BPSK, however, $p(\tilde{x}_k^{\mathrm{re}})$ is not a Gaussian distribution, as it is revealed by considering that the relation

$$p(\tilde{x}_k^{\mathrm{re}}) = \sum_{s_k^{\mathrm{re}} \in \mathcal{S}^{\mathrm{re}}} \sum_{s_k^{\mathrm{im}} \in \mathcal{S}^{\mathrm{im}}} p(\tilde{x}_k^{\mathrm{re}}|s_k^{\mathrm{re}}, s_k^{\mathrm{im}}) p(s_k^{\mathrm{re}}) p(s_k^{\mathrm{im}}) \tag{5.19}$$

holds. Rewriting Equation (5.11) as

$$\tilde{x}_k^{\mathrm{re}} = \left[s_k^{\mathrm{re}}, -s_k^{\mathrm{im}} \right] \cdot \begin{bmatrix} \tilde{h}_k^{\mathrm{re}} \\ \tilde{h}_k^{\mathrm{im}} \end{bmatrix} \tag{5.20}$$

reveals that for a given modulation symbol, $\tilde{x}_k^{\mathrm{re}}$ is obtained by a linear transformation of vector $\left[\tilde{h}_k^{\mathrm{re}}, \tilde{h}_k^{\mathrm{im}}\right]^{\mathrm{T}}$. This vector is jointly Gaussian with both components being uncorrelated, having zero mean and variance $\sigma_{\tilde{h}}^2/2$, respectively. As a result, $p(\tilde{x}_k^{\mathrm{re}}|s_k^{\mathrm{re}}, s_k^{\mathrm{im}})$ in Equation (5.19) is a Gaussian distribution with mean zero and variance

$$\sigma_{\tilde{x}_k^{\mathrm{re}}}^2 = \frac{\sigma_{\tilde{h}}^2}{2} |s_k^{\mathrm{re}}|^2 + \frac{\sigma_{\tilde{h}}^2}{2} |s_k^{\mathrm{im}}|^2. \tag{5.21}$$

Therefore, $p(\tilde{x}_k^{\mathrm{re}})$ is a weighted sum of Gaussian distributions, also known as a Gaussian mixture distribution.

Finally, for equally likely symbols $s_k^{\mathrm{re}} + j s_k^{\mathrm{im}}$, the desired joint distribution is obtained as

$$p(\tilde{x}_k^{\mathrm{re}}, \tilde{r}_k^{\mathrm{re}}) = \frac{1}{|\mathcal{S}^{\mathrm{re}}||\mathcal{S}^{\mathrm{im}}|} \sum_{s_k^{\mathrm{re}} \in \mathcal{S}^{\mathrm{re}}} \sum_{s_k^{\mathrm{im}} \in \mathcal{S}^{\mathrm{im}}} \frac{1}{\pi \sigma_{\tilde{n}} \sigma_{\tilde{h}} \sqrt{|s_k^{\mathrm{re}}|^2 + |s_k^{\mathrm{im}}|^2}} \cdot \exp\left(-\frac{(\tilde{r}_k^{\mathrm{re}} - \tilde{x}_k^{\mathrm{re}})^2}{\sigma_{\tilde{n}}^2} - \frac{(\tilde{x}_k^{\mathrm{re}})^2}{\sigma_{\tilde{h}}^2 \left(|s_k^{\mathrm{re}}|^2 + |s_k^{\mathrm{im}}|^2\right)} \right). \tag{5.22}$$

Again, 16-QAM is considered exemplarily. Figure 5.10a compares the quantization regions obtained using the Lloyd-Max algorithm and Algorithm 4 for various $\sigma_{\tilde{n}}^2$ and $|\mathcal{R}^{\mathrm{re}}| = 8$. As it can be seen there, fundamentally, the Lloyd-Max quantizers and the Information Bottleneck quantizers behave similarly as for BPSK under fading. However, their quantization regions do not exactly converge for increasing $\sigma_{\tilde{n}}^2$ as it could be observed in the case of BPSK in Figure 5.9a. The comparison of the preserved relevant information of both quantization approaches in Figure 5.10b reveals that both algorithms preserve comparable amounts of relevant information. Therefore, it can be concluded that also in this case, both quantization algorithms find approximately equivalent quantizers in terms of the preservation of relevant information.

5.1.4 Summary

Information Bottleneck quantizer design for the channel output of several memoryless transmission channels under BPSK and QAM modulation has been investigated. For this purpose, first a modified variant of the sequential Information Bottleneck algorithm has been developed and presented. For the AWGN channel with BPSK, it could be shown by comparison to the globally optimum quantizers from [58] that the proposed Information Bottleneck algorithm delivers quantizers practically identical to the global optimum ones at reduced computational complexity. The optimum quantization regions differ significantly from the Lloyd-Max quantization regions for BPSK and QAM under AWGN. For the BPSK case, it was illustrated exemplarily that using a relevant information preserving quantizer can significantly improve the bit error rate performance of a receiver with soft decision decoding in comparison to a Lloyd-Max quantizer. This especially holds for small quantization bit widths $q \leq 3$.

Quantizer design for the frequency flat Rayleigh fading channel under BPSK modulation has shown that the proposed Information Bottleneck algorithm finds quantizers practically equivalent to the ones from the Lloyd-Max algorithm in terms of their preservation of relevant information. For low signal-to-noise ratios, the quantization regions of both algorithms are equivalent. It shall be noted that these results are in line with some observations on the achievability of the optimum trade-off between compression rate and preserved relevant information with Lloyd-Max quantizers for purely Gaussian random variables described in [14, 59]. For QAM under flat fading, a similar behaviour has been observed.

In the following sections, the quantizers designed with the Information Bottleneck method will be applied as the first signal processing step designed with the Information Bottleneck method at the receiver. The quantizers deliver unsigned integer indices to be processed in the remainder of the signal processing chain. An output integer can be represented in the receiver hardware using q bits if the quantizer has 2^q output levels. Hence, if coarse quantization is applied, the quantized signal is represented very compactly in the hardware by using the quantization indices. Moreover, it removes the need for any high precision signal processing, if signal processing algorithms can be found which

are able to handle the plain indices from the quantizer. The ambitious aim in the following is to also construct channel decoders and detection schemes which also only work on such unsigned integer indices with the Information Bottleneck method.

5.2 Information Bottleneck LDPC Decoders for Regular Codes

The state-of-the-art decoding algorithms for LDPC codes process LLRs, as it has been explained in Section 2.3.3. Unfortunately, LLRs are real numbers and have to be represented and processed using many bits per LLR. The threshold decisions from the channel output quantizer, in contrast, can be represented as unsigned integers which can be stored and processed very efficiently in hardware. As a consequence, it would be desirable to build a decoding algorithm which has no need to process real valued LLRs, but that directly works on the quantization indices from the channel output quantizer. However, if one wanted to dismiss LLRs from the decoding process and to use quantization indices instead, one would also have to replace the node operations in the decoding process because these are defined for LLRs.

Already in [53], Richardson and Urbanke described decoding algorithms with *discrete* or *finite* message alphabets. In [28], Kurkoski *et al.* introduced a systematical concept for construction of such discrete decoders which uses mutual information as a design criterion. They proposed to use a greedy mutual information maximizing quantizer design algorithm to find lookup tables serving as node operations in a discrete decoder. The decoding thresholds of the resulting decoders were investigated, but no performance results for practical decoders were presented in [28]. Anyway, the innovative idea of this work was choosing an output message from a finite set of integers such that it shares a desired huge amount of mutual information with the bit this message represents. This focal idea laid the foundation for later works [29, 32]. Here, the decoder construction scheme from [28] which is based on a discrete density evolution technique was modified by application of a quantizer design algorithm from [58] in the density evolution process and shown to obtain decoders with performance close to belief propagation decoding. This approach was also

taken in [30] which deals with a concrete implementation of decoders based on mutual information preserving lookup tables for the variable node operation. Some more ideas and details on the decoders developed in [30] were presented and discussed in deep in [31].

At the same time, in [65, 66, 70] we proposed to use an Information Bottleneck algorithm in discrete density evolution to construct the node operations in the scope of this thesis. It turns out that the Information Bottleneck method is the perfect tool to construct the desired message passing decoders which only work on quantization indices. This section explains, how such decoders can be constructed by repeated application of Information Bottleneck algorithms and how they can be modelled using Information Bottleneck graphs. In this section, only regular LDPC codes with BPSK modulation under AWGN will be investigated. Irregular codes and other transmission channels will be considered later.

5.2.1 Design of Check Node Decoding Functions with the Information Bottleneck Method

Consider a setup, where two arbitrary codeword bits b_0, b_1 from an LDPC codeword are transmitted over an AWGN channel using BPSK modulation. At the output of this channel, a q bit Information Bottleneck quantizer is applied, such that effectively, one has a discrete input and discrete output transmission channel which delivers quantization indices from the set $\mathcal{Y}_{\text{chan}} = \{0, 1, \ldots, 2^q - 1\}$. Let the channel outputs corresponding to input bits b_0, b_1 be denoted by $y_0, y_1 \in \mathcal{Y}_{\text{chan}}$. Due to the structure of the applied channel code, it is known at the receiver that a third bit x_0 is given by the exclusive or operation of b_0 and b_1, that is, $x_0 = b_0 \oplus b_1$. An elementary question is how to generate extrinsic information on x_0 being given the observations y_0 and y_1 from the transmission channel. The state-of-the-art approach to this problem is either application of the sum-product algorithm to obtain $p(x_0|y_0, y_1)$, as it has been explained in Example 2.4.3.1, or, equivalenty, evaluation of

$$L^{\text{ext}}(x_0) = L^{\text{ch}}(b_0) \boxplus L^{\text{ch}}(b_1) \tag{5.23}$$

using the respective channel LLRs and the box-plus operation. Figure 5.11

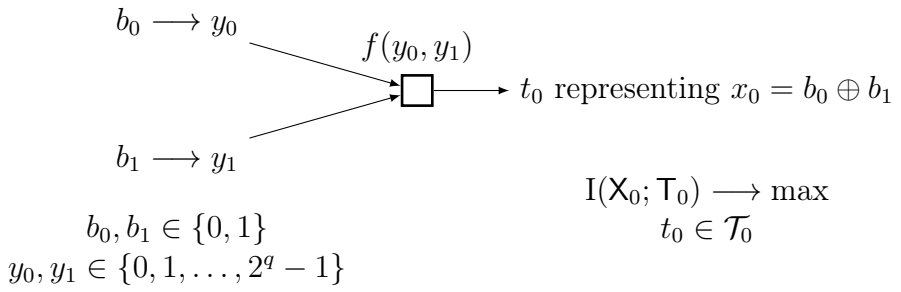


Figure 5.11: Elementary two-input lookup table replacing the box-plus operation in an Information Bottleneck LDPC decoder.

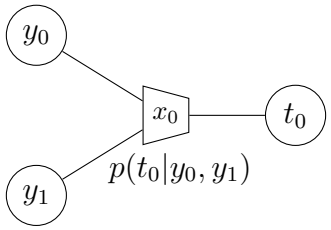


Figure 5.12: Information Bottleneck graph of an elementary two-input lookup table used to replace the box-plus operation of two LLRs.

shows a different perspective on the problem considered. In order to generate extrinsic information on X_0, one essentially requires a function $f(y_0, y_1)$ which delivers a $t_0 = f(y_0, y_1)$ in a way which preserves the maximum possible mutual information $\mathrm{I}(\mathsf{X}_0; \mathsf{T}_0) \leq \mathrm{I}(\mathsf{X}_0; \mathsf{Y}_0, \mathsf{Y}_1)$. The design of the considered two-input operation can be seen as an Information Bottleneck problem. Figure 5.12 shows the corresponding Information Bottleneck graph. The joint distribution required to design the shown Information Bottleneck node $p(t_0|y_0, y_1)$ is $p(x_0, y_0, y_1)$. It is obtained as

$$\begin{aligned} p(x_0, y_0, y_1) = & \sum_{b_0 \in \{0,1\}} \sum_{b_1 \in \{0,1\}} p(x_0, b_0, b_1, y_0, y_1) = \\ & \sum_{b_0 \in \{0,1\}} \sum_{b_1 \in \{0,1\}} p(x_0|b_0, b_1) p(b_0, y_0) p(b_1, y_1) = \\ & \sum_{b_0 \in \{0,1\}} \sum_{b_1 \in \{0,1\}} \delta(x_0 \oplus b_0 \oplus b_1) p(b_0, y_0) p(b_1, y_1) = \\ & \sum_{\substack{(b_0, b_1): \\ b_0 \oplus b_1 = x_0}} p(b_0, y_0) p(b_1, y_1). \end{aligned} \quad (5.24)$$

Please note that $p(b_0, y_0)$ and $p(b_1, y_1)$ are identical and known from the design algorithm of the channel output quantizer. By feeding the joint probability distribution $p(x_0, y_0, y_1)$ from Equation (5.24) to an Information Bottleneck algorithm, one can design the desired node $p(t_0|y_0, y_1)$ required to generate extrinsic information on x_0 from the quantization indices y_0 and y_1.

Applicability of Algorithm 4

Algorithm 4 can also be used to design $p(t_0|y_0, y_1)$. This first seems counterintuitive because the original design targets of Algorithm 4 were channel output quantizers with continuous quantization regions. It is not obvious, how this algorithm can also be applied to cluster pairs (y_0, y_1) of received quantization indices, as it is required here. In [58], it was shown that an optimum quantizer with contiguous quantization regions exists, if the event space of the observation is sorted by its channel LLR. Having access to joint distribution $p(x_0, y_0, y_1)$ from Equation (5.24), it is easy to calculate LLRs $L(x_0|y_0, y_1)$ as

$$L(x_0|y_0, y_1) = \log \frac{\Pr(\mathsf{X}_0 = 0|y_0, y_1)}{\Pr(\mathsf{X}_0 = 1|y_0, y_1)}. \tag{5.25}$$

These LLRs exactly equal $L^{\text{ext}}(x_0) = L^{\text{ch}}(b_0) \boxplus L^{\text{ch}}(b_1)$. Interpreting the pair (y_0, y_1) as the joined output of two channel uses, one can, therefore, order the pairs (y_0, y_1) by these LLRs. This shall be exemplified in Table 5.1 for an exemplary channel $p(y_k|b_k)$ with four different outputs. The upper part of Table 5.1 shows the considered transition probabilities $p(y_k|b_k)$. The lower left part shows the corresponding LLRs $L^{\text{ch}}(b_0), L^{\text{ch}}(b_1)$ and $L^{\text{ext}}(x_0)$ in a natural order of the pairs (y_0, y_1). As it can be seen, the LLRs $L^{\text{ext}}(x_0)$ are entirely unsorted in this order. On the right, the combinations (y_0, y_1) are rearranged, such that their corresponding LLRs $L^{\text{ext}}(x_0)$ are sorted increasingly. The only required step for making Algorithm 4 applicable is now to aggregate all pairs (y_0, y_1) resulting in the same LLR $L^{\text{ext}}(x_0)$ in a virtual channel output $y_{0,1}$. This step is also illustrated in the rightmost column of Table 5.1. The joint distribution $p(x_0, y_{0,1})$ is obtained by summation over $p(x_0, y_0, y_1)$ according to

$$p(x_0, y_{0,1}) = \sum_{\substack{(y_0,y_1):L^{\text{ext}}(x_0)= \\ L^{\text{ch}}(b_0)\boxplus L^{\text{ch}}(b_1)}} p(x_0, y_0, y_1). \tag{5.26}$$

Table 5.1: Exemplary transition probabilities $p(y_k|b_k)$ and corresponding LLRs

y_0 or y_1	0	1	2	3
$p(y_k\|\mathsf{B}_k = 0)$	0.0193	0.0873	0.2304	0.6625
$p(y_k\|\mathsf{B}_k = 1)$	0.6625	0.2304	0.0873	0.0193

y_0	y_1	$L^{\mathrm{ch}}(b_0)$	$L^{\mathrm{ch}}(b_1)$	$L^{\mathrm{ext}}(x_0)$	y_0	y_1	$L^{\mathrm{ext}}(x_0)$	$y_{0,1}$
0	0	−3.54	−3.54	+2.84	0	3	−2.84	0
0	1	−3.54	−0.97	+0.91	3	0	−2.84	0
0	2	−3.54	+0.97	−0.91	0	2	−0.91	1
0	3	−3.54	+3.54	−2.84	1	3	−0.91	1
1	0	−0.97	−3.54	+0.91	2	0	−0.91	1
1	1	−0.97	−0.97	+0.41	3	1	−0.91	1
1	2	−0.97	+0.97	−0.41	1	2	−0.41	2
1	3	−0.97	+3.54	−0.91	2	1	−0.41	2
2	0	+0.97	−3.54	−0.91	1	1	+0.41	3
2	1	+0.97	−0.97	−0.41	2	2	+0.41	3
2	2	+0.97	+0.97	+0.41	0	1	+0.91	4
2	3	+0.97	+3.54	+0.91	1	0	+0.91	4
3	0	+3.54	−3.54	−2.84	2	3	+0.91	4
3	1	+3.54	−0.97	−0.91	3	2	+0.91	4
3	2	+3.54	+0.97	+0.91	0	0	+2.84	5
3	3	+3.54	+3.54	+2.84	3	3	+2.84	5

This joint distribution can be fed to Algorithm 4 to obtain $p(t_0|y_{0,1})$. It is easy to obtain the desired $p(t_0|y_0, y_1)$ from $p(t_0|y_{0,1})$ by considering which equivalent pairs (y_0, y_1) were mapped onto $y_{0,1}$ at the beginning.

Table 5.2 shows the resulting $p(t_0|y_0, y_1)$ as a lookup table. For simplicity, the output cardinality $|\mathcal{T}_0|$ was chosen to be $|\mathcal{T}_0| = 4$, such that t_0 is from the same alphabet as y_0 and y_1. The rightmost column of the table also holds LLRs $L(x_0|t_0)$ calculated from $p(x_0|t_0)$ which is a side product of the applied Information Bottleneck algorithm. It shall be emphasized, that the LLRs are only provided for illustration purposes. The generation of extrinsic information using the designed node $p(t_0|y_0, y_1)$ can be carried out only using the integers t_0 from Table 5.2 and does not require these LLRs. Interestingly, however, the

Table 5.2: Lookup table $p(t_0|y_0, y_1)$ obtained using Algorithm 4.

y_0	y_1	t_0	$L(x_0\|t_0)$
0	0	3	+2.84
0	1	2	+0.81
0	2	1	−0.81
0	3	0	−2.84
1	0	2	+0.81
1	1	2	+0.81
1	2	1	−0.81
1	3	1	−0.81
2	0	1	−0.81
2	1	1	−0.81
2	2	2	+0.81
2	3	2	+0.81
3	0	0	−2.84
3	1	1	−0.81
3	2	2	+0.81
3	3	3	+2.84

LLRs $L(x_0|t_0)$ approximate the LLRs $L^{\text{ext}}(x_0)$ as it is revealed by comparing Tables 5.1 and 5.2. A formal description of this approximation can be found in [35].

Using the described technique, Algorithm 4 can be used to construct the Information Bottleneck node $p(t_0|y_0, y_1)$ from Figure 5.12. This node is a lookup table maximizing the mutual information between its output T_0 and the bit X_0 it is supposed to represent. When this lookup table is used to obtain extrinsic information, the most significant difference to the state-of-the-art is that inputs and outputs are from the set of integers and, therefore, no processing of any real numbers is required.

Decomposing Node Operations

So far, only two inputs can be processed by Information Bottleneck node $p(t_0|y_0, y_1)$ from Figure 5.12. In a particular iteration i of an LDPC decoder,

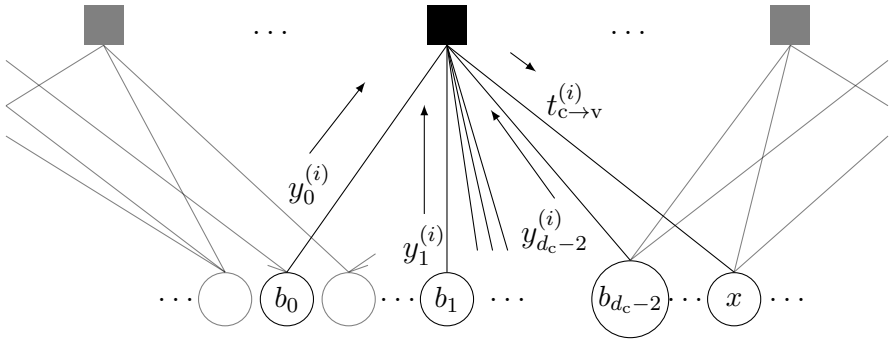


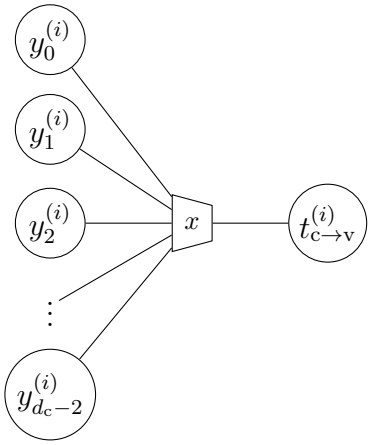
Figure 5.13: Message generation of a check node for one particular target edge. The check node processes messages $y_n^{(i)}$ to generate an outgoing message $t^{(i)}_{\mathrm{c}\to\mathrm{v}}$ for one particular target edge.

however, a degree d_c node has to process $d_\mathrm{c}-1$ incoming messages for generation of extrinsic information to be propagated along a particular target edge. This is illustrated in Figure 5.13. Let us consider message generation for the depicted codeword bit denoted by x in Figure 5.13 which is the codeword bit connected to the rightmost edge of the central check node from Figure 5.13. Please recall Equation (2.58), which defines the binary sum x_m over $m+2$ bits connected to a check node. With this definition, due to the structure of the applied LDPC code, this bit is given by the sum over $d_\mathrm{c}-1$ codeword bits $b_0, b_1, \ldots, b_{d_\mathrm{c}-2}$. Thus, we have

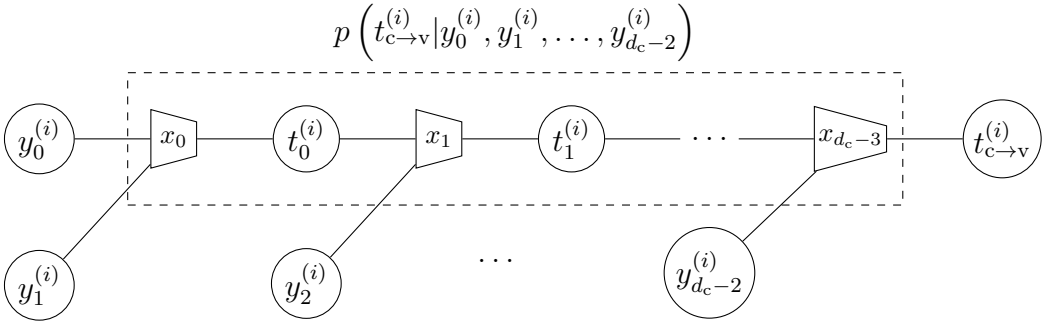
$$x = x_{d_\mathrm{c}-3} = \sum_{n=0}^{d_\mathrm{c}-2} \oplus\, b_n. \tag{5.27}$$

In order to generate extrinsic information on this bit, the check node has to process the incoming messages $y_0^{(i)}, y_1^{(i)}, \ldots, y_{d_\mathrm{c}-2}^{(i)}$ in iteration i as it is depicted. The corresponding outgoing message is denoted by $t^{(i)}_{\mathrm{c}\to\mathrm{v}}$. Message generation has to be carried out in each iteration for all other connected edges of the node equivalenty, by excluding the respective incoming message that reaches the node along the target edge.

Figure 5.14a illustrates the task, the check node is facing in an Information Bottleneck graph. As it is depicted there, the outgoing message $t^{(i)}_{\mathrm{c}\to\mathrm{v}}$ of the node shall be highly informative about the codeword bit x. Unfortunately, designing and storing the Information Bottleneck node from Figure 5.14a has exponential complexity in the number of inputs, as it has been explained in Section 3.4. Consider, for example, $q = 4$ bit quantization and the initial



(a) Information Bottleneck graph of a check node operation with closed node $p\left(t_{\mathrm{c}\to\mathrm{v}}^{(i)}|y_0^{(i)}, y_1^{(i)}, \ldots, y_{d_\mathrm{c}-2}^{(i)}\right)$.



(b) Information Bottleneck graph of a check node operation with opened node $p\left(t_{\mathrm{c}\to\mathrm{v}}^{(i)}|y_0^{(i)}, y_1^{(i)}, \ldots, y_{d_\mathrm{c}-2}^{(i)}\right)$.

Figure 5.14: Information Bottleneck graphs of a check node lookup table.

decoding iteration $i = 0$, where all incoming messages of the check node come directly from the channel output quantizer. In this situation, every input shown in Figure 5.14a can take 16 different values. Already for a relatively small check node degree of $d_\mathrm{c} = 6$, there are more than 1 million possible input combinations of the node.

In [28], Kurkoski *et al.* proposed to deal with the exponential complexity in the node degree, by splitting up the node operation in a series of concatenated two-input operations. This strategy effectively corresponds to opening the Information Bottleneck node $p\left(t_{\mathrm{c}\to\mathrm{v}}^{(i)}|y_0^{(i)}, y_1^{(i)}, \ldots, y_{d_\mathrm{c}-2}^{(i)}\right)$, as it is shown in Figure 5.14b. The shown structure decomposes the check node operation into

a series of two-input operations. This decomposition has a major effect on the number of lookup table entries required to implement the Information Bottleneck node. When choosing the output cardinalities of all applied Information Bottleneck algorithms as $|\mathcal{T}_{\text{chk}}| = |\mathcal{Y}_{\text{chan}}| = 2^q$, the opened node structure from Figure 5.14b requires a total of $(d_{\text{c}} - 2) \cdot 2^{2q}$ entries. For the example from above, this results in a reduction from more than 1 million table entries to $4 \cdot 2^8 = 1024$.

The joint distribution required to design the first Information Bottleneck node $p\left(t_0^{(0)}|y_0^{(0)}, y_1^{(0)}\right)$ for decoding iteration $i = 0$ is obtained equivalently as in Equation (5.24) by making use of the output distributions of the quantizer design algorithm. For the other concatenated nodes $p\left(t_m^{(i)}|t_{m-1}^{(i)}, y_{m+1}^{(i)}\right)$, they can be derived similarly and are given by

$$p\left(x_m, t_{m-1}^{(i)}, y_{m+1}^{(i)}\right) = \sum_{\substack{(x_{m-1}, b_{m+1}):\\ x_{m-1} \oplus b_{m+1} = x_m}} p\left(x_{m-1}, t_{m-1}^{(i)}\right) p\left(b_{m+1}, y_{m+1}^{(i)}\right). \quad (5.28)$$

for all $m \in \{1, 2, \ldots, d_{\text{c}} - 3\}$. The distribution $p\left(x_{m-1}, t_{m-1}^{(i)}\right)$ in Equation (5.28) is delivered by the Information Bottleneck algorithm used to design the respective ancestor node. Also here, in the initial decoding iteration $i = 0$ the joint distribution $p\left(b_{m+1}, y_{m+1}^{(0)}\right)$ in Equation (5.28) is known from the Information Bottleneck algorithm applied for quantizer design. It will be explained in Section 5.2.3, how $p\left(b_{m+1}, y_{m+1}^{(i)}\right)$ is obtained also for the iterations $i > 0$.

It has already been mentioned in Chapter 4, that opening nodes in an Information Bottleneck graph offers a degree of freedom in the choice of the compression cardinalities of all nodes designed in an opened node structure. For simplicity, it is assumed that all compression variables inside the opened node from Figure 5.14b are from the same alphabet $\mathcal{T}_{\text{chk}} = \{0, 1, \ldots, |\mathcal{T}_{\text{chk}}| - 1\}$. This, however, again is a design choice which offers a huge flexibility in the design of the opened node structure. As a result, the outgoing message from the check nodes to the variable nodes also is from this set, that is, $t_{\text{c}\to\text{v}}^{(i)} \in \mathcal{T}_{\text{chk}}$.

The decomposition from Figure 5.14b has the desirable characteristic that it is applicable to any node degree d_{c} with the presented equations in an identical manner. However, for a certain considered node degree d_{c}, typically also other possible decompositions exist which might be favorable from an implementation perspective. Consider, for example, the Information Bottleneck graph

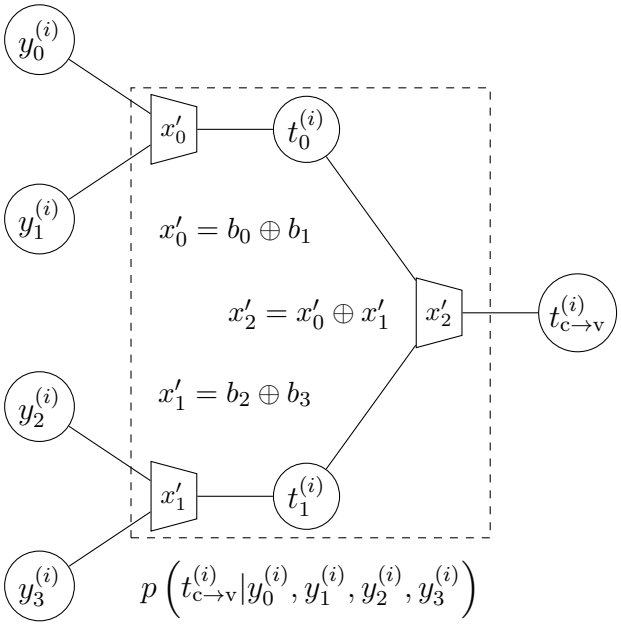


Figure 5.15: Information Bottleneck graph of a degree $d_\mathrm{c} = 5$ check node operation with opened node $p\left(t_{\mathrm{c}\to\mathrm{v}}^{(i)} | y_0^{(i)}, y_1^{(i)}, y_2^{(i)}, y_3^{(i)}\right)$. The node is opened in a binary tree structure.

shown in Figure 5.15 for a degree $d_\mathrm{c} = 5$ check node which has to process $d_\mathrm{c} - 1 = 4$ incoming messages. The shown decomposition of the check node operation is a binary tree. An advantage of the shown decomposition is, that the joint distributions $p\left(x'_0, y_0^{(i)}, y_1^{(i)}\right)$ and $p\left(x'_1, y_2^{(i)}, y_3^{(i)}\right)$ are identical. As a result, the nodes $p\left(t_0^{(i)} | y_0^{(i)}, y_1^{(i)}\right)$ and $p\left(t_1^{(i)} | y_2^{(i)}, y_3^{(i)}\right)$ can use the same lookup table. This is never the case in the serial decomposition from Figure 5.12. A full binary tree decomposition as shown here, however, is in its pure form only possible for a number of inputs which is a power of two.

Interestingly, the choice of the structure of the opened node between both presented structures has almost no influence on the output information of the node. This is visualized in Figure 5.16, where the output information $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{c}\to\mathrm{v}}^{(0)}\right)$ over the channel noise variance $\sigma_{\tilde{n}}^2$ is plotted for various node degrees d_c which allow for a full binary tree decomposition and both structures. In this experiment, all compression cardinalities of the Information Bottleneck nodes involved in the respective decomposition were chosen identical as $|\mathcal{T}_\mathrm{chk}| = 16$. As it can be seen in Figure 5.16, the curves referring to both types of decompositions match.

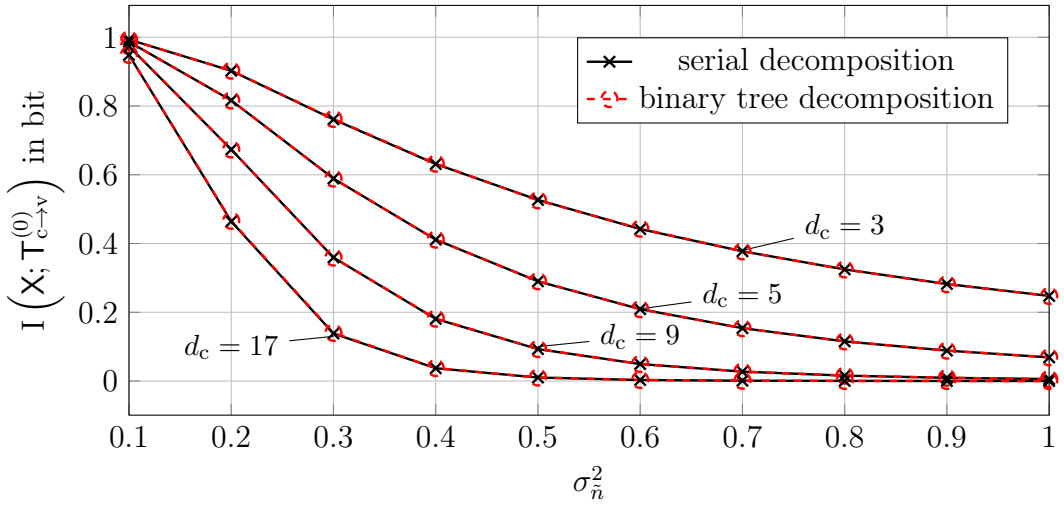


Figure 5.16: Comparison of the output information $\mathrm{I}(\mathsf{X}; \mathsf{T}_{c\to v}^{(0)})$ for a serial node decomposition of a check node operation and a binary tree decomposition for various node degrees d_c and channel noise variance $\sigma_{\tilde{n}}^2$.

It will be discussed later in Chapter 6, that when considering an implementation, more relevant aspects for an optimum choice of a node decomposition can be taken into account. However, this choice is highly code and implementation specific and the equations involved in the decoder design only vary marginally. Therefore, in this section, we will stick with the decomposition from Figure 5.14b originally proposed by Kurkoski *et al.* in [28] which works for any node degree.

So far, it has been described in this section, how a degree d_c check node for the first decoding iteration $i = 0$, where all incoming messages of the check node come directly from the channel output quantizer, can be designed and visualized in an Information Bottleneck graph. It is important that the Information Bottleneck algorithm applied to design of the last concatenated node in the respective opened node structure delivers $p\left(x, t_{c\to v}^{(i)}\right) = p\left(x|t_{c\to v}^{(i)}\right) p\left(t_{c\to v}^{(i)}\right)$ as a side product. The output message $t_{c\to v}^{(i)}$ is propagated back into the decoding graph in the current step of message passing decoding, and has to be processed by a connected variable node, i.e., $t_{c\to v}^{(i)}$ is an observation of the variable node. Therefore, the joint distribution $p\left(x, t_{c\to v}^{(i)}\right)$ is required to design variable node operations with the Information Bottleneck method in the following.

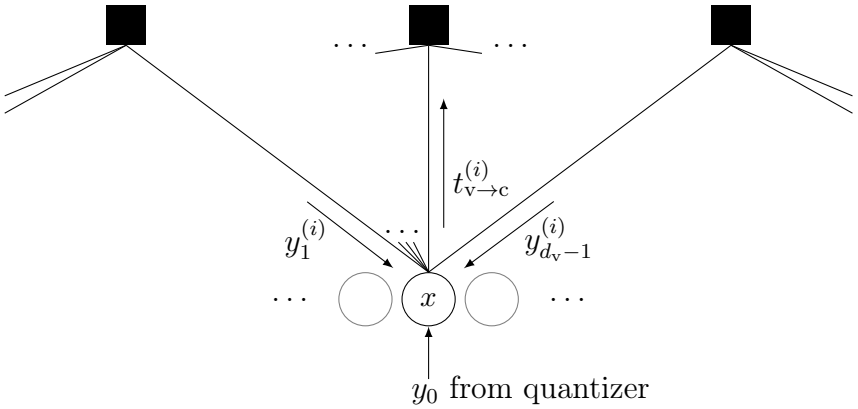


Figure 5.17: Message generation of a variable node for one particular target edge. The variable node processes messages $y_n^{(i)}$ to generate an outgoing message $t_{\mathrm{v}\to\mathrm{c}}^{(i)}$ for one particular target edge.

5.2.2 Design of Variable Node Decoding Functions with the Information Bottleneck Method

Once all check nodes have generated their outgoing integer messages $t_{\mathrm{c}\to\mathrm{v}}^{(i)} \in \mathcal{T}_{\mathrm{chk}}$ in decoding iteration i, these messages arrive at the variable nodes, where they have to be processed for message generation. As a consequence, the outgoing messages $t_{\mathrm{c}\to\mathrm{v}}^{(i)} \in \mathcal{T}_{\mathrm{chk}}$ from the check nodes build the incoming messages $y_n^{(i)}$ of the variable nodes, as it is illustrated for an exemplary variable node in Figure 5.17. In the situation depicted in Figure 5.17, again message generation of an outgoing message $t_{\mathrm{v}\to\mathrm{c}}^{(i)}$ for one particular target edge is being considered. The variable node has to process $d_\mathrm{v} - 1$ messages $y_1^{(i)}, y_2^{(i)}, \ldots, y_{d_\mathrm{v}-1}^{(i)}$ arriving from its connected check nodes and a message from the channel to generate the message considered. Of course, message generation has to be carried out analogously for all other connected edges.

An apparent difference to the message generation process at the check nodes is that all incoming messages represent the same codeword bit. This codeword bit is denoted by x in the following, as it is illustrated in Figure 5.17. Another important difference to the check nodes is, that the variable node has access to the channel message coming directly from the channel output quantizer. This message is denoted by y_0, as it is depicted in Figure 5.17. Please note that y_0 does not depend on the decoder iteration i. Hence, the variable node has to process a total of d_v messages to generate the messages passed into

the decoding graph during the iterative decoding process. A slight difference appears when the final bit decision shall be performed. Then the variable node has to process all d_v incoming messages from its connected check nodes and the channel message as a posteriori instead of extrinsic information shall be obtained. This is depicted in Figure 5.18a in an Information Bottleneck graph. As it can be seen there, the node uses a mutual information maximizing lookup table $p\left(t_{\mathrm{v}\to\mathrm{c}}^{(i)}|y_0, y_1^{(i)}, \ldots, y_{d_\mathrm{v}-1}^{(i)}\right)$ to process the channel message y_0 and $d_\mathrm{v}-1$ incoming messages $y_1^{(i)}, y_2^{(i)}, \ldots, y_{d_\mathrm{v}-1}^{(i)}$ from the check nodes to generate $t_{\mathrm{v}\to\mathrm{c}}^{(i)}$ which shall be highly informative about the codeword bit x. The message $t_{\mathrm{v}\to\mathrm{c}}^{(i)} \in \mathcal{T}_\mathrm{var}$ is passed back to the check nodes during iterative decoding.

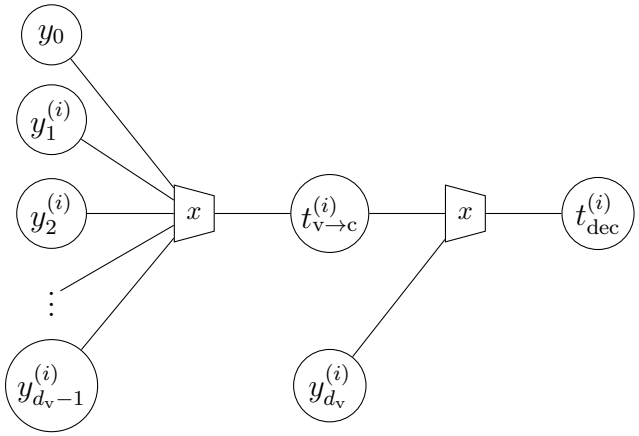
For the final bit decision, another incoming message $y_{d_\mathrm{v}}^{(i)}$ has to be processed, which is handled by the concatenated Information Bottleneck node $p\left(t_\mathrm{dec}^{(i)}|t_{\mathrm{v}\to\mathrm{c}}^{(i)}, y_{d_\mathrm{v}}^{(i)}\right)$. The final bit decision in iteration i has to be performed based on $t_\mathrm{dec}^{(i)}$.

Also the variable node operation can be decomposed into two-input operations, as it is shown in Figure 5.18b. In contrast to the check node operation, where all Information Bottleneck nodes had different relevant random variables X_m, for the variable node all concatenated nodes inside the opened node structure have the same relevant random variable, as all incoming messages represent the same codeword bit. Algorithm 4 can also be applied to design the variable node operation, as it has been explained in Section 5.2.1. The joint distributions required to design the Information Bottleneck nodes from Figure 5.18b are obtained as

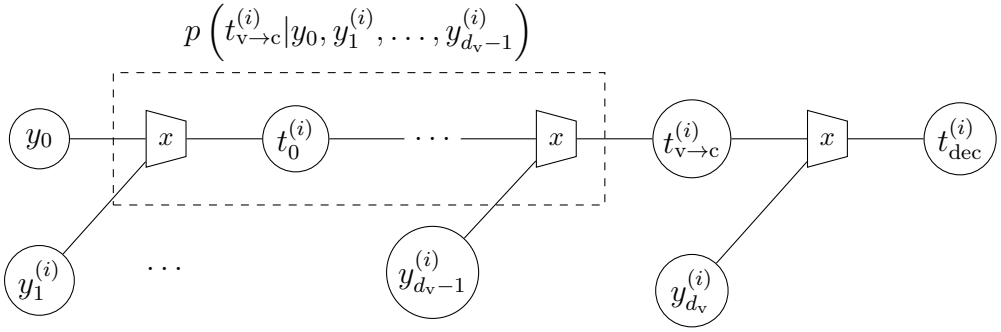
$$p\left(x, y_0, y_1^{(i)}\right) = p(x, y_0) p\left(x, y_1^{(i)}\right) \frac{1}{p(x)} \tag{5.29}$$

for the leftmost node $p\left(t_0^{(i)}|y_0, y_1^{(i)}\right)$. In Equation (5.29), the joint distribution $p(x, y_0)$ is delivered by the Information Bottleneck algorithm used for quantizer design, as y_0 directly comes from the channel output quantizer in every iteration i. The joint probability distribution $p\left(x, y_1^{(i)}\right)$ is given by $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(i)}\right)$ from the design of the check node operation for this iteration. For the following concatenated nodes $p\left(t_m^{(i)}|t_{m-1}^{(i)}, y_{m+1}^{(i)}\right)$ from Figure 5.18b, the required joint distributions are given by

$$p\left(x, t_{m-1}^{(i)}, y_{m+1}^{(i)}\right) = p\left(x, t_{m-1}^{(i)}\right) p\left(x, y_{m+1}^{(i)}\right) \frac{1}{p(x)}. \tag{5.30}$$



(a) Information Bottleneck graph of a variable node operation with closed node $p\left(t_{\mathrm{v}\to\mathrm{c}}^{(i)}|y_0, y_1^{(i)}, \ldots, y_{d_\mathrm{v}-1}^{(i)}\right)$ and concatenated node $p\left(t_{\mathrm{dec}}^{(i)}|t_{\mathrm{v}\to\mathrm{c}}^{(i)}, y_{d_\mathrm{v}}^{(i)}\right)$. The concatenated node is required for the final bit decision.



(b) Information Bottleneck graph of a variable node operation with opened node $p\left(t_{\mathrm{v}\to\mathrm{c}}^{(i)}|y_0, y_1^{(i)}, \ldots, y_{d_\mathrm{v}-1}^{(i)}\right)$.

Figure 5.18: Information Bottleneck graphs of a variable node lookup table.

The joint probability distribution $p\left(x, t_{m-1}^{(i)}\right)$ in Equation (5.30) is delivered by the Information Bottleneck algorithm applied to design the respective ancestor node in the concatenated structure from Figure 5.18b. The joint distribution $p\left(x, y_{m+1}^{(i)}\right)$ in Equation (5.30) is given by $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(i)}\right)$ from the design of the check node operation as the incoming messages $y_{m+1}^{(i)}$ are incoming from the connected check nodes. For simplicity, again all output cardinalities of the Information Bottleneck nodes shown in Figure 5.18b are chosen identically, such that all intermediate results, $t_{\mathrm{v}\to\mathrm{c}}^{(i)}$ and $t_{\mathrm{dec}}^{(i)}$ are from the identical set $\mathcal{T}_{\mathrm{var}} = \{0, 1, \ldots, |\mathcal{T}_{\mathrm{var}}| - 1\}$. The message $t_{\mathrm{v}\to\mathrm{c}}^{(i)}$ is passed to a connected check

node in the next decoding iteration. It is, therefore, quite intuitive that the joint distribution $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(i)}\right)$ is required to construct an updated check node operation for iteration $i+1$. This will be explained in Section 5.2.3.

In some implementations, for example, the ones to be presented in Chapter 6, LDPC decoders have to be able to perform their final bit decision in each decoding iteration i to stop the decoding once a valid codeword has been found. In this case, the rightmost Information Bottleneck node $p\left(t_{\mathrm{dec}}^{(i)} | t_{\mathrm{v}\to\mathrm{c}}^{(i)}, y_{d_{\mathrm{v}}-1}^{(i)}\right)$ from Figure 5.18b has to be constructed for each decoding iteration i. The Information Bottleneck algorithm applied to design this node also delivers $p\left(x, t_{\mathrm{dec}}^{(i)}\right)$. This joint distribution will be used to investigate the design process of the iterative decoder later. It allows to calculate the mutual information $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{dec}}^{(i)}\right)$. This mutual information reflects the reliability of the decision bits of the LDPC decoder in iteration i for a code with infinite length.

5.2.3 Decoder Design with Discrete Density Evolution

The decoder construction process described above started in the initial decoding iteration $i = 0$ at the check nodes. We have used the joint distributions $p\left(b_n, y_n^{(0)}\right)$ for construction of the corresponding check node operation in an Information Bottleneck graph. These distributions were known from the design algorithm of the channel output quantizer, as the incoming messages of the check node were given by output indices from the channel output quantizer for $i = 0$. The resulting joint distribution $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(0)}\right)$ from the check node design process has been used to construct the variable node operation for $i = 0$ afterwards. Again, the involved Information Bottleneck algorithm used to design the last concatenated node in the respective opened node structure delivered the distribution $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(0)}\right)$ as a side product.

In the next step of decoding, integer messages are passed back from the variable nodes to the check nodes which refer to this joint distribution $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(0)}\right)$. As a result, in iteration $i = 1$, an adapted check node lookup table has to be designed. Its design is completely analogous to the one described in Section 5.2.1 with the only difference, that the distributions $p\left(b_n, y_n^{(1)}\right)$ in Equation (5.28) are now given by $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(0)}\right)$. After the Information Bottleneck check node design for iteration $i = 1$ has finished, one also has access to $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(1)}\right)$ which

can in turn be used to design adapted variable node lookup tables equivalently to Section 5.2.2 and to obtain $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(1)}\right)$ as a side product.

The described process can be repeated iteratively, for a desired maximum number $i_{\max}$ of decoding iterations to construct the variable and the check node operations for all performed decoder iterations. In this process, joint distributions $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(i)}\right)$ are being passed from the check node operation construction step to the variable node operation construction step. The variable node operation construction process in turn delivers joint distributions $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(i)}\right)$ to the check node operation construction step. This approach is very similar to the density evolution technique described in Section 2.3.4.

Density evolution has the purpose of determining the decoding threshold of LDPC ensembles under LLR based decoding. Here, it is primarily used to determine the lookup table based node operations involved in the iterative decoding process by repeated application of Information Bottleneck algorithms. Due to the fact that all involved random variables in the Information Bottleneck decoder are discrete, the considered density evolution like technique is termed *discrete* density evolution in the following. It turns out that discrete density evolution can also be applied to determine the decoding thresholds of regular LDPC ensembles under the proposed decoding method as it has first similarly been proposed in [28]. For this purpose, one requires the joint distribution $p\left(x, t_{\mathrm{dec}}^{(i)}\right)$ that has already been mentioned in Section 5.2.2. The corresponding preserved relevant information $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ for $i \to \infty$ tells, whether or not an LDPC ensemble can correct all errors under the proposed lookup table based decoding. If $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{dec}}^{(\infty)}\right) = 1$ bit, the codeword bit is known with absolute certainty and, therefore, the LDPC ensemble can correct all errors. If, in contrast, $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{dec}}^{(\infty)}\right) < 1$ bit, it cannot correct all errors. Determining the decoding threshold of a regular LDPC ensemble under Information Bottleneck decoding shall be illustrated in the following example, which compares to Example 2.3.4.1 for density evolution under LLR based decoding.

Example 5.2.3.1. Consider the $(d_{\mathrm{v}}, d_{\mathrm{c}}) = (3, 6)$ regular LDPC ensemble. Figure 5.19 shows the evolution of the variable node output information $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ under AWGN with BPSK modulation for the proposed Information Bottleneck lookup table decoding for several E_b/N_0 as a function of the decoder iteration i. In the considered decoder the channel messages, the variable node messages

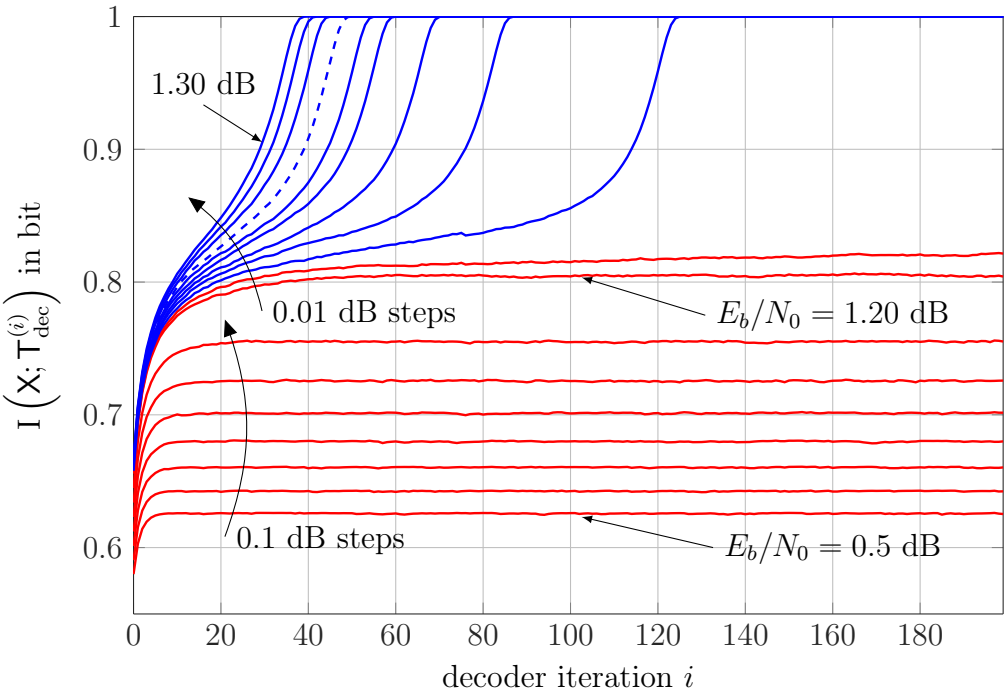


Figure 5.19: Evolution of the variable node output information over the decoder iterations for the $(3, 6)$ regular LDPC ensemble under Information Bottleneck decoding.

and the check node messages all use the same alphabet $\mathcal{Y}_{\mathrm{chan}} = \mathcal{T}_{\mathrm{var}} = \mathcal{T}_{\mathrm{chk}} = \{0, 1, \ldots, 15\}$ corresponding to a bit width of $q = 4$ bit. As it can be seen, for $E_b/N_0 < 1.22$ dB the output information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ of the variable nodes saturates to a value smaller than 1 bit. In contrast, for $E_b/N_0 \geq 1.22$ dB, the maximum value of 1 bit is achieved. This allows for the conclusion, that the decoding treshold of the $(3, 6)$ regular LDPC ensemble under the considered Information Bottleneck decoding with 4 bit messages lies between $E_b/N_0 = 1.21$ and $E_b/N_0 = 1.22$ dB.

Recall that Example 2.3.4.1 has shown that under LLR based belief propagation decoding, the respective critical E_b/N_0 lies between $E_b/N_0 = 1.1$ and 1.12 dB. As a result, the difference in the decoding thresholds between floating point belief propagation decoding and the proposed Information Bottleneck decoding with 4 bit messages is just about 0.1 dB over E_b/N_0 which is a remarkable result. —*End of the example.*

Example 5.2.3.1 has shown that the Information Bottleneck based lookup table decoding has a decoding threshold which is remarkably close to the one of quasi continuous belief propagation decoding. This already indicates that the proposed decoding approach might offer good performance with unsigned integer messages that can be stored very compactly. The bit error rate performance of the proposed decoders for finite length LDPC codes will be investigated in Section 5.2.5. In the remainder of the thesis, the presented approach to LDPC decoding will be termed *Information Bottleneck decoding*.

5.2.4 Mismatched Quantizer and Decoder Design

Since the proposed design technique of the discrete node operations with the Information Bottleneck method is computationally quite expensive, it is desirable to generate the proposed LDPC decoders offline and only for one particular E_b/N_0. Moreover, the channel output quantizer should also be used mismatched, to avoid the need of matching it to the channel E_b/N_0 on the fly. Then the channel output quantizer and the designed lookup tables in the discrete LDPC decoders have to be used mismatched to the actual E_b/N_0. This raises the question how to identify a particular E_b/N_0 used for the generation of the quantizer and the lookup tables which offers good performance if the decoder is used mismatched to the actual channel quality. The E_b/N_0 the components are generated for will be termed the *design-*E_b/N_0 from now on. In the following, a simple method to identify a design-E_b/N_0 for the quantizer and the decoder which offers good performance also if the decoder is used mismatched to the actual E_b/N_0 will be described.

It has been discussed above that the decoder construction process allows to determine $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ in each decoder iteration and that this information reflects the reliability of the bit decision of the decoder in iteration i for a code with infinite length. In practice, the maximum number of decoder iterations $i_{\max}$ has to be limited to meet power consumption or delay constraints. It has been found experimentally, that a decoder which is designed such that $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i_{\max}-1)}\right)$ approaches 1 bit in the design process of the decoder offers good performance over a wide range of E_b/N_0, even if the decoder and the channel output quantizer are used mismatched to the design-E_b/N_0. For example, Figure 5.19 suggests that if a maximum of $i_{\max} = 50$ decoder iterations shall

be performed, a decoder for a $(3, 6)$ regular LDPC code with a design-E_b/N_0 of 1.27 dB (dashed curve in Figure 5.19) can be expected to offer good performance. Please note, however, that this design recommendation is a heuristical rule, which does not describe the actual output information of a practical decoder. This is due to the fact that the decoder is used mismatched to the design-E_b/N_0 and the applied code normally has cycles in its Tanner graph which are not modelled in the decoder construction process. The effects of the design-E_b/N_0 on the decoding performance will be illustrated in the following section in detail.

5.2.5 Performance and Discussion

In this section, the performance of Information Bottleneck LDPC decoding for regular LDPC codes under AWGN with BPSK modulation shall be evaluated and compared with double precision belief propagation and min-sum decoding. Moreover, the influence of the design-E_b/N_0 shall be studied. A challenging benchmark is to approach or even achieve the decoding performance of double precision belief propagation decoding with Information Bottleneck decoding while using messages that can be stored very compactly using only a few bits in the hardware. For simplicity, in most Information Bottleneck decoders applied in this section the message alphabets $\mathcal{Y}_{\text{chan}}, \mathcal{T}_{\text{chk}}$ and $\mathcal{T}_{\text{var}}$ are chosen to be identical, that is, $\mathcal{Y}_{\text{chan}} = \mathcal{T}_{\text{chk}} = \mathcal{T}_{\text{var}} = \{0, 1, \ldots, 2^q - 1\}$, such that all messages exchanged in the iterative decoding process are q bit integers. First, the message bit width $q = 4$ is used. It will be shown that this bit width is sufficient to achieve performance close to that of double precision belief propagation decoding in the considered scenario. The influence of choosing different message alphabet cardinalities for $\mathcal{Y}_{\text{chan}}$, $\mathcal{T}_{\text{chk}}$ and $\mathcal{T}_{\text{var}}$ will be analyzed towards the end of this section.

The following regular LDPC codes will be considered for the evaluation of the bit error rate performance over E_b/N_0 in this section:

1. The length $8,000$ $(3, 6)$ regular LDPC code 8000.4000.3.483 from [68] with code rate $R = 0.5$. This code is termed *regular code a)*.

2. The length $2,640$ $(3, 6)$ regular LDPC code Margulis2640.1320.3 from [68] with code rate $R = 0.5$. This code is termed *regular code b)*.

Table 5.3: Code parameters of the applied regular LDPC codes

Code	node degrees $(d_\mathrm{v}, d_\mathrm{c})$	code rate R	codeword length
regular code a)	$(3, 6)$	0.5	$8,000$
regular code b)	$(3, 6)$	0.5	$2,640$
regular code c)	$(3, 27)$	0.889	999
regular code d)	$(5, 8)$	0.375	$20,000$

3. The length 999 $(3, 27)$ regular LDPC code 999.111.5565 with code rate $R \approx 0.889$ from [68]. This code is termed *regular code c)*.

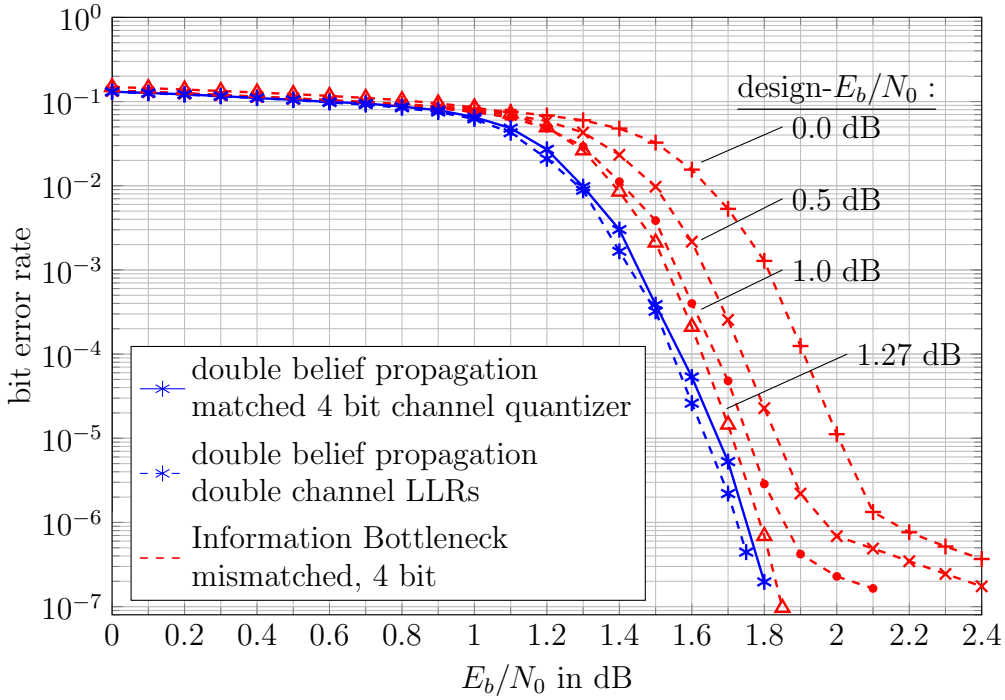
4. A length $20,000$ $(5, 8)$ regular LDPC code with code rate $R = 0.375$ which has been constructed randomly. This code is termed *regular code d)*.

The codes were deliberately chosen to have different codeword lengths and code rates. The parameters of the used codes are consolidated in Table 5.3 for a quick overview.

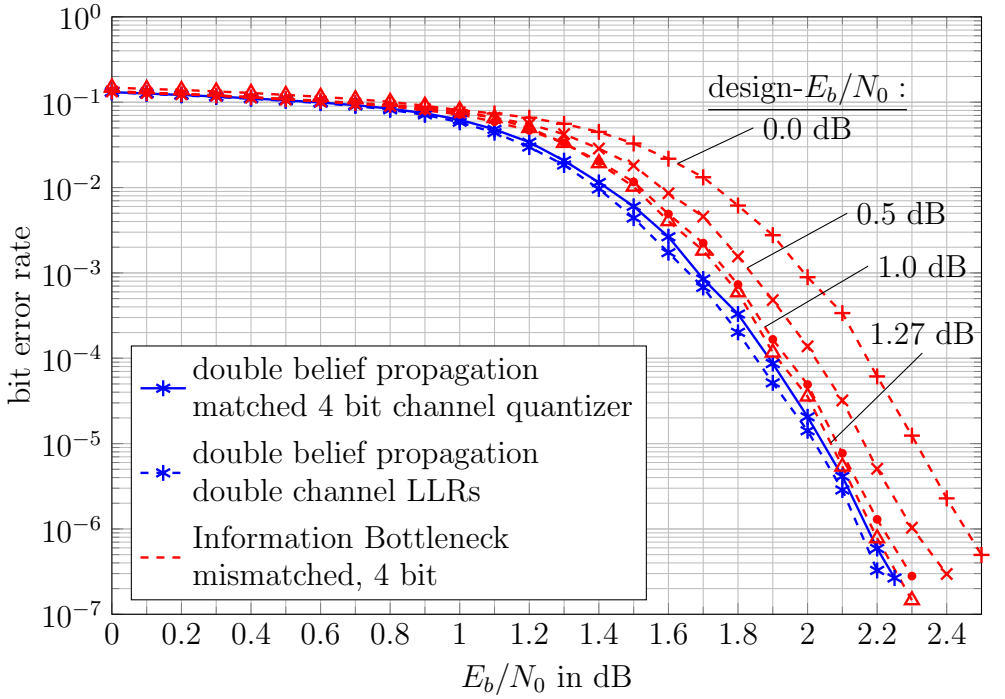
Performance and Influence of the Design-E_b/N_0

Figure 5.20 illustrates the influence of the design-E_b/N_0 on the performance of the resulting Information Bottleneck decoders with BPSK modulation under AWGN for a maximum of $i_\mathrm{max} = 50$ decoding interations. The applied codes are *regular code a)* in Figure 5.20a at the top and *regular code b)* in Figure 5.20b at the bottom. In both figures, curves for double precision belief propagation decoding with and without a 4 bit channel output quantizer serve as references. The channel output quantizers were constructed using Algorithm 4. For the belief propagation decoder, the quantizer was matched to the actual E_b/N_0 on the channel, in order to provide matched LLRs to the decoder. This was not the case for the applied $q = 4$ bit Information Bottleneck decoders, which just used a static quantizer and lookup tables only matched to their respective design-E_b/N_0.

It can be concluded from Figure 5.20 that the design-E_b/N_0 has a strong influence on the decoder performance of the Information Bottleneck decoders. The shown curves for design-$E_b/N_0 < 1.27$ dB in Figure 5.20 show that choosing the design-E_b/N_0 too small results in losses over E_b/N_0 in the waterfall



(a) Bit error rate of *regular code a)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.



(b) Bit error rate of *regular code b)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.

Figure 5.20: Bit error rates of *regular code a)* and *regular code b)* with BPSK modulation under AWGN over E_b/N_0. The influence of the design-E_b/N_0 on the performance of the Information Bottleneck decoders becomes visible.

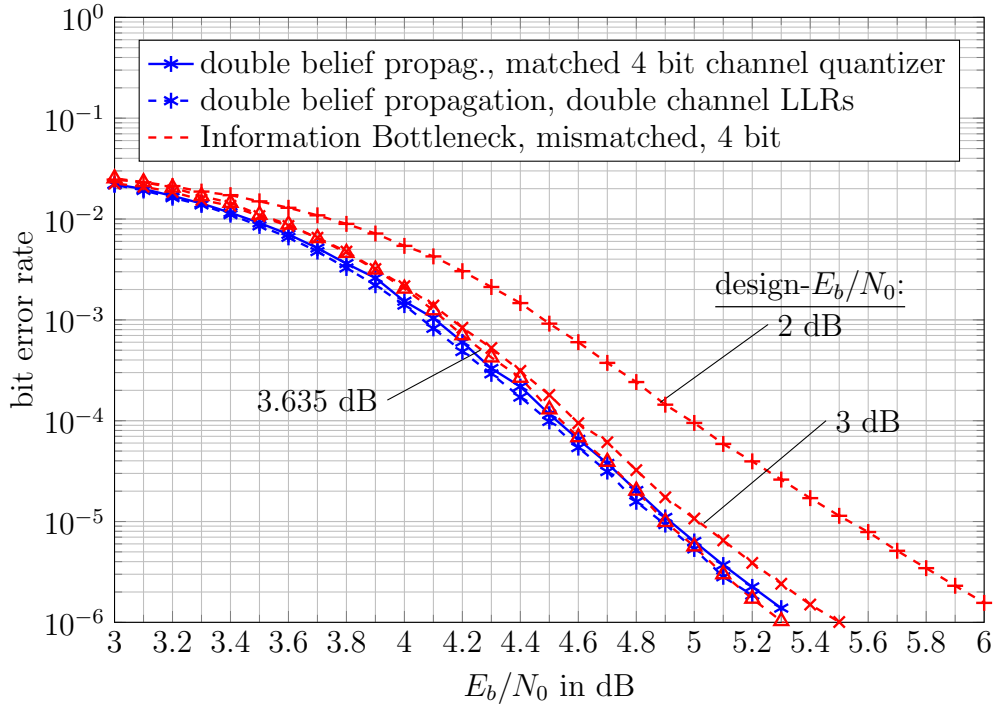
region of the code and also in an early error floor. The effects of choosing the design-E_b/N_0 too high will be studied later.

Recalling Figure 5.19 tells that the decoder which offers best performance among the decoders investigated in Figure 5.20 is designed such that during the design process, the variable node output information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ approaches 1 bit in the final decoder iteration. The decoder with design-$E_b/N_0 = 1.27$ dB offers performance enormously close to that of double precision belief propagation decoding with and without a quantized channel output. The loss of the best found Information Bottleneck decoder with respect to the double precision belief propagation competitor with channel output quantization is smaller than 0.1 dB over E_b/N_0 in the waterfall regions of both investigated $(3, 6)$ regular codes.

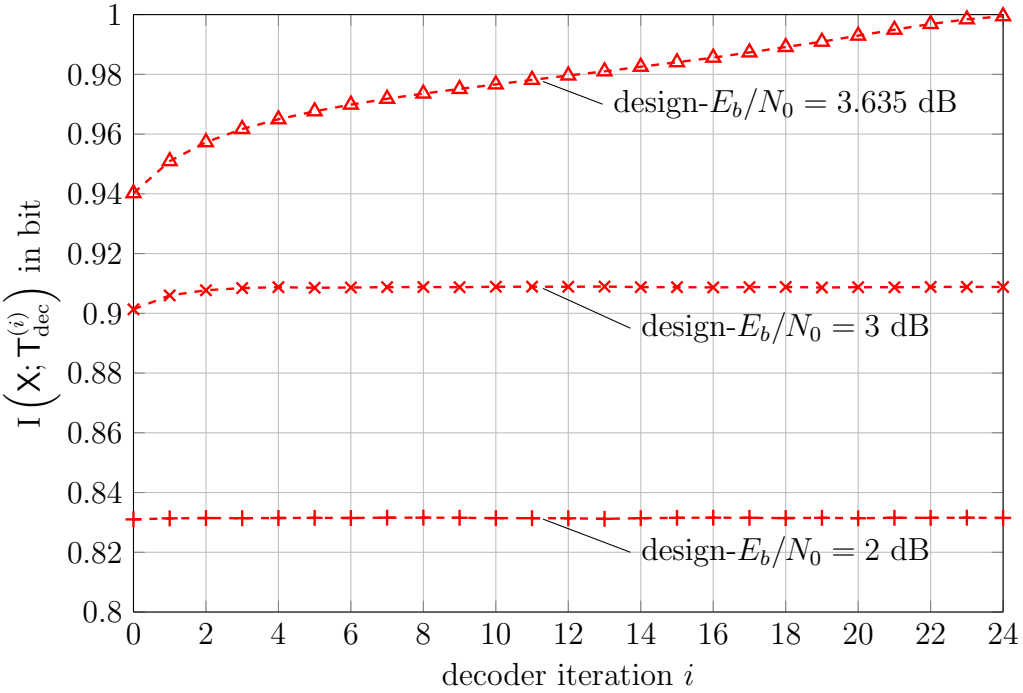
The Figures 5.21 and 5.22 similarly investigate the bit error rate performances of Information Bottleneck LDPC decoders constructed for various design-E_b/N_0 for codes *c)* and *d)* and different numbers of maximum decoder iterations $i_{\max}$. For the (3,27) *regular code c)* in Figure 5.21 the maximum number of iterations was $i_{\max} = 25$ and for the (5,8) *regular code d)* in Figure 5.22 it was set to $i_{\max} = 40$.

In the top plot of both figures, the bit error rate performances of the respective Information Bottleneck decoders constructed with different design-E_b/N_0 and belief propagation decoders with and without channel output quantization are shown. The bottom plot of both figures shows the variable node output information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ of the decoders with distinct design-E_b/N_0 from the design process of the decoders used above over the decoder iterations.

Again, both figures reveal that the Information Bottneck decoders designed such that $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ approaches 1 bit in the final decoder iteration during the design process of the decoder perform best. Interestingly, for the very short and high rate *regular code c)*, the best found Information Bottleneck decoder with design-$E_b/N_0 = 3.635$ dB performs even slightly better than the double precision belief propagation decoder with 4 bit channel output quantization. This effect appears for $E_b/N_0 > 4.7$ dB in Figure 5.21a. A similar observation for LDPC decoders with mutual information maximizing lookup tables and a high rate code has also been made independently in [32]. For *regular code d)* investigated in Figure 5.22a, the loss of the best Information Bottleneck decoder

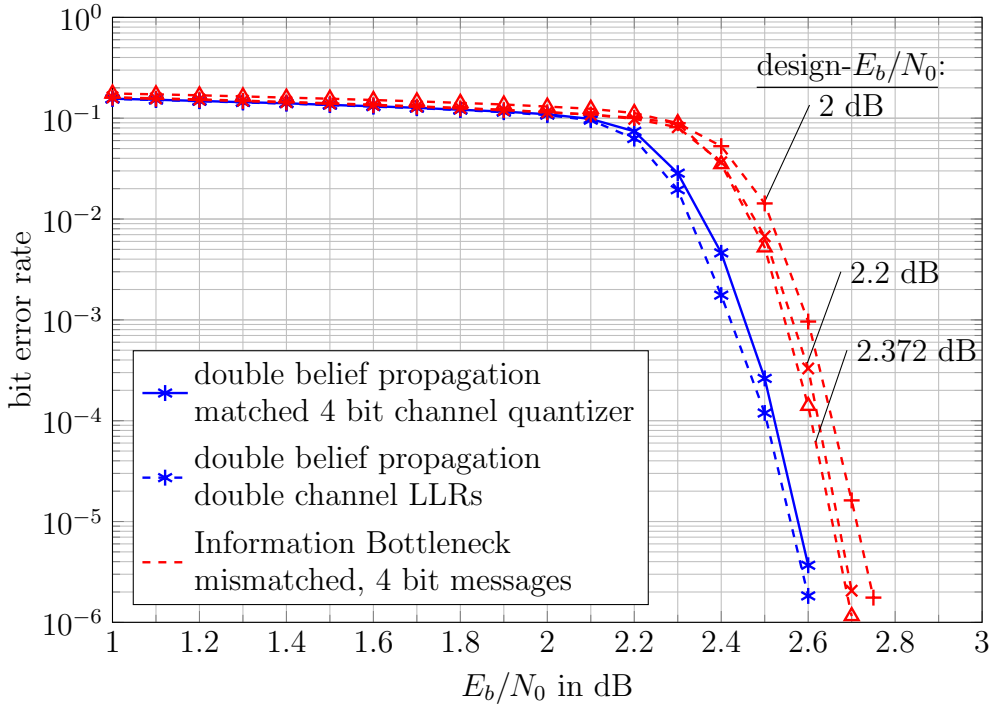


(a) Bit error rate of *regular code c)* ($R = 0.889$) with $i_{\max} = 25$ decoding iterations.

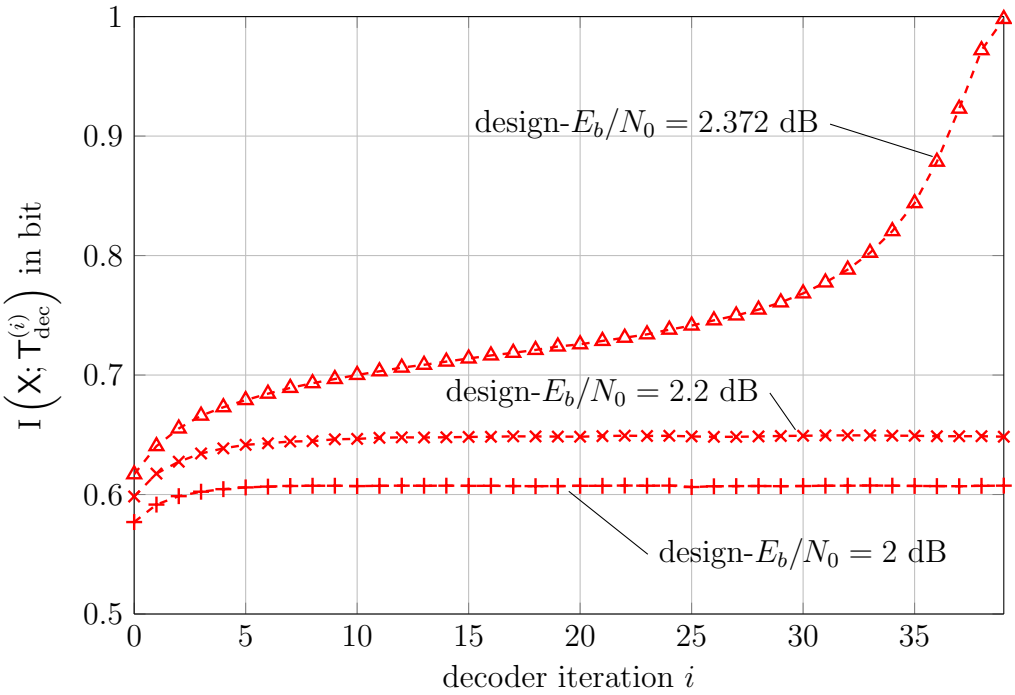


(b) Variable node output information for the $(3, 27)$ regular LDPC ensemble.

Figure 5.21: Analysis of Information Bottleneck LDPC decoders for *regular code c)* with BPSK modulation under AWGN. Information Bottleneck decoders with 4 bit messages and $i_{\max} = 25$ iterations are applied.



(a) Bit error rate of *regular code d)* ($R = 0.375$) with $i_{\max} = 40$ decoding iterations.



(b) Variable node output information for the $(5, 8)$ regular LDPC ensemble.

Figure 5.22: Analysis of Information Bottleneck LDPC decoders for *regular code d)* with BPSK modulation under AWGN. Information Bottleneck decoders with 4 bit messages and $i_{\max} = 40$ iterations are applied.

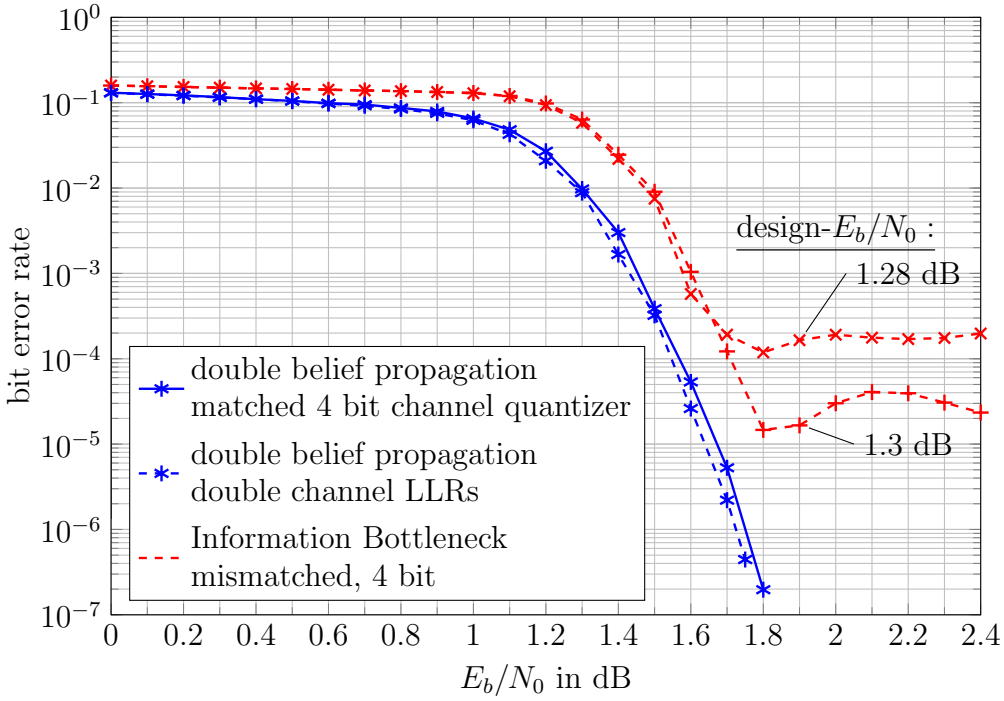
with respect to double precision belief propagation decoding with quantized channel output again is approximately 0.1 dB in the waterfall region of *regular code d)*.

Finally, the effect of choosing the design-E_b/N_0 too high, meaning that $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ approaches 1 bit at $i << i_{\max}-1$ in the design process of the decoder, shall exemplarily be illustrated for *regular codes a)* and *b)*. This is done in Figure 5.23. For the design-E_b/N_0 chosen there, the mutual information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i_{\max}-1)}\right) \approx 1$ for $i << i_{\max}-1$ in the design process of the decoder (see Figure 5.19). The bit error rate results from Figure 5.23 illustrate that this causes a harsh loss over E_b/N_0 and an early error floor. Therefore, it should strictly be avoided.

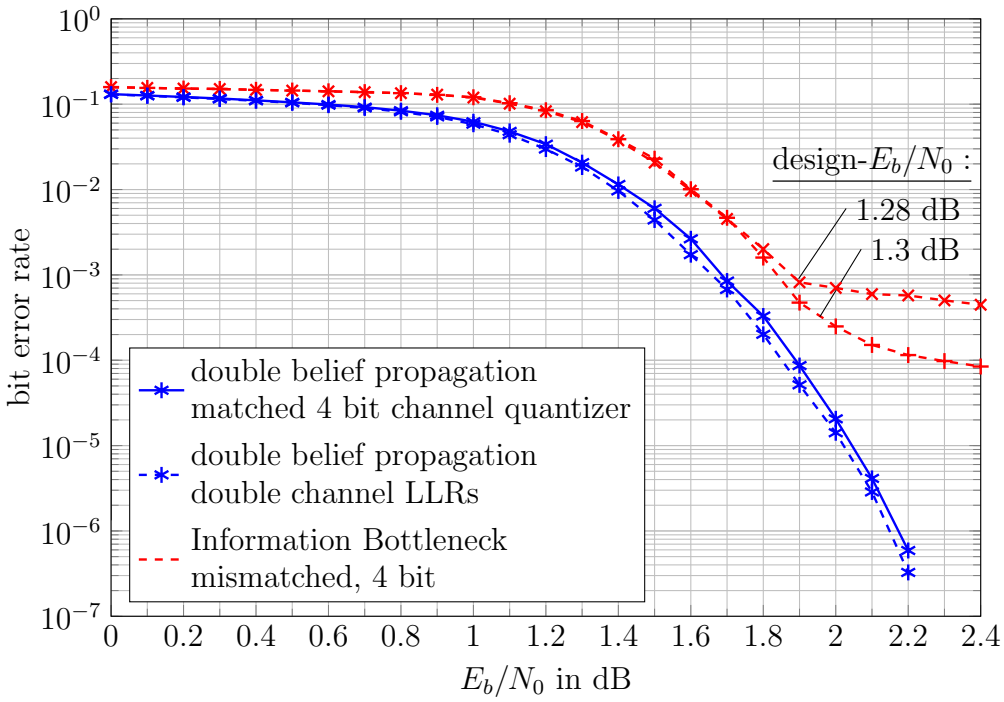
Providing a formal explanation for the observed influence of the design-E_b/N_0 is cumbersome. The Information Bottleneck decoders are used mismatched to their actual design-E_b/N_0 in the decoding process. As a result, the evolution of the variable node output information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ that appears in the decoder design process does not adequately describe the evolution taking place in a decoder which is used mismatched for all values of E_b/N_0. In fact, it does not even exactly match this evolution if the E_b/N_0 on the channel is identical to the design-E_b/N_0. The reason is that discrete density evolution assumes statistical independence of the messages passed along different edges in the Tanner graph of the parity-check matrix which is an idealistic assumption that does not hold for graphs with cycles.

Anyway, a possible intuitive explanation of the observed influence of the design-E_b/N_0 could be that the decoder construction scheme overestimates the reliability of the passed messages in the actual decoder, if the design-E_b/N_0 is choosen too high, while it underestimates this reliability, if the design-E_b/N_0 is choosen too low. The overestimation could be more problematic because if the mutual information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right) \approx 1$ for several iterations, the decoder has no need to represent any soft information on X in the iterative decoding, as all information can can be captured by the hard decision only. However, finding a more detailed explanation for the observed influence of the design-E_b/N_0 is an interesting matter of further research.

From the investigation above, one can draw the conclusion that choosing the design-E_b/N_0 as it has been explained in Section 5.2.4 is reasonable and yields



(a) Bit error rates of *regular code a)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.



(b) Bit error rates for *regular code b)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.

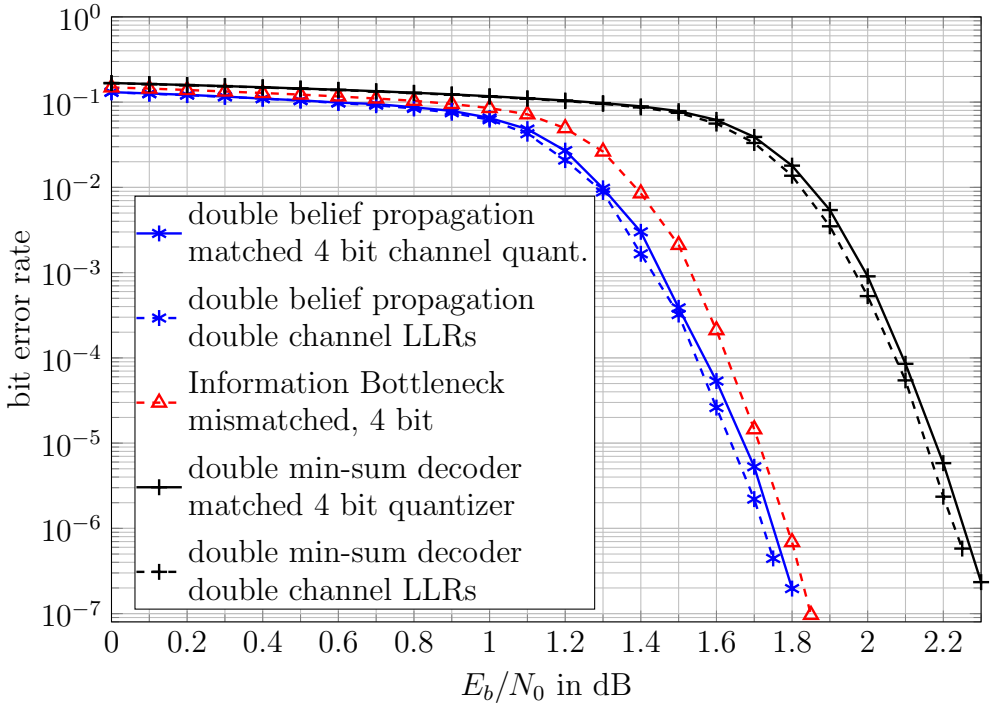
Figure 5.23: Bit error rates of *regular code a)* and *b)* with BPSK modulation under AWGN over E_b/N_0. The effect of choosing the design-E_b/N_0 of the Information Bottleneck decoders too high becomes visible.

4 bit Information Bottleneck decoders with performance enormously close to that of double precision belief propagation decoding for the investigated regular LDPC codes. For *regular codes a)*, *b)* and *d)* the loss in comparison to a double precision belief propagation decoder with a $q = 4$ bit channel output quantizer was only a fraction of a decibel over E_b/N_0 although only $q = 4$ bit messages were used in the Information Bottleneck decoders and all required operations for the decoding were just lookup operations in static tables. For the very short and high rate *regular code c)*, partially even better relative performance could be achieved.

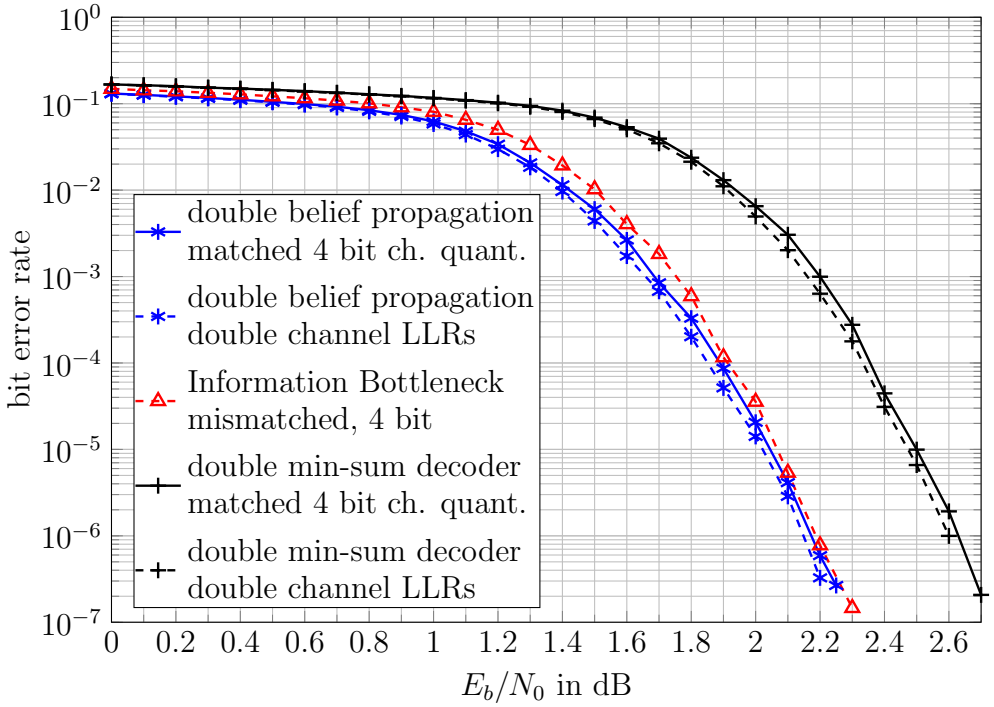
Comparison with Min-Sum Decoding

Above we have seen that $q = 4$ bit Information Bottleneck decoders can be constructed which have performance very close to that of double precision belief propagation decoding. In this section, the best found Information Bottleneck decoders for *regular codes a)*, *b)*, *c)* and *d)* shall be compared to the min-sum decoder. A double precision min-sum decoder which is applied at the output of a quantizer constructed using Algorithm 4 is considered as a fair competitor. Anyway, also a comparison with double precision min-sum decoders without channel output quantization is provided for all considered LDPC codes. Figures 5.24 and 5.25 show the respective bit error rates over E_b/N_0 for the considered codes. The maximum number of decoder iterations $i_{\max}$ was chosen as it has been done in the previous investigation for belief propagation decoding and is mentioned in the caption of each plot.

Figure 5.24 shows that for the two $(3, 6)$ *regular codes a)* and *b)*, the loss of the min-sum decoder with channel output quantization in comparison to the 4 bit Information Bottleneck decoder is approximately 0.4 dB over E_b/N_0 for a bit error rate of 10^{-5}. Figure 5.25a tells that for the very short and high rate *regular code c)*, the gain of the Information Bottleneck decoder is smaller. This gain is approximately 0.1 dB over E_b/N_0 at a bit error rate of 10^{-5} with respect to its fair competitor, i.e., the min-sum decoder with channel output quantization. For the low rate and long *regular code d)* and a bit error rate of 10^{-6}, the gain of the Information Bottleneck decoder even is around 1 dB over E_b/N_0.

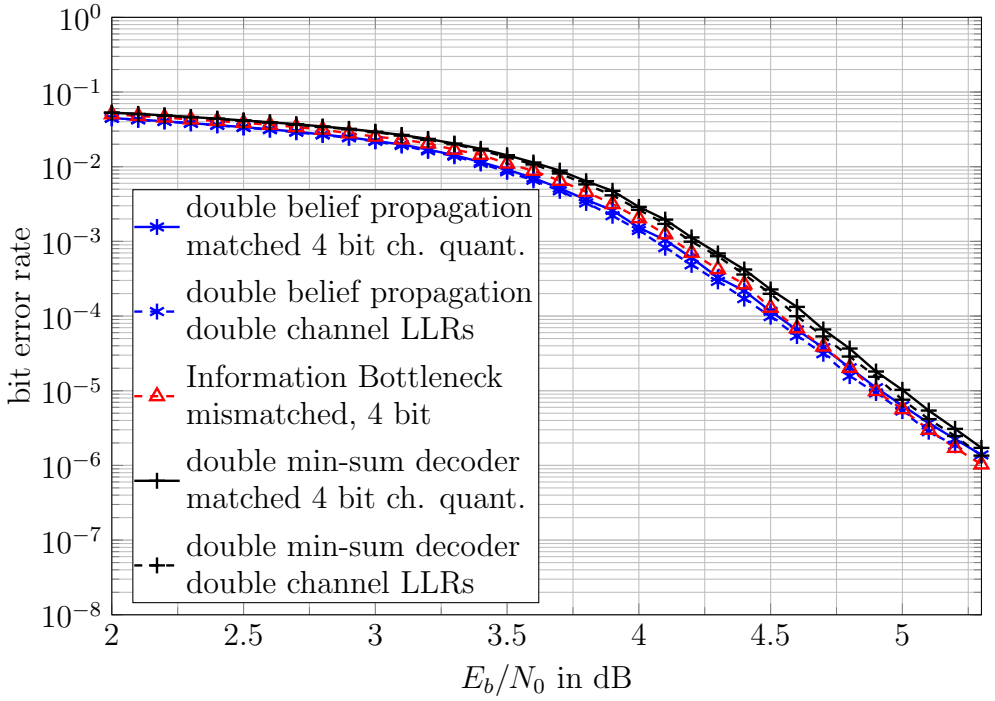


(a) Bit error rates of *regular code a)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.

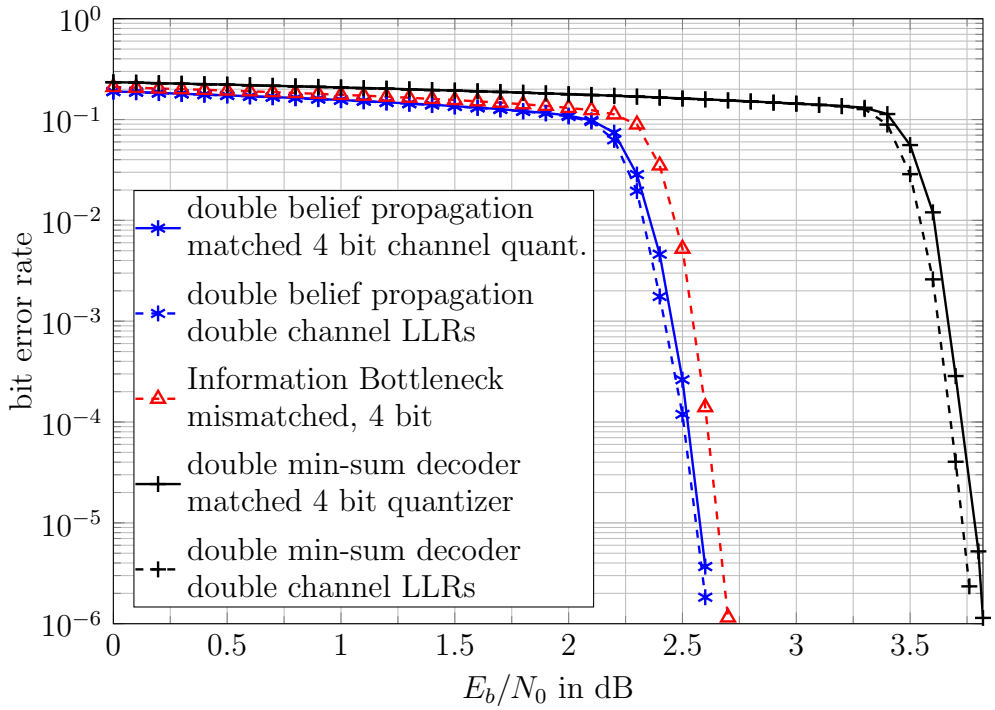


(b) Bit error rates of *regular code b)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.

Figure 5.24: Bit error rates of various decoders for *regular code a)* and *b)* with BPSK modulation under AWGN over E_b/N_0.



(a) Bit error rates of *regular code c)* ($R = 0.889$) with $i_{\max} = 25$ decoding iterations.



(b) Bit error rates of *regular code d)* ($R = 0.375$) with $i_{\max} = 40$ decoding iterations.

Figure 5.25: Bit error rates of various decoders for *regular code c)* and *regular code d)* with BPSK modulation under AWGN over E_b/N_0.

In summary, the Information Bottleneck decoders outperform the min-sum decoder in terms of their error correction capabilities for all investigated codes, despite they use strongly quantized $q = 4$ bit messages in the entire decoding process and all operations applied are simple lookup operations. For completeness, it shall be mentioned again at this point that some additional postprocessing of the messages in the min-sum algorithm could improve its performance at the expense of some additional complexity, as it has been described in Section 2.3.3.

The performance gaps between the distinct decoders vary for the applied *regular codes a)*, *b)*, *c)* and *d)*, depending on their respective code rate an codeword length. A general trend concerning these performance gaps that could be observed from the results presented in this section is that the gains of the Information Bottleneck decoders in comparison with min-sum decoders are greater for strong codes with low rate and large codeword lengths than they are for weaker codes with short codeword length and high rate. For LLR based decoding algorithms, it can be observed above and in the literature [2] that codes with low rates tend to suffer from stronger losses under quantized and suboptimal decoding algorithms than high rate codes. Low rate LDPC codes can, therefore, be understood to be more critical under quantized or suboptimal decoding algorithms than high rate codes.

Influence of the Message Alphabet Cardinalities

Above, we have seen that $q = 4$ bit Information Bottleneck decoders can achieve performance up to 0.1 dB over E_b/N_0 for most of the investigated codes. A question remaining is, if even better Information Bottleneck LDPC decoders can be constructed by further increasing the bit width of the messages. This question shall be answered in this section. Moreover, the effect of decreasing the bit width shall be studied. The *regular code d)* is chosen exemplarily for this investigation as it is the most critical code under suboptimal decoding due to its lowest rate among the investigated codes. It suffers from a loss of approximately 0.1 dB in the waterfall region under decoding with the best found $q = 4$ bit Information Bottleneck decoder in comparison to double precision belief propagation (see Figure 5.22a).

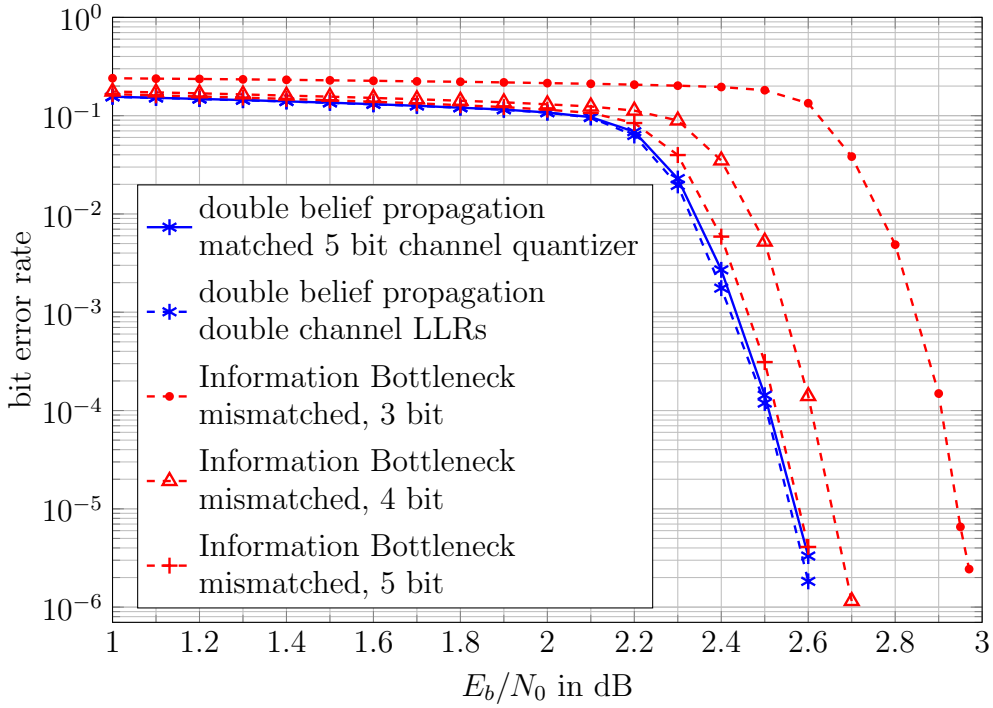


Figure 5.26: Bit error rates of *regular code d)* ($R = 0.375$) under Information Bottleneck decoding with various bit widths, BPSK modulation and AWGN over E_b/N_0.

Figure 5.26 compares the bit error rates of Information Bottleneck LDPC decoders with $q = 3$, $q = 4$ and $q = 5$ bit messages. The design-E_b/N_0 was chosen as it has been explained in the preliminary section for all decoders and the maximum number of performed decoding iterations was $i_{\max} = 40$. Again, double precision belief propagation decoding serves as a reference.

The figure shows that increasing the message bit width to $q = 5$ bits yields an Information Bottleneck decoder which has virtually identical performance as the double precision belief propagation decoder with a $q = 5$ bit channel output quantizer.

In the experiments conducted so far, the message alphabets $\mathcal{Y}_{\text{chan}}, \mathcal{T}_{\text{var}}$ and $\mathcal{T}_{\text{chk}}$ were chosen identical by design. In some implementations, for example, the one to be presented in Chapter 6, this is an intuitive choice because like this, the channel messages and the messages passed during the iterative decoding process need an identical number of bits to be represented in the hardware. Information Bottleneck decoders, however, allow for different sizes of these

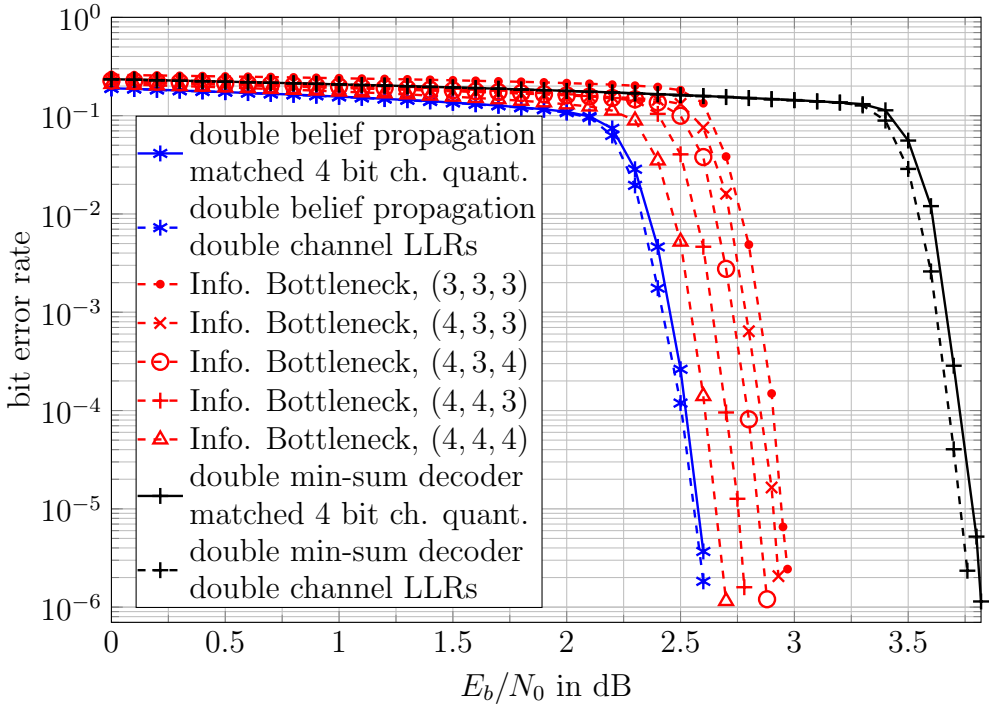


Figure 5.27: Bit error rates of *regular code d)* ($R = 0.375$) for various decoders performing $i_{\max} = 40$ iterations. The applied Information Bottleneck decoders use different bit widths for the channel messages, the variable-to-check node messages and the check-to-variable node messages. All Information Bottleneck decoders are used mismatched to their design-E_b/N_0.

message alphabets and, therefore, different bit widths in general. They can easily be adjusted by choosing the compression cardinalities of the Information Bottleneck algorithms applied to the design of the channel output quantizer, the variable node lookup tables and the check node lookup tables according to the desired bit widths of the respective messages. To illustrate this fact, Figure 5.27 exemplarily investigates the bit error rate performances of Information Bottleneck decoders with various message alphabet cardinalities. The applied code is again *regular code d)* with $i_{\max} = 40$ decoding iterations. The format used to label the curves mentions the bit widths required to store the channel messages, the variable-to-check node messages and the check-to-variable node messages as a triple $(\log_2|\mathcal{Y}_{\text{chan}}|, \log_2|\mathcal{T}_{\text{var}}|, \log_2|\mathcal{T}_{\text{chk}}|)$ in this figure.

The effect of choosing different alphabet sizes for $\mathcal{Y}_{\text{chan}}, \mathcal{T}_{\text{var}}$ and $\mathcal{T}_{\text{chk}}$ can clearly be observed. Different choices result in different performances of the

resulting decoders. A reduction of the bit width of any message from 4 to 3 bits results in a small loss over E_b/N_0. It is particularly interesting that the Information Bottleneck decoder with parameters $(4, 4, 3)$ which uses four bits to represent the variable-to-check node messages and three bits to represent the check-to-variable node messages performs better than the $(4, 3, 4)$ decoder which represents the messages the other way around.

The results from Figure 5.27 illustrate an enormous flexibility in chosing parameters for the needs in a particular implementation by just adapting the compression cardinalities used to design the node operations in Information Bottleneck LDPC decoders. It is possible to trade performance for a smaller bit width required to store the output in all the Information Bottleneck nodes. All Information Bottleneck decoders investigated, even the one with parameters $(3, 3, 3)$, outperform the min-sum decoder.

As it can also be seen in Figure 5.27 for a given maximum bit width, the best performance can be achieved by choosing the cardinalities of $\mathcal{Y}_{\text{chan}}, \mathcal{T}_{\text{var}}$ and $\mathcal{T}_{\text{chk}}$ identical to this maximum bit width, but of course, this choice also requires the largest number of bits in the hardware.

5.2.6 Lookup Table Memory Demands

The memory demand required to store the lookup tables is an important factor. This memory demand depends on the bit widths used for all messages in the decoder, on the structures of the opened nodes used for decomposing the variable and the check node operation and also on the maximum number of decoder iterations $i_{\max}$. For simplicity, it is assumed here that all messages use an identical bit width q.

A decoder with a maximum of $i_{\max}$ decoder iterations, q bit messages and the considered serial decomposition of the node operations as in [28] requires a total lookup table memory amount of

$$\text{size}^{\text{dec}} = q\frac{i_{\max} \cdot 2^{2q} \cdot (d_{\text{v}} + d_{\text{c}} - 2)}{8000} \text{ kilobytes.} \tag{5.31}$$

The memory demands for the considered example codes from above are consolidated in Table 5.4 for $q = 4$ bit messages. So far, these figures show that the size of the lookup tables is in the order of several kilobytes and, therefore, easy to handle in practice. A more detailed comparison of the memory demands

Table 5.4: Memory requirements of Information Bottleneck decoders with $q = 4$ bit messages and serial node decomposition

Code	$(d_\mathrm{v}, d_\mathrm{c})$	i_max	Table size in kilobytes
regular code a)	$(3, 6)$	50	44.8
regular code b)	$(3, 6)$	50	44.8
regular code c)	$(3, 27)$	25	89.6
regular code d)	$(5, 8)$	40	56.32

notation: 1.0 kilobyte = 1,000 bytes

of conventional and the proposed decoders in a DSP implementation will be provided in Chapter 6.

5.2.7 Summary

In this section, LDPC decoders for regular LDPC codes have been presented and investigated which were constructed entirely using the Information Bottleneck method. The applied channel model was AWGN and the considered modulation scheme was BPSK. The key idea for decoder construction with the Information Bottleneck method is application of a discretized density evolution scheme which is used to determine the required input distributions to design lookup tables that serve as node operations in the iterative decoding process. These lookup tables are designed to maximize the preserved relevant information on the bit their output is supposed to represent under a constraint for the output cardinality of the table. The node operations themselve were decomposed to make their implementation practically feasible. It was shown that Information Bottleneck graphs are a perfect tool to visualize the information relations resulting from such a decomposition and that they can be used to describe several possible node decompositions.

The construction process of Information Bottleneck decoders requires the extensive application of Information Bottleneck algorithms and is, therefore, computationally demanding. However, interestingly these computational costs are one time costs. We have seen that Information Bottleneck decoders can be constructed which being used mismatched to their actual design-E_b/N_0 offer performance very close to that of belief propagation decoders. A simple rule for

choosing a design-E_b/N_0 which results in very good decoder performance has been presented and investigated. As a result the entire construction process of Information Bottleneck LDPC decoders can be performed offline and results in a list of static lookup tables which characterize the variable and the check node operations in all decoder iterations i. The size of the resultant lookup tables is manageable in practice.

A comprehensive investigation of the bit error rate performances of the resulting decoders has shown that $q = 4$ bit messages are sufficient to achieve a performance degradation of only 0.1 dB over E_b/N_0 in comparison to a belief propagation decoder which works with double precision arithmetic. For a very short and high rate code, even better performance could be achieved for high E_b/N_0. It was also shown that $q = 4$ bit Information Bottleneck decoders outperform the double precision min-sum decoding algorithm for all investigated LDPC codes.

Finally, it has been described that by adapting the bit widths used to represent the messages in the decoder it is possible to easily trade bit error performance for implementation complexity in Information Bottleneck LDPC decoders. By increasing the bit width of all decoder messages to $q = 5$ bits, a decoder could be constructed which has no mentionable loss with respect to the double precision belief propagation decoding algorithm. It has also been discussed that the message alphabet sizes of the channel messages, the check-to-variable node and the variable-to-check node messages can be chosen differently, also resulting in an enormous amount of flexibility in a decoder implementation.

5.3 Information Bottleneck LDPC Decoders and Quadrature Amplitude Modulation

In many communication systems the spectral efficiency is increased by application of higher order modulation schemes in a coded transmission, for example, QAM. It has been described in Section 5.1.2, how an Information Bottleneck algorithm can be used to design a quantizer which aims to preserve the maximum possible amount of relevant information on the QAM symbols which entered the channel. The preceding section described how to design an Infor-

mation Bottleneck decoder working on the quantization indices delivered by such a quantizer for a BPSK modulated transmission. In this section, the aim is to apply Information Bottleneck LDPC decoders at the receiving end of a QAM modulated transmission scheme with a quantized channel output. The channel model considered again is AWGN.

If an LLR based decoder is used, application of this decoder in this scenario is not very challenging. In fact, one can just calculate LLRs for each codeword bit from the received signal. This process is typically termed *soft demodulation*. The LLRs from the QAM demodulator can then simply be fed to the belief propagation decoding algorithm or any other LLR based algorithm which performs the decoding of the LDPC code. If an Information Bottleneck LDPC decoder shall be used, perhaps surprisingly, several unexpected problems arise which lead to the fact that their integer based decoding process cannot be adapted to the QAM modulation as simple as this is the case for the state-of-the-art decoders. In contrast, first an Information Bottleneck equivalent is required which acts similar to the demodulator in a conventional system.

In the following, it is first explained that in a QAM scheme, several virtual bit channels exist which typically have different qualities. As a result, distinct codeword bits experience different channels, effectively. Afterwards, it is discussed that facing this fact makes a straightforward adaption of the construction process of an Information Bottleneck decoder practically infeasible. A technique called *message alignment* is used to overcome this impairment and the performance of the resulting decoders is investigated. Message alignment has been developed in the scope of this thesis and applied to pairing Information Bottleneck LDPC decoders with QAM in [71]. Later in Section 5.4 it will be shown, that message alignment can also be applied to design Information Bottleneck LDPC decoders for *irregular* LDPC codes.

5.3.1 Pairing Quadrature Amplitude Modulation with Information Bottleneck LDPC Decoders

The major difference between any binary modulation scheme, for example, BPSK and a higher order modulation scheme like $|\mathcal{S}|$-QAM is that in the latter each transmitted modulation symbol $s_k \in \mathcal{S}$ encodes $\log_2 |\mathcal{S}| > 1$ bits.

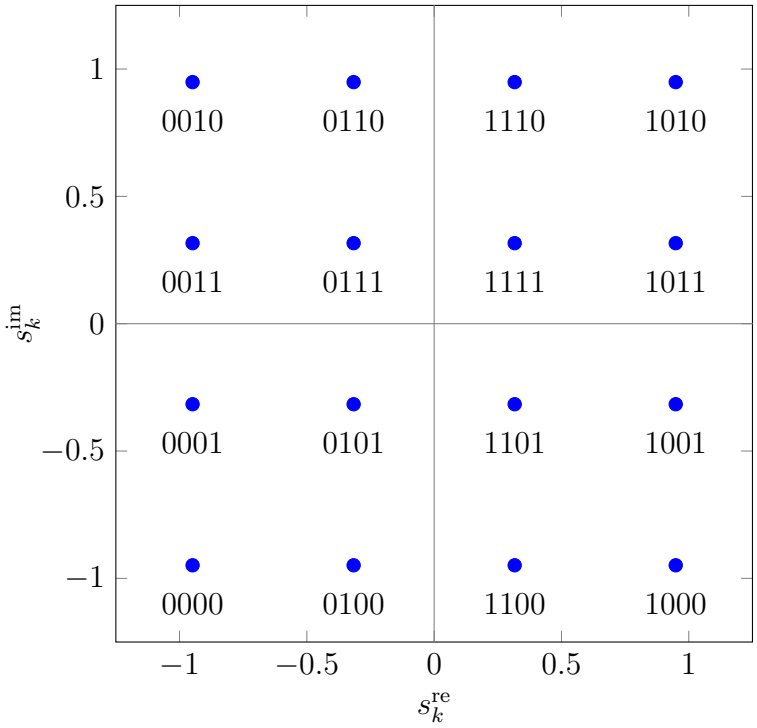


Figure 5.28: Illustration of a 16-QAM modulation alphabet with a Gray labeling.

As it has already been explained in Section 5.1.2, the particular choice of the assignment of bits to modulation symbols is termed *bit labeling*. In Figure 5.28 a Gray labeling is depicted, where the bit labels of neighboring modulation symbols only differ in one bit. This bit labeling was taken from the IEEE 802.11 wireless LAN standard [42].

Let $c_{k,l} \in \{0, 1\}$ denote the bits assigned to the transmitted modulation symbol s_k at time instance k, such that the index $l \in \{0, 1, \ldots, \log_2 |\mathcal{S}| - 1\}$ tells the position of the bit in the bit label of the respective modulation symbol s_k. In the considered LDPC encoded transmission, each bit $c_{k,l}$ is a codeword bit from the codeword of the applied LDPC code. The transmitter has to map the codeword bits onto the respective constellation points. The most practical approach is just sequentially mapping $\log_2 |\mathcal{S}|$ consecutive codeword bits onto a modulation symbol. If other channel codes than LDPC codes, for example, convolutional codes shall be applied it is also very common to include an additional interleaver before the modulation to perform so called

bit interleaved coded modulation. In general, LDPC codes do not require an additional interleaver due to a random structure of the parity-check matrix.

In Section 5.1.2, a quantizer has been built for the received signal of an AWGN channel which aims to preserve the maximum possible amount of relevant information on the modulation symbols which entered the channel. As a result, each pair $(r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ of quantization indices from the quantizer corresponds to one modulation symbol.

The first task we are facing for application of any LDPC decoder is obtaining knowledge on the bits $c_{k,l}$ for all bit positions l from a pair of quantization indices $(r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$. This task is very similar to the process of soft demodulation in a QAM system without a quantizer.

For any QAM modulation pattern with a given bit labeling, it is easy to determine the probabilities of any bit $c_{k,l}$ taking a certain value given s_k^{re} and s_k^{im}, that is, $p(c_{k,l}|s_k^{\mathrm{re}}, s_k^{\mathrm{im}})$. This is simply done by analyzing whether the bit $c_{k,l}$ takes value 0 or 1 for the constellation point $s_k = s_k^{\mathrm{re}} + j s_k^{\mathrm{im}}$.

Using $p(c_{k,l}|s_k^{\mathrm{re}}, s_k^{\mathrm{im}})$ one can determine

$$p(c_{k,l}, r_k^{\mathrm{re}}, r_k^{\mathrm{im}}) = \sum_{s_k^{\mathrm{re}} \in \mathcal{S}^{\mathrm{re}}} \sum_{s_k^{\mathrm{im}} \in \mathcal{S}^{\mathrm{im}}} p(c_{k,l}|s_k^{\mathrm{re}}, s_k^{\mathrm{im}}) p(s_k^{\mathrm{re}}, r_k^{\mathrm{re}}) p(s_k^{\mathrm{im}}, r_k^{\mathrm{im}}). \tag{5.32}$$

Please note that $p(s_k^{\mathrm{re}}, r_k^{\mathrm{re}})$ and $p(s_k^{\mathrm{im}}, r_k^{\mathrm{im}})$ in this equation are known from the Information Bottleneck design algorithm of the channel output quantizer. The joint distribution $p(c_{k,l}, r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ also enables to calculate the posterior distribution $p(c_{k,l}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$. Using this distribution, one can further determine LLRs as

$$L(c_{k,l}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}}) = \log \frac{\Pr(\mathsf{C}_{k,l} = 0|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})}{\Pr(\mathsf{C}_{k,l} = 1|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})}. \tag{5.33}$$

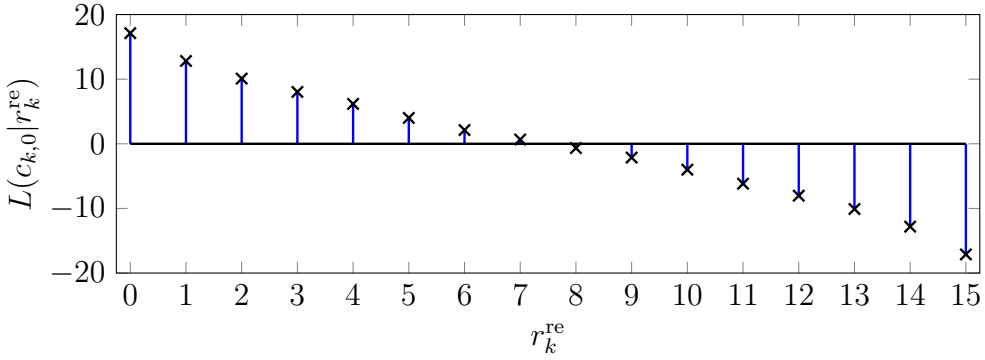
Moreover, it enables to calculate the distributions

$$p(c_{k,l}, r_k^{\mathrm{re}}) = \sum_{r_k^{\mathrm{im}} \in \mathcal{R}^{\mathrm{im}}} p(c_{k,l}, r_k^{\mathrm{re}}, r_k^{\mathrm{im}}) \tag{5.34}$$

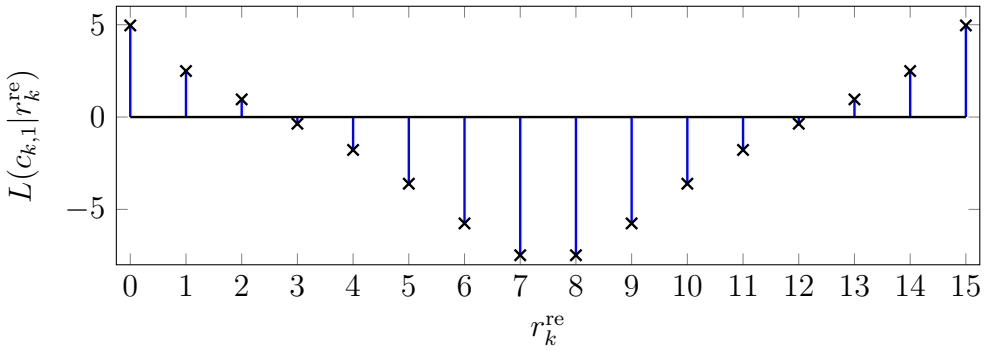
$$p(c_{k,l}, r_k^{\mathrm{im}}) = \sum_{r_k^{\mathrm{re}} \in \mathcal{R}^{\mathrm{re}}} p(c_{k,l}, r_k^{\mathrm{re}}, r_k^{\mathrm{im}}) \tag{5.35}$$

which in turn also allow for calculation of the posteriors $p(c_{k,l}|r_k^{\mathrm{re}})$ and $p(c_{k,l}|r_k^{\mathrm{im}})$.

Some bit labelings, for example, the 16-QAM bit labeling from Figure 5.28, have the property that half of the bits $c_{k,l}$ depend only on s_k^{re} and the other half only depends on s_k^{im}. In Figure 5.28 it is easy to see that the first two bits



(a) LLRs for the first bit $c_{k,0}$ in the bit label.



(b) LLRs for the second bit $c_{k,1}$ in the bit label.

Figure 5.29: Distinct meanings of a single received r_k^{re} for two bits $c_{k,0}$ and $c_{k,1}$ in the bit label of a 16-QAM modulation symbol for AWGN with noise variance $\sigma_{\tilde{n}}^2 = 0.1$.

$c_{k,0}, c_{k,1}$ are determined by s_k^{re} and the last two bits $c_{k,2}, c_{k,3}$ are determined by s_k^{im}. As a result also the LLRs $L(c_{k,l}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ either only depend on r_k^{re} or r_k^{im}. Moreover, the bit labeling from Figure 5.28 is symmetrical between the real and the imaginary signal components for the two bits assigned to s_k^{re} and s_k^{im}, respectively. To visualize that in this scenario, a single r_k^{re} implies beliefs $p(c_{k,l}|r_k^{\mathrm{re}})$ for two bit positions $l \in \{0, 1\}$, Figure 5.29 shows the LLRs $L(c_{k,0}|r_k^{\mathrm{re}})$ corresponding to the particular quantization index r_k^{re} in the upper part and $L(c_{k,1}|r_k^{\mathrm{re}})$ in the lower part of the figure for an exemplary noise variance of $\sigma_{\tilde{n}}^2 = 0.1$. The quadrature component $\tilde{r}_k^{\mathrm{re}}$ was quantized using 4 bit quantization indices, that is, $r_k^{\mathrm{re}} \in \{0, 1, \ldots, 15\}$ in this example.

Obviously, the meaning of the event $\mathsf{R}_k^{\mathrm{re}} = r_k^{\mathrm{re}}$ for the two bits $c_{k,0}$ and $c_{k,1}$ in the bit label differs significantly. It is eyecatching that the maximum LLR

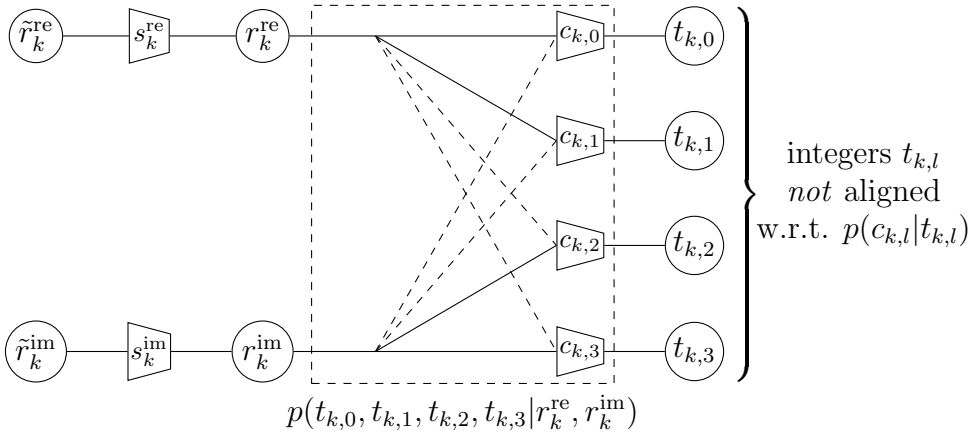


Figure 5.30: Information Bottleneck graph of an Information Bottleneck demodulator for 16-QAM.

$L(c_{k,0}|r_k^{\mathrm{re}})$ is much higher than the maximum LLR $L(c_{k,1}|r_k^{\mathrm{re}})$, indicating that more reliable bit decisions are possible for bit $c_{k,0}$ than for bit $c_{k,1}$. As a result, one can conclude that for the bit labeling from Figure 5.28, effectively two distinct channels with different channel qualities exist under the considered QAM modulation. This effect depends on the used bit labeling and is typical for $|\mathcal{S}|$-QAM.

The question is now, what needs to be done with $(r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ to apply an Information Bottleneck LDPC decoder. A simple idea is trying to extract relevant information on each bit $c_{k,l}$ from $(r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ with the Information Bottleneck method. This approach is visualized in an Information Bottleneck graph in Figure 5.30 for 16-QAM exemplarily. The shown Information Bottleneck graph steps into the position of a demodulator in a conventional system because it extracts relevant information on each bit $c_{k,l}$. The demodulator node $p(t_{k,0}, t_{k,1}, t_{k,2}, t_{k,3}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ provides integers $t_{k,l}$ for a received pair $(r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ for $l \in \{0, 1, \ldots, \log_2 |\mathcal{S}| - 1\}$ while aiming to maximize $\mathrm{I}(\mathsf{C}_{k,l}; \mathsf{T}_{k,l})$. The joint distributions $p(c_{k,l}, r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$, $l \in \{0, 1, \ldots, \log_2 |\mathcal{S}| - 1\}$ required to design the nodes $p(t_{k,l}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ are given by Equation (5.32).

The Information Bottleneck demodulator can even be paired with a conventional decoding algorithm, by calculating LLRs $L(c_{k,l}|t_{k,l})$ from the integer indices $t_{k,l}$ because the Information Bottleneck algorithms applied to design the nodes $p(t_{k,l}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ deliver $p(c_{k,l}|t_{k,l})$ for all l as side products. Using the LLRs $L(c_{k,l}|t_{k,l})$ which can be obtained for any pair $(r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ it is straightfor-

ward to apply a conventional LDPC decoding algorithm to the decoding. This allows for a fair comparison between any LLR based decoding algorithm and an Information Bottleneck decoder. However, it requires to match all lookup tables in the demodulator to the E_b/N_0 on the channel to obtain meaningful LLRs.

Inside the opened node $p(t_{k,0}, t_{k,1}, t_{k,2}, t_{k,3}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$, some connections to the inputs of the Information Bottleneck nodes $p(t_{k,l}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ are drawn dashed to indicate that, depending on the bit labeling, these input connections could be dismissed, if a certain bit $c_{k,l}$ only depends on one of the quadrature components, as it has been explained for the 16-QAM Gray labeling from Figure 5.28. In this case, the joint distributions to design the respective nodes are given by $p(c_{k,l}, r_k^{\mathrm{re}})$ and $p(c_{k,l}, r_k^{\mathrm{im}})$ from Equations (5.34) and (5.35). This thesis will focus on this case, but the presented ideas are easily extendable for arbitrary bit labelings using the provided equations.

A simple, but very interesting special case is the case, where the dashed connections in Figure 5.30 can be dismissed due to the applied Gray labeling from Figure 5.28 and moreover, the cardinalities $|\mathcal{T}_{k,l}| = |\mathcal{R}^{\mathrm{re}}| = |\mathcal{R}^{\mathrm{im}}| \, \forall l$. This was investigated in the scope of this thesis in [71]. From the perspective of preserving relevant information on each bit $c_{k,l}$, in this case it would clearly be an optimal choice to implement the Information Bottleneck nodes $p(t_{k,l}|r_k^{\mathrm{re}})$, $l \in \{0, 1\}$ and $p(t_{k,l}|r_k^{\mathrm{im}})$, $l \in \{2, 3\}$ as identity mappings because an identity mapping cannot loose any information. In this case, the quantization indices from the quantizer would directly be passed to the check nodes in the first step of decoding, just as it was the case in the BPSK system, such that $t_{k,0} = t_{k,1} = r_k^{\mathrm{re}}$ and $t_{k,2} = t_{k,3} = r_k^{\mathrm{im}}$. This procedure, however, is problematic for QAM.

Recall that in the design process of the first applied check node operation in an Information Bottleneck decoder, joint distributions $p\left(b_n, y_n^{(i)}\right)$ which connect the bits b_n in the neighborhood of a check node and the messages $y_n^{(i)}$ passed over the edges of the Tanner graph are used to design the node operation (see Figure 5.13). If QAM modulation is applied, any bit b_n corresponds to some bit $c_{k,l}$ in the bit label of a QAM symbol s_k. As a result, the joint distributions $p\left(b_n, y_n^{(0)}\right)$ would correspond to some joint distributions $p(c_{k,l}, r_k^{\mathrm{re}})$ and $p(c_{k,l}, r_k^{\mathrm{im}})$, if the quantization indices r_k^{re} and r_k^{im} were passed

to the check nodes directly in the initial step of decoding. The problem is now that the joint distributions $p(c_{k,l}, r_k^{\mathrm{re}})$ and $p(c_{k,l}, r_k^{\mathrm{im}})$ differ significantly for different l, as it can be concluded from the comparison of the LLRs in Figure 5.29. Designing the check node operations for the first decoding iteration, however, requires knowledge of the particular combinations of $p\left(b_n, y_n^{(i)}\right)$ for each check node. As a result, it would matter for the design of *each* check node operation, which particular combination (n, k, l) is implied by the parity-check matrix of the code. In this case, other than for BPSK, each check node operation would have to be designed differently. Moreover, the joint output distributions $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(i)}\right)$ would depend on the edge configuration of each check node. Tracking all different involved joint distributions for all nodes and all iterations in the iterative decoder construction scheme from Section 5.2.3 would have prohibitive complexity.

This problem which we have described in [71] for the special case of passing r_k^{re} and r_k^{im} directly to the check nodes in the initial decoding step, similarly exists in the case illustrated in Figure 5.30, where the integers $t_{k,l}$ are passed into the decoding graph for all l. The joint distributions $p(c_{k,l}, t_{k,l})$ differ for different values of the position l in the bit label of a QAM symbol, but a receiving check node cannot resolve l. We will stick with the more general description of passing $t_{k,l}$ into the decoding graph in the remainder. The reason that the distributions $p(c_{k,l}, t_{k,l})$ differ is that the nodes $p(t_{k,l}|r_k^{\mathrm{re}}, r_k^{\mathrm{im}})$ are designed independently.

A proper way to formalize the fact that the joint distributions of the codeword bits b_n in the neighborhood of a check node and the messages $y_n^{(0)}$ passed to this node correspond to some $p(c_{k,l}, t_{k,l})$ is considering the conditional distribution $p\left(b_n, y_n^{(0)}|l\right)$ which takes the probabilities from $p\left(c_{k,l}, t_{k,l}\right)$ into account, that is,

$$p\left(b_n, y_n^{(0)}|l\right) = \Pr\left(\mathsf{C}_{k,l} = b_n, \mathsf{T}_{k,l} = y_n^{(0)}\right). \tag{5.36}$$

The condition on l expresses that an incoming message $y_n^{(0)}$ corresponds to the bit label position l. To eliminate the dependency on l and to obtain a single distribution $p\left(b_n, y_n^{(0)}\right)$ that can be used for the design of the check node operation in Equation (5.28), one could think of evaluating

$$p\left(b_n, y_n^{(0)}\right) = \sum_{l=0}^{\log_2|\mathcal{S}|-1} p\left(b_n, y_n^{(0)}|l\right) p(l). \tag{5.37}$$

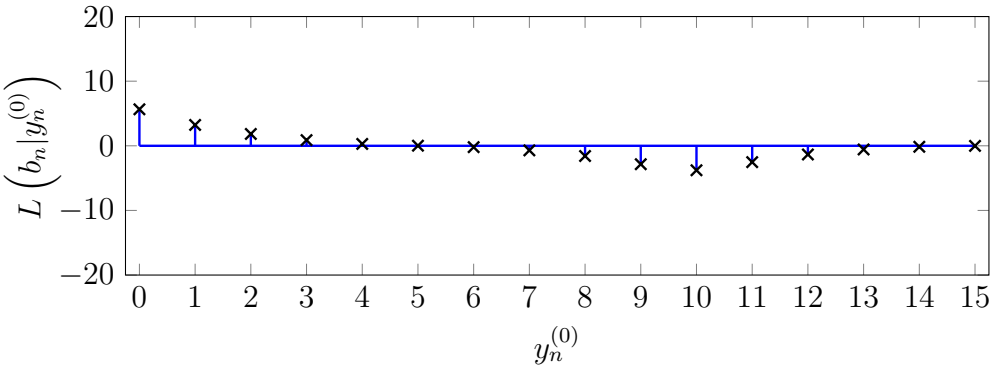


Figure 5.31: LLRs corresponding to message $y_n^{(0)}$ for a codeword bit b_n under 16-QAM with AWGN noise variance $\sigma_{\tilde{n}}^2 = 0.1$ without message alignment. The incoming messages $y_n^{(0)}$ of the check nodes directly correspond to the integers $t_{k,l}$ from the Information Bottleneck demodulator.

This step averages over all $p\left(b_n, y_n^{(0)}|l\right)$ for all values of l and causes a catastrophic result. To visualize this, reconsider the simplest case again, where $t_{k,0} = t_{k,1} = r_k^{\mathrm{re}}$ and $t_{k,2} = t_{k,3} = r_k^{\mathrm{im}}$.

Figure 5.31 shows the LLRs $L\left(b_n|y_n^{(0)}\right)$ which result from the joint distribution $p\left(b_n, y_n^{(0)}\right)$ in Equation (5.37) for the considered 16-QAM bit labeling from Figure 5.28 and a noise variance $\sigma_{\tilde{n}}^2 = 0.1$ in this case. As it is revealed by a comparison with Figure 5.29, the absolute magnitudes of the LLRs are strongly weakened. Several LLRs are pushed approximately to zero by the averaging in Equation (5.37) and the maximum reliability is severely degraded. The reason is that the distributions $p\left(b_n|y_n^{(0)}, l\right)$ differ significantly for different l, but fixed $y_n^{(0)}$, as it can be concluded from Figures 5.29 and 5.31.

As a result, directly passing the $t_{k,l}$ from Figure 5.30 to the check nodes in the initial decoding iteration without a further processing and trying to design an Information Bottleneck decoder for this scenario results in a total decoding failure. Obviously, the problem is that identical $t_{k,l}$ for different l in Figure 5.30 do not refer to the same $p(c_{k,l}|t_{k,l})$ or, equivalently, to the same LLR $L(c_{k,l}|t_{k,l})$. One can consider the distributions $p(c_{k,l}|t_{k,l})$ to not be *aligned* with respect to their meaning for a codeword bit for different values of l. The term *alignment* in this case describes that identical messages which arrive at the check nodes should refer to an identical, or at least a similar belief on

the corresponding codeword bit, independent of their bit label index l. In the following, a technique is presented to overcome this impairment.

5.3.2 Message Alignment

Instead of passing the integers $t_{k,l}$ to the check nodes in the initial decoding step, the idea is to design deterministic mappings $p(z_{k,l}|t_{k,l})$ which map each $t_{k,l} \in \mathcal{T}_l$ onto a $z_{k,l} \in \mathcal{Z}_l$. These integers $z_{k,l}$ are then passed into the decoding graph and form the incoming messages $y_n^{(0)}$ of the check nodes in Figure 5.13.

For the sake of a simple notation, the indices are dropped in the following discussion because they are not important to the problem. Therefore, let b denote *any* codeword bit in the neighborhood of any check node and let y be the corresponding incoming message to this node, which is given by some $z_{k,l}$ in the considered setup. The key here is that b is any codeword bit which corresponds to any $c_{k,l}$ in the bit label of a modulation symbol and the corresponding bit label index l is unknown to the receiving node. The design idea for the deterministic mappings $p(z_{k,l}|t_{k,l})$ is choosing them such that the mutual information $\mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y})$ is minimized in this setup. The argumentation for this design idea is as follows.

Due to the chain rule of mutual information, $\mathrm{I}(\mathsf{B};\mathsf{Y},\mathsf{L})$ is given by

$$\mathrm{I}(\mathsf{B};\mathsf{Y},\mathsf{L}) = \mathrm{I}(\mathsf{B};\mathsf{Y}) + \mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y}). \tag{5.38}$$

As the mutual information $\mathrm{I}(\mathsf{B};\mathsf{Y},\mathsf{L})$ tells what is known about b from knowing y and l, an interpretation of this equation reveals that $\mathrm{I}(\mathsf{B};\mathsf{Y})$ is the information on b that results from receiving y alone. The conditional mutual information $\mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y})$ can be interpreted as the *additional* amount of information on bit b which results from also getting to know the bit label position l, if the incoming message y is already known. In our setup, we want that knowing l in addition to the incoming integer y does not provide additional information. The conditional mutual information $\mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y})$ can be rewritten as

$$\mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y}) = \mathbb{E}_{\mathsf{Y},\mathsf{L}}\left\{D_{\mathrm{KL}}\left\{p(b|y,l) \mid p(b|y)\right\}\right\}. \tag{5.39}$$

This equation is derived in Appendix A.6.

The approach chosen here is now to try to minimize the expectation of the Kullback-Leibler divergence $D_{\mathrm{KL}}\left\{p(b|y,l) \mid p(b|y)\right\}$ which appears in Equation (5.39). At the same time, we aim to maximize the relevant information $\mathrm{I}(\mathsf{B};\mathsf{Y})$.

The best possible case in terms of the minimization of the expectation $\mathbb{E}_{\mathsf{Y},\mathsf{L}}\left\{D_{\mathrm{KL}}\left\{p\left(b|y,l\right)\,|\,p\left(b|y\right)\right\}\right\}$, is the one where $D_{\mathrm{KL}}\left\{p\left(b|y,l\right)\,|\,p\left(b|y\right)\right\}=0$, $\forall(y,l)$ and as a consequence $\mathrm{I}\left(\mathsf{B};\mathsf{L}|\mathsf{Y}\right)=0$. This is the case, if $p(b|y,l)=p(b|y)$, that is, if B is statistically independent of L given Y.

Vividly, this means that the LLRs $L(b|y,l)$ for pairs (y,l) must be identical for any l to achieve this independency which is definitely not the case, if r_k^{re} and r_k^{im} form the incoming messages of the check nodes in the initial decoding iteration, as it has been discussed above and illustrated in Figure 5.29.

A simple, yet effective strategy to construct mappings $p(z_{k,l}|t_{k,l})$ with the desired characteristics is presented in the following subsection.

5.3.3 A Simple Message Alignment Algorithm

Minimizing the expectation from Equation (5.39) over the mappings $p(z_{k,l}|t_{k,l})$ is cumbersome, because $p(y,l)$ inherently depends on all $p(z_{k,l}|t_{k,l})$. Although no closed form solution to obtain the mappings $p(z_{k,l}|t_{k,l})$ could be found in the scope of this thesis, a very simple strategy to obtain mappings enabling to pair Information Bottleneck LDPC decoders with QAM is presented in the following which results in very good bit error performance.

At the beginning of the design process of the mappings $p(z_{k,l}|t_{k,l})$ one has access to the distributions $p(c_{k,l},t_{k,l})$ from the design process of the Information Bottleneck nodes $p(t_{k,l}|r_k^{\mathrm{re}},r_k^{\mathrm{im}})$ inside the Information Bottleneck demodulator from Figure 5.30. Please note that in the case where the $t_{k,l}$ are given by r_k^{re} or r_k^{im}, these distributions equal $p(c_{k,l},r_k^{\mathrm{re}})$ or $p(c_{k,l},r_k^{\mathrm{im}})$ from Equations (5.34) and (5.35). From these distributions one first finds the bit label position l which results in the highest mutual information $\mathrm{I}\left(\mathsf{C}_{k,l};\mathsf{T}_{k,l}\right)$. This identifies the best effective channel with index l' of all bit label positions l.

The mappings $p(z_{k,l}|t_{k,l})$ are designed as follows. In the initial step, the mapping $p\left(z_{k,l'}|t_{k,l'}\right)$ for the most realiable effective channel with the highest mutual information $\mathrm{I}\left(\mathsf{C}_{k,l'};\mathsf{T}_{\mathsf{k},\mathsf{l}'}\right)$ is initialized as an identity mapping. This is mostly motivated by the fact that we aim for preservation of the output information of this node.

The other mappings $p\left(z_{k,l}|t_{k,l}\right)$ are then designed according to the rule

$$z_{k,l}=\arg\min_{z_{k,l'}\in\mathcal{Z}_{l'}} D_{\mathrm{KL}}\left\{p\left(b|t_{k,l}\right)\,|\,p\left(b|z_{k,l'}\right)\right\}\;\forall t_{k,l}\in\mathcal{T}_l. \tag{5.40}$$

Here, $p(b|t_{k,l})$ corresponds to $p\left(c_{k,l}|t_{k,l}\right)$ with relabeled $c_{k,l} = b$.

Using this strategy, each $t_{k,l} \in \mathcal{T}_l$ is mapped onto one $z_{k,l} \in \mathcal{Z}_{l'}$ which fits best to the intended belief $p(b|t_{k,l})$ in the sense of minimizing the Kullback-Leibler divergence from Equation (5.40) in the initial step.

Once the initial mappings $p\left(z_{k,l}|t_{k,l}\right)$ have been constructed for each l, one uses them to determine

$$p\left(b, z_{k,l}\right) = \sum_{t_{k,l} \in \mathcal{T}_l} p\left(z_{k,l}|t_{k,l}\right) p(b, t_{k,l}) \tag{5.41}$$

for all l. From the perspective of the check nodes receiving incoming messages $y = z_{k,l}$, the distribution $p\left(b, z_{k,l}\right)$ corresponds to $p(b, y|l)$ for $y = z_{k,l}$ and, hence, can be used to determine

$$p(b, y) = \sum_{l=0}^{\log_2 |\mathcal{S}|-1} p(b, y|l)p(l) = \frac{1}{\log_2 |\mathcal{S}|} \sum_{l=0}^{\log_2 |\mathcal{S}|-1} p(b, y|l) \tag{5.42}$$

$$p(b|y) = \frac{p(b, y)}{p(y)}. \tag{5.43}$$

Note now that $p(b|y)$ describes a belief on any bit b from having received a particular $y = z_{k,l}$ for an unknown bit label index l.

At this point, one starts over from Equation (5.40), but using $p(b|y)$ as the second argument of the Kullback-Leibler divergence to re-determine all mappings $p(z_{k,l}|t_{k,l})$. This procedure is repeated until the resulting mappings do not change any more. This algorithm converges very fast, typically within one or two iterations. The resultant mappings $p(z_{k,l}|t_{k,l})$ are stored and termed the *message alignment mappings* in the following.

The joint distribution $p(b, y)$ from Equation (5.42) is the distribution of *any* codeword bit b and *any* incoming message y of a check node in the initial decoding iteration $i = 0$. As a result, it steps into the place of $p\left(b_n, y_n^{(0)}\right)$ in Equation (5.28) and an Information Bottleneck LDPC decoder for a regular LDPC code can be constructed for any QAM scheme, exactly as it has been done for BPSK in Section 5.2. The only additional steps that have to be included before decoding are lookups of $z_{k,l}$ for each $t_{k,l}$, such that the integers $z_{k,l}$ can be passed to the check nodes in the initial decoding iteration.

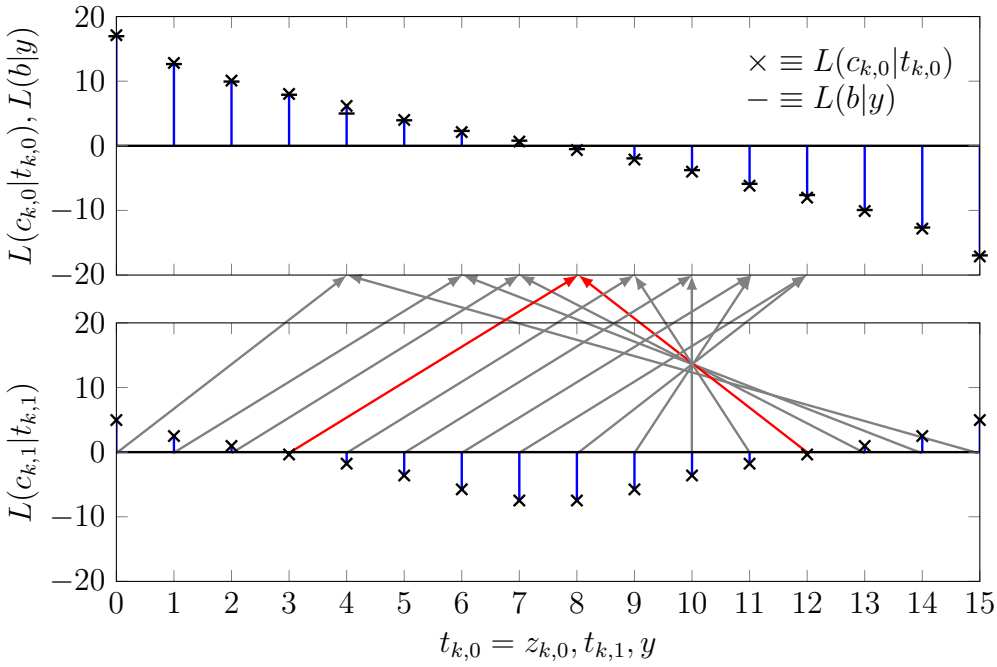


Figure 5.32: Illustration of an exemplary message alignment mapping $p(z_{k,1}|t_{k,1})$. Red arrows illustrate that for $t_{k,1} = 3$ and $t_{k,1} = 12$, the integer $z_{k,1} = 8$ is passed to the check nodes for bit $c_{k,1}$.

5.3.4 Results and Discussion

In this section, first a graphical interpretation of message alignment shall be provided allowing to vividly illustrate the consequences of the application of the message alignment mappings. Afterwards, the bit error rate performance of pairing QAM with Information Bottleneck decoders for regular codes under AWGN is investigated.

Illustration of Message Alignment

Recall the case, where the 16-QAM bit labeling from Figure 5.28 is used and $t_{k,0} = t_{k,1} = r_k^{\mathrm{re}}$ are the outputs of the demodulator referring to the bits $c_{k,0}$ and $c_{k,1}$ which has already been considered above. For an exemplary channel noise variance $\sigma_{\tilde{n}}^2 = 0.1$, in this case the message alignment technique presented in Section 5.3.3 delivered an identity mapping $p(z_{k,0}|t_{k,0})$, such that $z_{k,0} = r_k^{\mathrm{re}}$. It shall be emphasized, however, that this is a special case chosen for exemplary illustration. In general, the iterative message alignment algorithm from

Section 5.3.3 could also deliver a $p(z_{k,0}|t_{k,0})$ which is not an identity mapping for other bit labelings or other values of $\sigma_{\tilde{n}}^2$. The mapping $p(z_{k,1}|t_{k,1})$ in this example is more interesting and visualized using arrows in Figure 5.32. The arrows indicate which $t_{k,1}$ are mapped onto which $z_{k,1}$. For example, the arrows highlighted in red indicate that for $t_{k,1} = 3$ and $t_{k,1} = 12$, the integer $z_{k,1} = 8$ is passed to the check nodes for decoding. As it can be seen by following the arrows from the bottom to the top plot, the message alignment mapping maps those $t_{k,1}$ onto $z_{k,1}$ which imply similar LLRs $L(c_{k,1}|z_{k,1})$ as $L(c_{k,0}|z_{k,0})$ for $z_{k,0} = z_{k,1}$. Hence, whenever an integer message $z_{k,l}$ is passed into the decoding graph, it corresponds to approximately the same LLR $L(c_{k,l}|z_{k,l})$, irrespective of the bit label position l, but only depending on the particular value of $y = z_{k,l}$. To visualize this fact, the top plot from Figure 5.32 also shows the LLRs $L(b|y)$ calculated from $p(b|y)$ from Equation (5.43) as horizontal line markers. These markers almost superimpose with the corresponding $\times$ markers for $L(c_{k,0}|t_{k,0})$ showing that the originally intended beliefs for all bits $c_{k,l}$ can now propagate through the graph accurately by passing the integers $z_{k,l}$ to the check nodes.

Knowing the distributions $p(b, y)$ from Equation (5.42) and $p(b, y, l) = p(b, y|l)p(l)$ for the investigated scenario also allows to calculate the mutual information values of $\mathrm{I}(\mathsf{B};\mathsf{Y})$ and $\mathrm{I}(\mathsf{B};\mathsf{Y},\mathsf{L})$. If the message alignment technique works properly, it should guarantee that with message alignment in place, $\mathrm{I}(\mathsf{B};\mathsf{Y},\mathsf{L}) \approx \mathrm{I}(\mathsf{B};\mathsf{Y})$, such that $\mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y}) \approx 0$. Figure 5.33 shows the mutual information $\mathrm{I}(\mathsf{B};\mathsf{Y},\mathsf{L})$ and also $\mathrm{I}(\mathsf{B};\mathsf{Y})$ under application of the obtained message alignment mappings and the considered 16-QAM over the channel noise variance $\sigma_{\tilde{n}}^2$. As it can be seen, both mutual information values almost match, such that $\mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y}) \approx 0$ as it was intended. For completeness, also a curve for $\mathrm{I}(\mathsf{B};\mathsf{Y})$ without message alignment is included, which corresponds to just using identity mappings $p(z_{k,l}|t_{k,l})$ for all l. As it can be seen, the message alignment has a drastical impact on the mutual information shared between the incoming messages y and the bits b they represent, hence showing that message alignment is a required key technique for pairing QAM with Information Bottleneck LDPC decoding. It shall be emphasized, that the dashed red curve for $\mathrm{I}(\mathsf{B};\mathsf{Y})$ with message alignment and the black curve for $\mathrm{I}(\mathsf{B};\mathsf{Y},\mathsf{L})$ do not exactly superimpose, but the red one lies slightly below the black one.

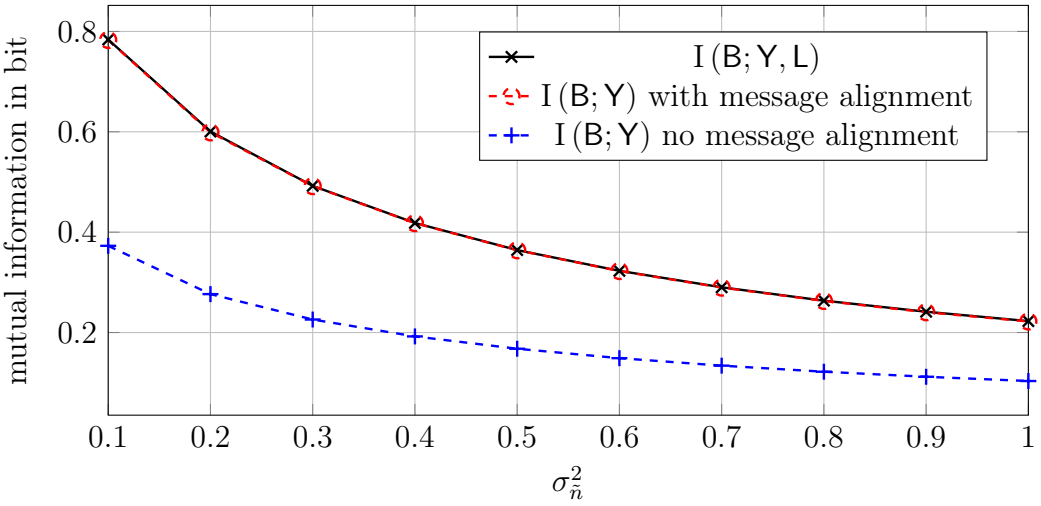


Figure 5.33: Comparison of the mutual information I (B; Y, L) and I (B; Y) with and without message alignment for Gray encoded 16-QAM.

Just as the lookup tables used inside an Information Bottleneck LDPC decoder, the message alignment mappings $p(z_{k,l}|t_{k,l})$ depend on the E_b/N_0. However, they have to be used mismatched to avoid the need of generating them on the fly. As a result, in this section the message alignment mappings $p(z_{k,l}|t_{k,l})$ are just matched to the design-E_b/N_0 of the applied Information Bottleneck LDPC decoders. The identification of the used design-E_b/N_0 for the applied decoders is performed completely analogous to Section 5.2.4.

Bit Error Rate Performance with Quadrature Amplitude Modulation and Message Alignment

This section investigates the bit error rate performances of several regular LDPC codes and $|\mathcal{S}|$-QAM modulation under Information Bottleneck decoding with message alignment. The Information Bottleneck decoding is compared with double precision belief propagation decoding and min-sum decoding. The used channel model is AWGN.

The following coding and modulation schemes will be investigated in this section:

1. The length $8,000$ $(3,6)$ regular LDPC code 8000.4000.3.483 from [68] with code rate $R = 0.5$ (*regular code a)*) together with 16-QAM modulation. This pair is termed *coding and modulation scheme a)*.

2. A length $12,000$ $(7, 20)$ randomly constructed regular LDPC code with code rate $R = 0.65$ together with 64-QAM. This pair is termed *coding and modulation scheme b)*.

Both applied QAM schemes use a Gray labeling where half of the bits assigned to a modulation symbol s_k only depend on s_k^{re} and half of them only depend on s_k^{im}.

The probability distributions $p(c_{k,l}|t_{k,l})$ from the design process of the Information Bottleneck demodulator lookup table allow to calculate LLRs $L(c_{k,l}|t_{k,l})$ from the integer indices $t_{k,l}$ which can be used to pair the conventional decoding algorithms with the Information Bottleneck quantization and demodulation approach. This provides a fair reference for the Information Bottleneck decoders.

The results to be presented in the following will be labeled with the respective output bit widths of the quantizer and the applied Information Bottleneck demodulator lookup table. However, it is important to note that the conventional decoding algorithms still processed real valued LLRs $L\left(c_{k,l}|t_{k,l}\right)$ in double precision. Moreover, for the conventional decoders, the quantizers and the demodulation lookup tables were matched to the actual channel noise variance in order to provide them meaningful LLRs. For the Information Bottleneck decoders, in contrast, all components were used only matched to their design-E_b/N_0.

Figure 5.34 shows the bit error rate of *coding and modulation scheme a)* for various demodulation and decoding approaches. The maximum number of decoder iterations was $i_{\max} = 50$ for all LDPC decoders applied in this experiment. The curves labeled *double channel LLRs* refer to the conventional decoding algorithms working on quasi continuous LLRs which were calculated from the continuous AWGN channel output in double precision. The curve labeled *Information Bottleneck mismatched,* 4 *bit all components* refers to a setup where the quantizer, the applied Information Bottleneck demodulator lookup table and the Information Bottleneck decoder all used an identical bit width of $q = 4$ bits. Message alignment was applied to the output of the Information Bottleneck demodulator lookup table for the Information Bottleneck system, such that 4 bit messages $z_{k,l}$ were used as channel messages for decoding in this system with an Information Bottleneck decoder.

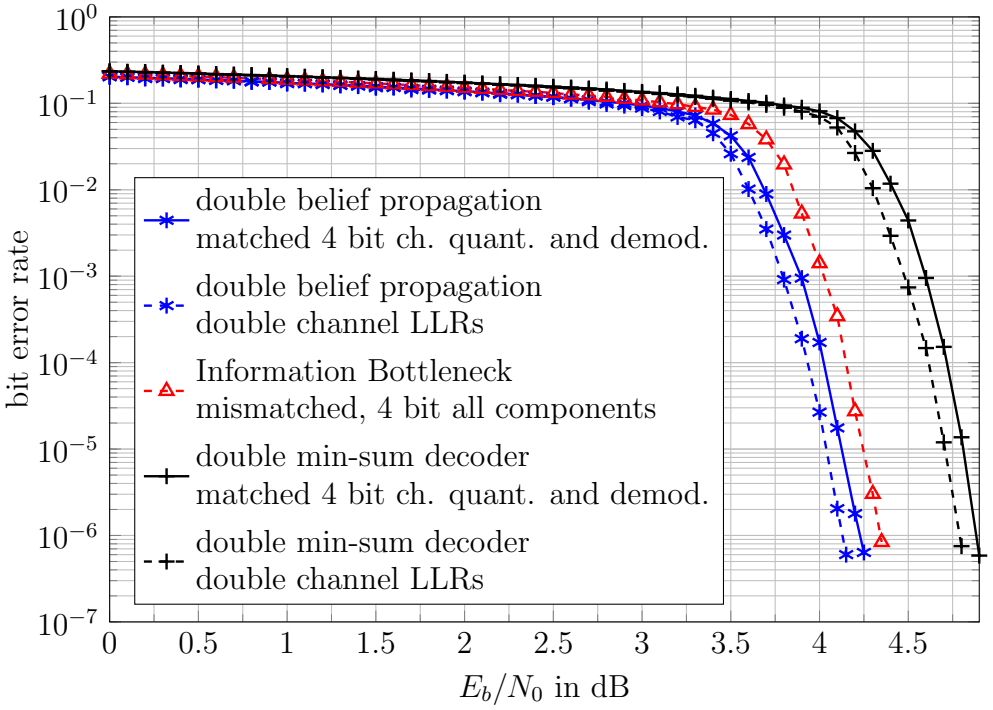


Figure 5.34: Bit error rates of various decoders for *coding and modulation scheme a)* (16-QAM, $R = 0.5$) under AWGN over E_b/N_0. All decoders performed $i_{\max} = 50$ iterations.

First, the influence of the integer based demodulation with LLR based decoding shall be studied. Similar as in the BPSK case, the $q = 4$ bit quantized Information Bottleneck quantization and demodulation approach, degrades the performance of the LLR based decoders roughly by 0.1 dB over E_b/N_0 as it can be seen by comparing the performances of the systems without quantization to the ones with belief propagation decoding and min-sum decoding and Information Bottleneck quantization and demodulation (solid blue/black curves versus dashed blue/black curves, respectively).

The proposed $q = 4$ bit Information Bottleneck system which only uses static lookup tables for demodulation and inside the Information Bottleneck decoder (dashed red curve) approaches the performance of its reference system with quantized channel output and belief propagation decoding also up to 0.1 dB over E_b/N_0 in the waterfall region of the code. The conventional system with min-sum decoding again is outperformed by the Information Bottleneck system.

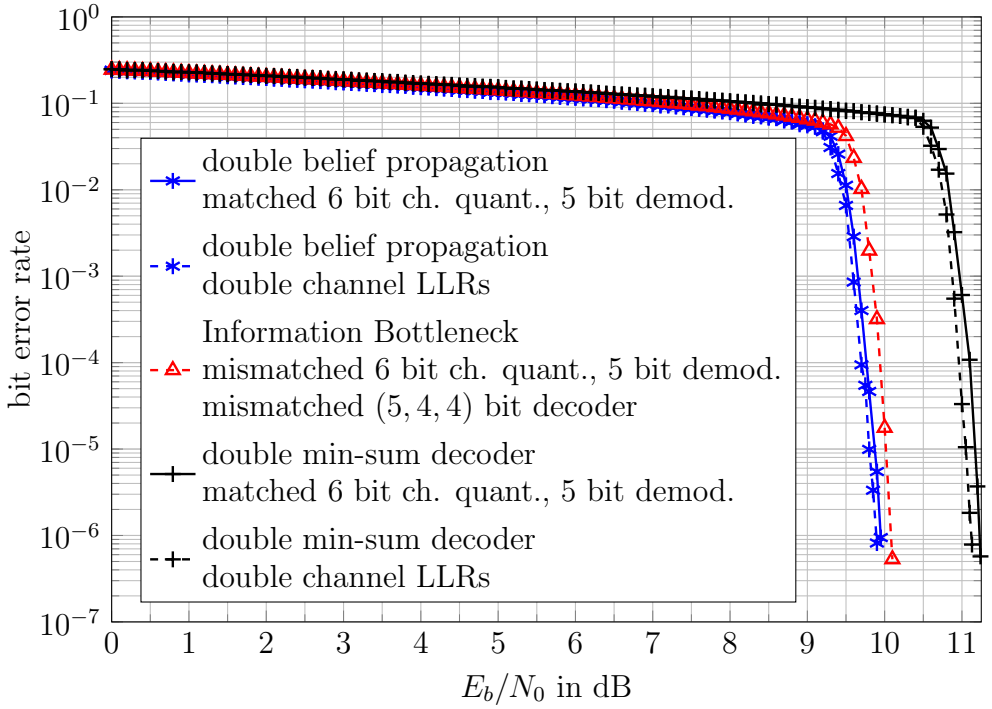


Figure 5.35: Bit error rates of various decoders for *coding and modulation scheme b)* (64-QAM, $R = 0.65$) under AWGN over E_b/N_0. All decoders performed $i_{\max} = 40$ iterations.

Figure 5.35 investigates a similar setup for *coding and modulation scheme b)*. In this case, the maximum number of decoding iterations was set to $i_{\max} = 40$ for all decoders. For the 64-QAM modulation scheme in *coding and modulation scheme b)*, it is intuitive that the number of quantization levels of the channel output quantizer has to be larger than for BPSK and 16-QAM to preserve a significant amount of relevant information. As a result, a 6 bit channel output quantizer has been applied on the continuous received samples $\tilde{r}_k^{\text{re}}$ and $\tilde{r}_k^{\text{im}}$. The quantized $r_k^{\text{re}}, r_k^{\text{im}} \in \{0, 1, \ldots, 63\}$ were fed to an Information Bottleneck demodulator similar to the one from Figure 5.30, but extended to cope with 64-QAM. The output cardinalities of the tables $p(t_{k,l}|r_k^{\text{re}})$ and $p(t_{k,l}|r_k^{\text{im}})$ in this demodulator lookup table were set to 32, resulting in a 5 bit Information Bottleneck demodulator. The output integers $t_{k,l}$ were again used to calculate LLRs for the conventional decoders. Also in this experiment, the quantizer and the demodulator were matched to the actual E_b/N_0 on the channel for the conventional LLR based decoders, while these components

were used entirely mismatched for the Information Bottleneck decoder. For the Information Bottleneck decoder again message alignment was performed and the integers $z_{k,l}$ were used as channel messages for decoding. The applied Information Bottleneck decoder used the 5 bit channel messages $z_{k,l}$ and 4 bit messages to represent the variable-to-check and the check-to-variable node messages. Hence, the cardinalities of the quantizer, the demodulator and the decoder were chosen to successively reduce the bit width from the quantizer to the decoder, such that the decoder worked on 4 bit messages in its iterative decoding process.

Figure 5.35 shows that this approach yields a completely integer based decoding and demodulation scheme for *coding and modulation scheme b)* which can compete with a conventional system with double precision demodulation and decoding. The loss with respect to the fair competitor with belief propagation decoding is around 0.1 dB over E_b/N_0 in the waterfall region. The min-sum schemes with and without channel output quantization are again outperformed.

5.3.5 Summary

In this section, a technique to pair Information Bottleneck channel output quantizers with Information Bottleneck LDPC decoders for QAM modulated signals has been developed and investigated. After having introduced an Information Bottleneck graph of an Information Bottleneck demodulator, a technique called message alignment has been developed. Message alignment is required due to the fact that different virtual channel exists for the bits assigned to a modulation symbol in a higher order modulation scheme like QAM. As a result, the beliefs on the codeword bits are not automatically aligned for a certain integer output of the demodulator, but depend on the position of the bit in the bit label.

With message alignment, the beliefs on the codeword bits for an incoming integer message become approximately independent of the bit label position. This allows to construct an Information Bottleneck LDPC decoder with the technique presented in Section 5.2. Similar as it has been observed for BPSK, the resulting decoders offer bit error rate performance very close to that of double precision belief propagation decoding.

5.4 Information Bottleneck Decoding of Irregular LDPC Codes

Although the conventional LLR based message passing algorithms like the belief propagation and the min-sum decoding algorithm are essentially identical for regular and irregular LDPC codes, it turns out that a generalization of the Information Bottleneck decoding principle for irregular LDPC codes is not that simple. The reason is that in an irregular LDPC code, various node degrees exist and a receiving node cannot resolve the degree of a node delivering an incoming integer message due to the quasi random structure of the parity-check matrix. However, the joint distributions $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(i)}\right)$ and $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(i)}\right)$ which have to be processed to construct the node operations depend on the degree of the delivering node. This problem is very similar to the message alignment problem which appears at the output of an Information Bottleneck demodulator for QAM described in Section 5.3.1.

In this section, the decoder design principle from Section 5.2 is generalized to also work for arbitrary irregular LDPC codes by the inclusion of message alignment in the decoder construction process. The presented ideas were published in [72, 73]. It is explained in detail, why a message alignment step is required before propagating integer messages from nodes with various degrees into the decoding graph to construct Information Bottleneck LDPC decoders with good performance. Finally, the bit error rate performance of the resulting receivers is investigated.

Related works by other authors dealing with a similar problem setup are [33, 35]. The authors of these works also aim for decoding of irregular LDPC codes with lookup tables that preserve relevant information and use a discrete density evolution technique to determine the required probability distributions. However, their design approach differs from the one presented in the following in several aspects. Some of the key differences are mentioned in the following.

First, the decoder design in [33, 35] is not based on Algorithm 4 which is a modified variant of the sequential Information Bottleneck algorithm. In contrast, it uses an algorithm from [58] which has originally been proposed for quantizer design.

Second, other than in this work, no message alignment step is included as an additional step in [33, 35]. Instead, a joint lookup table design for nodes with different degrees is proposed in [35]. Moreover, [33, 35] lie a special focus on a hybrid approach which uses relevant information preserving lookup tables only for the variable node updates, but a min-sum like operation for the check node updates.

Finally, the decoders from [33, 35] for irregular LDPC codes require optimization of the code ensembles of the applied LDPC codes to achieve good performance. Therefore, they do not seem to be applicable to LDPC ensembles which are not optimized for the lookup table based decoders from these works. In contrast, some code ensembles seem to be *ill-suited* for the decoding methods described in [33, 35], as it is stated in these works. To the best of the author's knowledge, such a restriction does not exist for the Information Bottleneck decoders presented in the following. In fact, some of the codes used for performance evaluation at the end of this section were deliberately chosen to be representants of ensembles stated to be problematic in [33, 35].

5.4.1 Application of Message Alignment in Information Bottleneck Decoder Design

In Section 5.3.1 it has been discussed that the assignment of several bits to one modulation symbol in QAM causes challenges in the construction of Information Bottleneck LDPC decoders due to the fact that the check nodes cannot distinguish the bit label position l of a codeword bit from their neighboring variable nodes.

A very similar problem exists in the Information Bottleneck decoding of irregular LDPC codes. Of course, it is possible to construct lookup table based node operations for all distinct node degrees with the Information Bottleneck method straightforwardly, just as at it was explained for regular codes in Section 5.2. In this case, we have to distinguish the integer messages $t_{\mathrm{c}\to\mathrm{v}}^{(d_{\mathrm{c}},i)}$ from check nodes with different degrees d_{c} and $t_{\mathrm{v}\to\mathrm{c}}^{(d_{\mathrm{v}},i)}$ from variable nodes with different degrees d_{v}. The significant difference in comparison to regular LDPC codes is that the node operations differ for nodes with different degrees because nodes with distinct degrees have to process different numbers of incoming messages.

The number of processed incoming messages influences the joint distributions $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right)$ and $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right)$ in an Information Bottleneck decoder. As a result, if the integers $t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}$ and $t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}$ were propagated into the decoding graph directly, the incoming messages at the nodes would not be aligned.

Consider, for example, the message generation process of a variable node which represents any codeword bit x in an Information Bottleneck LDPC decoder from incoming integer messages delivered by check nodes with different degrees. The degree d_v variable node receives incoming messages $y_n^{(i)}$ $n = 1, 2, \ldots, d_\mathrm{v}$ from its connected check nodes and a channel message y_0 from the channel.

From the perspective of the variable nodes with incoming messages $y_n^{(i)}$, the joint distributions $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right)$ correspond to the distributions $p\left(x, y_n^{(i)}|d_\mathrm{c}\right)$, that is, $t_{\mathrm{c}\to\mathrm{v}}^{(i)}$ is relabeled to $y_n^{(i)}$. The condition on d_c in $p\left(x, y_n^{(i)}|d_\mathrm{c}\right)$ expresses the condition that a message $y_n^{(i)}$ received along an edge originates from a check node with degree d_c.

On average over all edges connected to any variable node in the Tanner graph, the distribution $p\left(x, y_n^{(i)}\right)$ is given by

$$p\left(x, y_n^{(i)}\right) = \sum_{d_\mathrm{c}=2}^{d_\mathrm{c}^{\max}} p\left(x, y_n^{(i)}|d_\mathrm{c}\right) \rho_{d_\mathrm{c}}. \tag{5.44}$$

The coefficient ρ_{d_c} from the check node edge degree distribution of the applied code in this equation reflects the probability that any message received by the variable node originates from a degree d_c check node.

Equation (5.44) looks very similar to Equation (5.37) which appeared in the check node design under $|\mathcal{S}|$-QAM. Here, it determines the joint distribution $p\left(x, y_n^{(i)}\right)$ to be used for the generation of the variable node operations in an Information Bottleneck LDPC decoder. We know from Equation (5.37) that the averaging step involved in this equation can cause a catastrophic result if the distributions $p\left(x|y_n^{(i)}, d_\mathrm{c}\right)$ are not aligned for different node degrees d_c, that is, these distributions have to be similar for identical $y_n^{(i)}$, but different d_c. This cannot be guaranteed in the proposed decoder design because node operations for distinct degrees are designed independently.

Similarly, for the check node design process with the Information Bottleneck method, the distribution

$$p\left(b_n, y_n^{(i)}\right) = \sum_{d_\mathrm{v}=1}^{d_\mathrm{v}^{\max}} p\left(b_n, y_n^{(i)} | d_\mathrm{v}\right) \lambda_{d_\mathrm{v}} \tag{5.45}$$

has to be used. Here, the distributions $p\left(b_n, y_n^{(i)} | d_\mathrm{v}\right)$ correspond to the distributions $p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right)$ with relabeled $x = b_n$ and $t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)} = y_n^{(i)}$. These distributions are the side products of the Information Bottleneck algorithms applied to design the variable node operations with various degrees d_v. Obviously, the same problem concerning non-aligned messages exists for the check node design process.

As a result, a message alignment step has to be included after all variable node and all check node operations for nodes with different degrees have been constructed with the Information Bottleneck method in each decoding iteration. The decoder design process for irregular codes with message alignment is summarized in the following.

Message Alignment of the Check Nodes

The decoder design for irregular codes again starts at the check nodes as described in Section 5.2.1 for regular codes. The only difference is, that node operations for all appearing check node degrees are constructed and stored. Rather than propagating the outputs $t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}$ for every node degree directly into the decoding graph, these integer messages shall first be aligned using message alignment mappings $p\left(z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)} | t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right)$ for every node degree.

Once the check node lookup tables have been constructed, one identifies the node degree which provides the largest mutual information $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right)$ of all node degrees d_c. The joint distribution required to calculate this mutual information for every node degree d_c is $p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right)$. It is delivered by the Information Bottleneck algorithm used to design the check node operations automatically.

The node degree with the largest mutual information is denoted d_c'. Its message alignment mapping is initialized with an identity mapping $p\left(z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c}',i)} | t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c}',i)}\right)$. The conditional distribution $p\left(x | z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c}',i)}\right)$ then is identical to $p\left(x | t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c}',i)}\right)$ with relabeled $t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c}',i)} = z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c}',i)}$.

The other message alignment mappings are initially determined according to the rule

$$z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)} = \arg \min_{z_{\mathrm{c}\to\mathrm{v}}^{(d'_\mathrm{c},i)} \in \mathcal{T}_\mathrm{chk}} D_\mathrm{KL} \left\{ p\left(x|t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right) \,|\, p\left(x|z_{\mathrm{c}\to\mathrm{v}}^{(d'_\mathrm{c},i)}\right) \right\} \; \forall t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)} \in \mathcal{T}_\mathrm{chk}. \quad (5.46)$$

Afterwards, one determines

$$p\left(x, z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right) = \sum_{t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)} \in \mathcal{T}_\mathrm{chk}} p\left(z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}|t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right) p\left(x, t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right) \quad (5.47)$$

for all distinct node degrees d_c.

The integer messages $z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}$ form the incoming messages $y_n^{(i)}$ of the variable nodes. Hence, $p\left(x, y_n^{(i)}|d_\mathrm{c}\right)$ is given by $p\left(x, z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right)$ with relabeled $z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)} = y_n^{(i)}$ such that the condition on d_c in $p\left(x, y_n^{(i)}|d_\mathrm{c}\right)$ again expresses that the delivering node degree of message $y_n^{(i)}$ is d_c. Using this knowledge, one now calculates $p\left(x, y_n^{(i)}\right)$ using Equation (5.44) and from this $p\left(x|y_n^{(i)}\right)$ for the current message alignment mappings. Please note that this step includes processing the check node edge degree distribution of the code.

One then starts over from Equation (5.46), but using $p\left(x|y_n^{(i)}\right)$ as the second argument of the Kullback-Leibler divergence. Once a stable solution is found for all message alignment mappings $p\left(z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}|t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right)$, these mappings are stored. The resulting distribution $p\left(x, y_n^{(i)}\right)$ is used in the following step to design the variable node operations for all node degrees d_v, just as it has been described in Section 5.2.2 for regular LDPC codes.

Message Alignment of the Variable Nodes

An equivalent message alignment is performed for all variable nodes with distinct degrees in each decoder iteration. First, node operations for all occuring node degrees d_v are constructed using the joint distribution $p\left(x, y_n^{(i)}\right)$ from the design of the check nodes, exactly as it has been explained in Section 5.2.2 for regular codes.

Then also the variable node degree with the largest mutual information $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{v}\to\mathrm{c}}^{(d'_\mathrm{v},i)}\right)$ of all node degrees d_v is identified and its message alignment mapping $p\left(z_{\mathrm{v}\to\mathrm{c}}^{(d'_\mathrm{v},i)}|t_{\mathrm{v}\to\mathrm{c}}^{(d'_\mathrm{v},i)}\right)$ is initialized with an identity mapping.

The other message alignment mappings are determined according to the rule

$$z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)} = \arg \min_{z_{\mathrm{v}\to\mathrm{c}}^{(d'_\mathrm{v},i)} \in \mathcal{T}_\mathrm{var}} D_\mathrm{KL} \left\{ p\left(x|t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right) \,|\, p\left(x|z_{\mathrm{v}\to\mathrm{c}}^{(d'_\mathrm{v},i)}\right) \right\} \; \forall t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)} \in \mathcal{T}_\mathrm{var}. \quad (5.48)$$

Afterwards, for all distinct appearing node degrees d_v, one determines the joint probability distributions

$$p\left(x, z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right) = \sum_{t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)} \in \mathcal{T}_\mathrm{var}} p\left(z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}|t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right) p\left(x, t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right). \quad (5.49)$$

From the perspective of the check nodes, the $z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}$ are the incoming messages $y_n^{(i+1)}$ in the next decoding iteration and the codeword bit $x = b_n$. As a result, $p\left(b_n, y_n^{(i+1)}|d_\mathrm{v}\right)$ equals $p\left(x, z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right)$ with relabeled $x = b_n$ and $z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)} = y_n^{(i+1)}$. Analogous to Equation (5.45) it is then possible to determine $p\left(b_n, y_n^{(i+1)}\right)$ and $p\left(b_n|y_n^{(i+1)}\right)$. This step involves processing the variable node degree distribution from an edge perspective. Also for the variable nodes one starts over from Equation (5.48) with $p\left(b_n|y_n^{(i+1)}\right)$ as the second argument of the Kullback-Leibler divergence and repeats these steps until the message alignment mappings $p\left(z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}|t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right)$ do not change any more.

Once all message alignment mappings $p\left(z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}|t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right)$ are constructed, one stores them. The final distribution $p\left(b_n, y_n^{(i+1)}\right)$ is passed back to the construction process of the check nodes for the next decoding iteration.

5.4.2 Mismatched Quantizer and Decoder Design

Also for irregular LDPC codes, the channel output quantizer, the variable node and the check node lookup tables just as the message alignment mappings have to be designed offline and for a fixed design-E_b/N_0. This again raises the question how a design-E_b/N_0 can be identified which offers good bit error rate performance over a wide range of E_b/N_0 on the channel. In Section 5.2.4 it was proposed to investigate the variable node output information $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_\mathrm{dec}^{(i)}\right)$ for this purpose for regular LDPC codes. For an irregular LDPC code, however, no single variable node output information exists, but it depends on the node degrees of the variable nodes. Hence, $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_\mathrm{dec}^{(d_\mathrm{v},i)}\right)$ differs for the different variable node degrees d_v. As a result, one could investigate several output information measures for the various node degrees.

From the bit error rate simulations presented in the next section, it was found that the mutual information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(d_{\mathrm{v}}^{\max},i)}\right)$ is a useful measure to find a lower bound for the design-E_b/N_0 which should be used. Variable nodes with the maximum node degree $d_{\mathrm{v}}^{\max}$ will typically provide the most reliable bit decisions and, therefore, the maximum $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(d_{\mathrm{v}},i)}\right)$ of all node degrees d_{v}. It was found experimentally that choosing the design-E_b/N_0 such that the output information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(d_{\mathrm{v}}^{\max},i_{\max}-1)}\right)$ does not approach 1 bit in the design process of the decoder over the decoder iterations results in an early error floor of the resulting decoders. A design-E_b/N_0 value which guarantees that the maximum output information approaches 1 bit in the final decoder iteration can serve as a lower bound for the design-E_b/N_0 which should be chosen. However, from this lower bound for the design-E_b/N_0, the best decoder so far has to be found by increasing the design-E_b/N_0 in small steps and investigating the performance of the resulting decoders using bit error rate simulations. Aggregating measures, for example, the expectation of the output information over the nodes with various degrees were not found useful for identification of a well-suited design-E_b/N_0. The influence of the design-E_b/N_0 on the decoder performance will be investigated as a part of the performance evaluation in the next section.

5.4.3 Results and Discussion

In this section, the performance of Information Bottleneck LDPC decoders for irregular LDPC codes under AWGN with BPSK modulation shall be studied and compared to the state-of-the-art algorithms. Just as in the investigation for regular LDPC codes in Section 5.2.5, first the influence of the design-E_b/N_0 on the decoder performance is analyzed.

The following irregular LDPC codes are considered for the evaluation of the bit error rate performance over E_b/N_0 in this section:

1. The length $64,800$ irregular LDPC code from the DVB-S2 standard [43] with rate $R = 0.5$. This code is termed *irregular code a)*.

2. A length $14,000$ irregular LDPC code with rate $R = 0.5$ from a code ensemble described in [74]. The corresponding parity-check matrix was constructed using the progressive edge growth construction algorithm from [75]. This code is termed *irregular code b)*.

Table 5.5: Code parameters of the applied irregular LDPC codes.

irregular code a), code rate $R = 0.5$, codeword length $N_\mathrm{v} = 64,800$

$$\begin{aligned}\lambda(d) = \;& 4.40919052 \cdot 10^{-6} + 0.285706727\, d + 0.257143991\, d^2 \\ & +0.457144873\, d^7 \\ \rho(d) = \;& 2.64551431 \cdot 10^{-5}\, d^5 + 0.999973545\, d^6\end{aligned}$$

irregular code b), code rate $R = 0.5$, codeword length $N_\mathrm{v} = 14,000$

$$\begin{aligned}\lambda(d) = \;& 0.23802\, d + 0.20997\, d^2 + 0.03492\, d^3 + 0.12015\, d^4 \\ & +0.01587\, d^6 + 0.0048\, d^{13} + 0.37627\, d^{14} \\ \rho(d) = \;& 0.98013\, d^7 + 0.01987\, d^8\end{aligned}$$

irregular code c), code rate $R = 0.75$, codeword length $N_\mathrm{v} = 64,800$

$$\begin{aligned}\lambda(d) = \;& 4.40919052 \cdot 10^{-6} + 0.1428489544\, d + 0.571431091\, d^2 \\ & +0.285715545\, d^{11} \\ \rho(d) = \;& 5.73194767 \cdot 10^{-5}\, d^{12} + 0.999942681\, d^{13}\end{aligned}$$

irregular code d), code rate $R = 0.25$, codeword length $N_\mathrm{v} = 64,800$

$$\begin{aligned}\lambda(d) = \;& 5.144059383 \cdot 10^{-6} + 0.49999228\, d + 0.16666752\, d^2 \\ & +0.33333505\, d^{11} \\ \rho(d) = \;& 1.543217815 \cdot 10^{-5}\, d^2 + 0.99998457\, d^3\end{aligned}$$

3. A length $64,800$, irregular LDPC code from the DVB-S2 standard [43] with rate $R = 0.75$. This code is termed *irregular code c)*.

4. A length $64,800$, irregular LDPC code from the DVB-S2 standard [43] with rate $R = 0.25$. This code is termed *irregular code d)*.

The edge degree distributions and the parameters of the codes are listed in Table 5.5 for a quick overview.

Performance and Influence of the Design-E_b/N_0

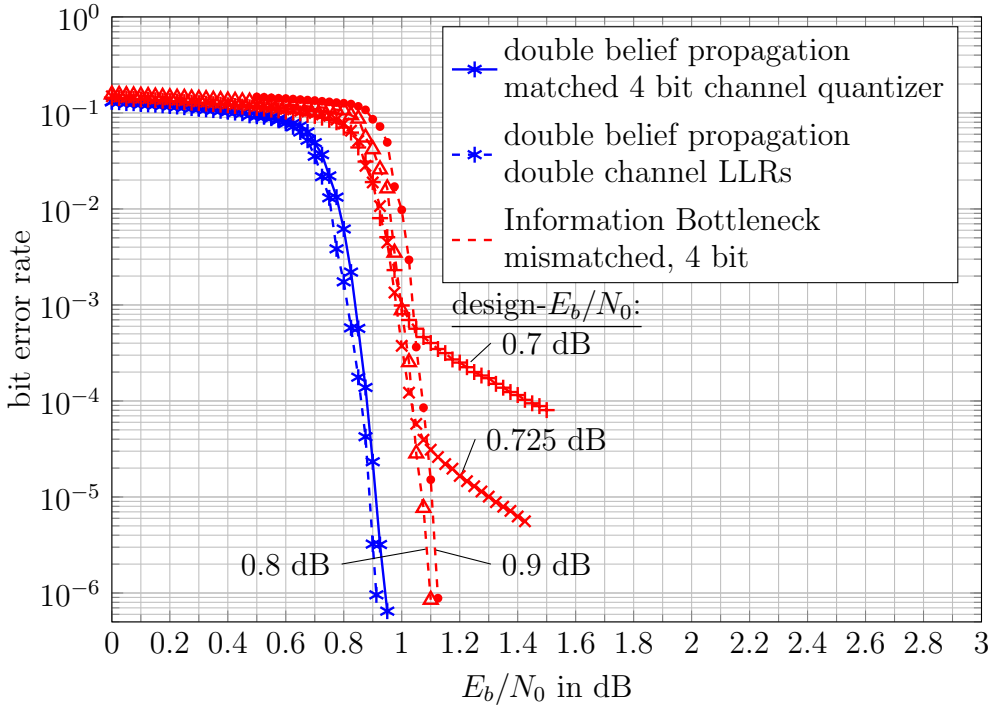
Figure 5.36a shows the bit error rate performances of *irregular code a)* under belief propagation decoding and several 4 bit Information Bottleneck decoders with message alignment and different design-E_b/N_0. The maximum number of decoder iterations was set to $i_\mathrm{max} = 50$ in this experiment. Underneath in Figure 5.36b, the variable node output information $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_\mathrm{dec}^{(d_\mathrm{v}^\mathrm{max}, i)}\right)$ for the

maximum variable node degree $d_{\mathrm{v}} = 8$ from the design process of the decoder is shown. Equivalent results are provided for *irregular code b)* with a maximum of $i_{\max} = 70$ decoding iterations in Figure 5.37, for *irregular code c)* with a maximum of $i_{\max} = 40$ decoding iterations in Figure 5.38 and for *irregular code d)* with a maximum of $i_{\max} = 60$ decoding iterations in Figure 5.39.

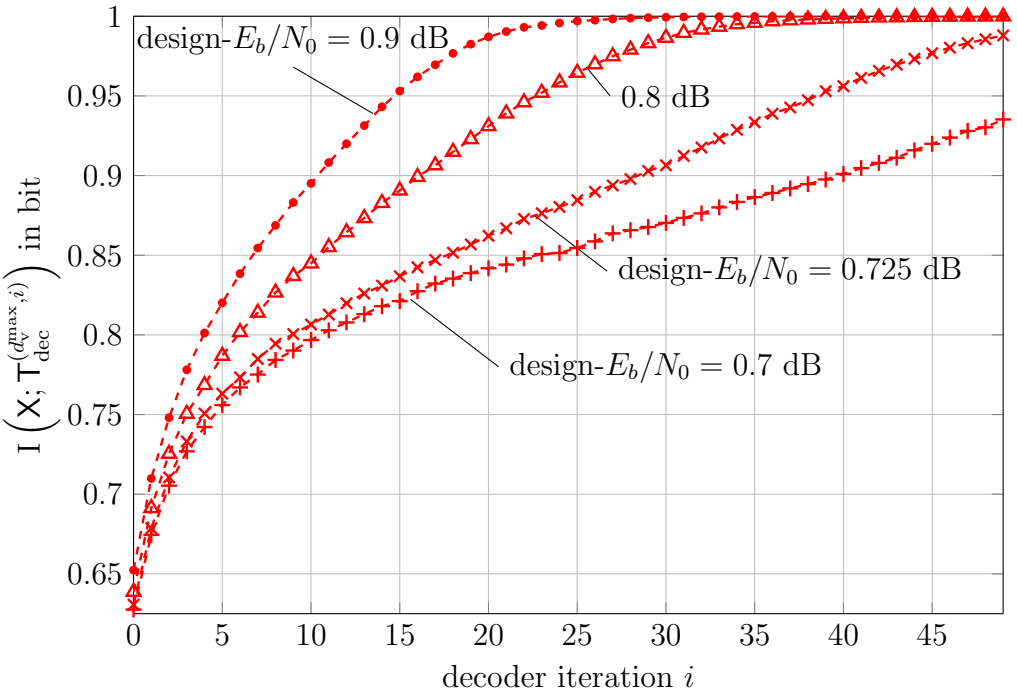
A comparison of the bit error rates from Figure 5.36a with the curves from Figure 5.36b reveals that choosing the design-E_b/N_0 too small, such that the maximum variable node output information in the design process of the decoder does not approach 1 bit, results in an early error floor. This effect can equivalently be observed for *irregular codes b), c)* and *d)* in Figures 5.37-5.39.

The observed error floor can be mitigated by choosing the design-E_b/N_0 such that $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{dec}}^{(d_{\mathrm{v}}^{\max}, i)}\right)$ approaches 1 bit in the final decoder iteration. Choosing the design-E_b/N_0 that high, however, also slightly degrades the decoder performance in the cliff regions, as it is especially visible for *irregular codes a)* and *b)*. Obviously, a trade-off exists between the error floor performance of the Information Bottleneck decoders and their loss with respect to the belief propagation decoder. Choosing larger design-E_b/N_0 will typically result in lower error floors at the expense of small losses in the waterall region. The discussion of the bit error rates above tells that a minimum design-E_b/N_0 should be chosen which results in the fact that $\mathrm{I}\left(\mathsf{X}; \mathsf{T}_{\mathrm{dec}}^{(d_{\mathrm{v}}^{\max}, i)}\right)$ approaches 1 bit in the design process of the decoder. However, this should be considered as a minimum requirement. Typically a higher design-E_b/N_0 will be required to prevent an early flattening of the bit error rate curves.

The performances of the best found 4 bit Information Bottleneck decoders in the waterfall regions of the used codes are compared with the ones of the double precision belief propagation decoder with a 4 bit channel output quantizer next. The 4 bit Information Bottleneck decoder for *irregular code a)* with design-$E_b/N_0 = 0.8$ dB approaches the performance of the double precision belief propagation decoder with 4 bit channel output quantization up to 0.2 dB over E_b/N_0 in the waterfall region. A similar result holds for *irregular code b)* for its 4 bit Information Bottleneck decoder with design-$E_b/N_0 = 0.7$ dB. Please recall that *irregular code a)* and *irregular code b)* both have code rate $R = 0.5$. For the higher rate $R = 0.75$ *irregular code c)*, the gap of the 4 bit Information Bottleneck decoder with design-$E_b/N_0 = 1.95$ dB reduces to

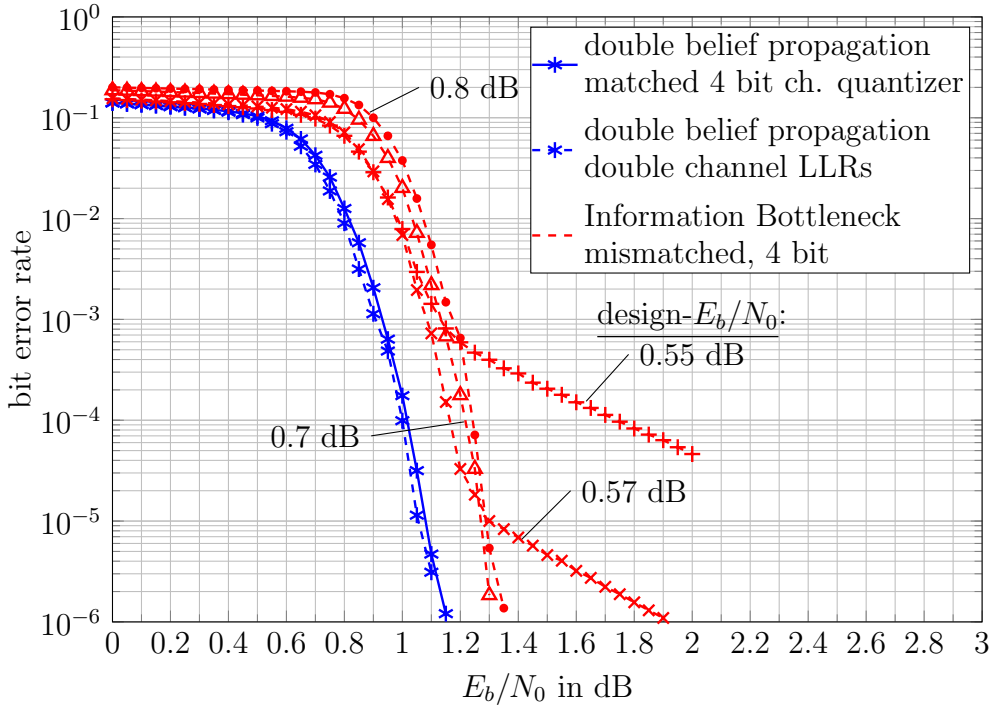


(a) Bit error rate of *irregular code a)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.

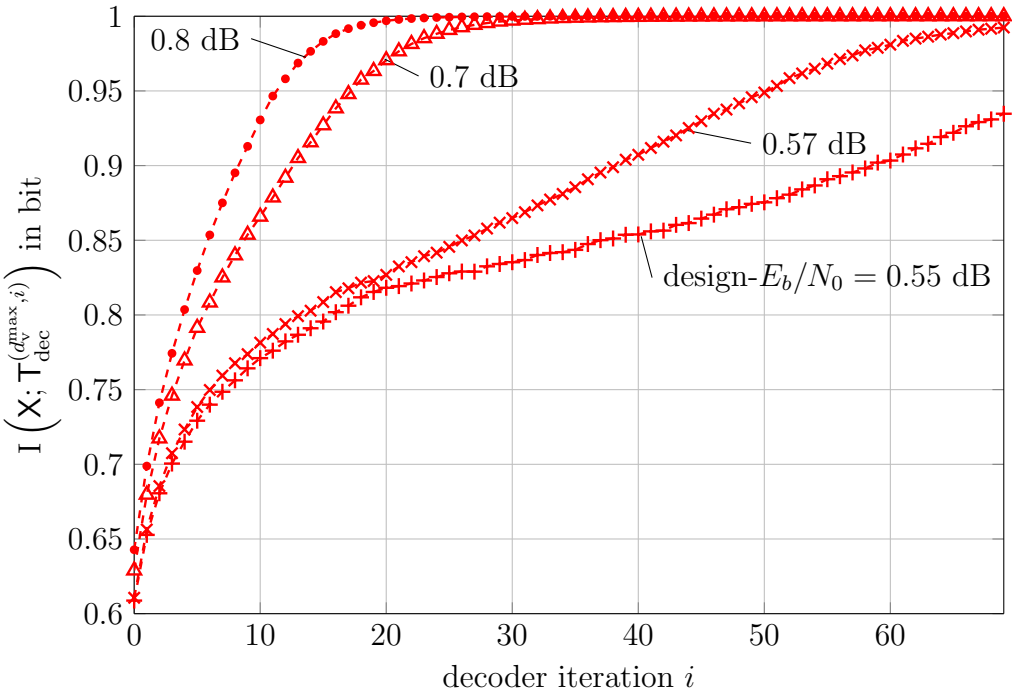


(b) Maximum variable node output information for the corresponding ensemble.

Figure 5.36: Analysis of Information Bottleneck LDPC decoders for *irregular code a)* with BPSK modulation under AWGN over E_b/N_0. Information Bottleneck decoders with 4 bit messages and $i_{\max} = 50$ iterations are applied.

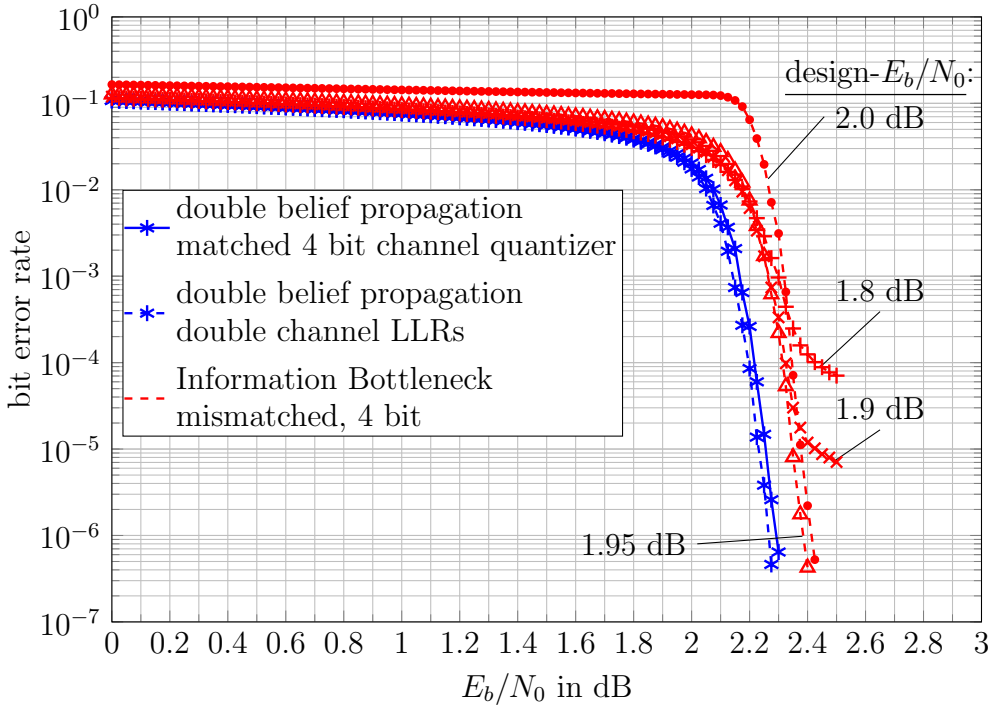


(a) Bit error rate of *irregular code b)* ($R = 0.5$) with $i_{\max} = 70$ decoding iterations.

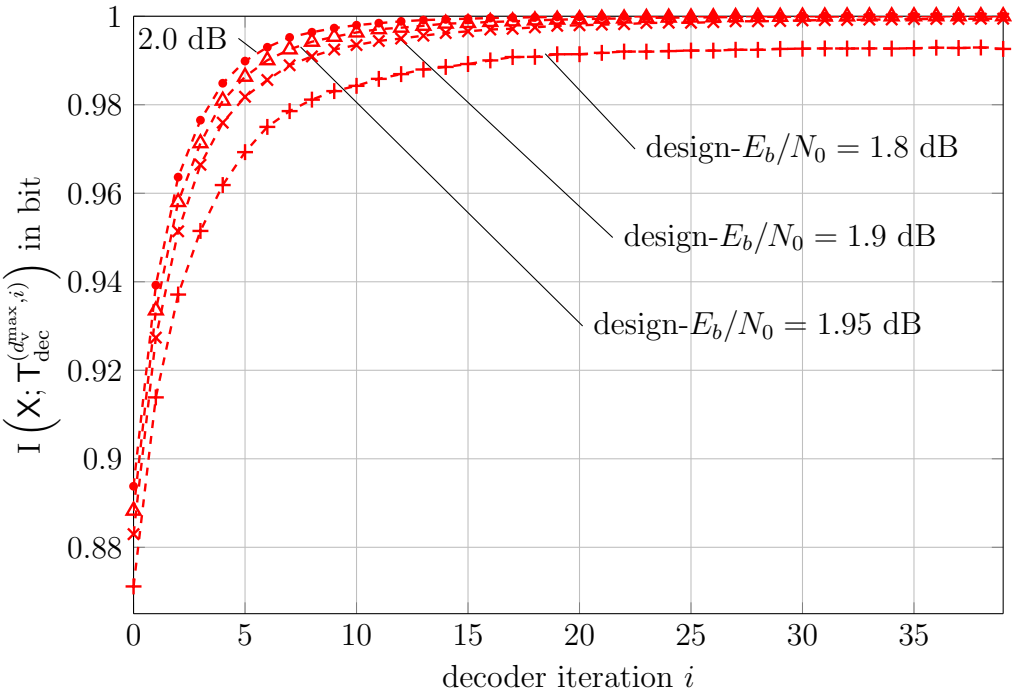


(b) Maximum variable node output information for the corresponding ensemble.

Figure 5.37: Analysis of *irregular code b)* with BPSK modulation under AWGN over E_b/N_0. Information Bottleneck decoders with 4 bit messages and $i_{\max} = 70$ iterations are applied.

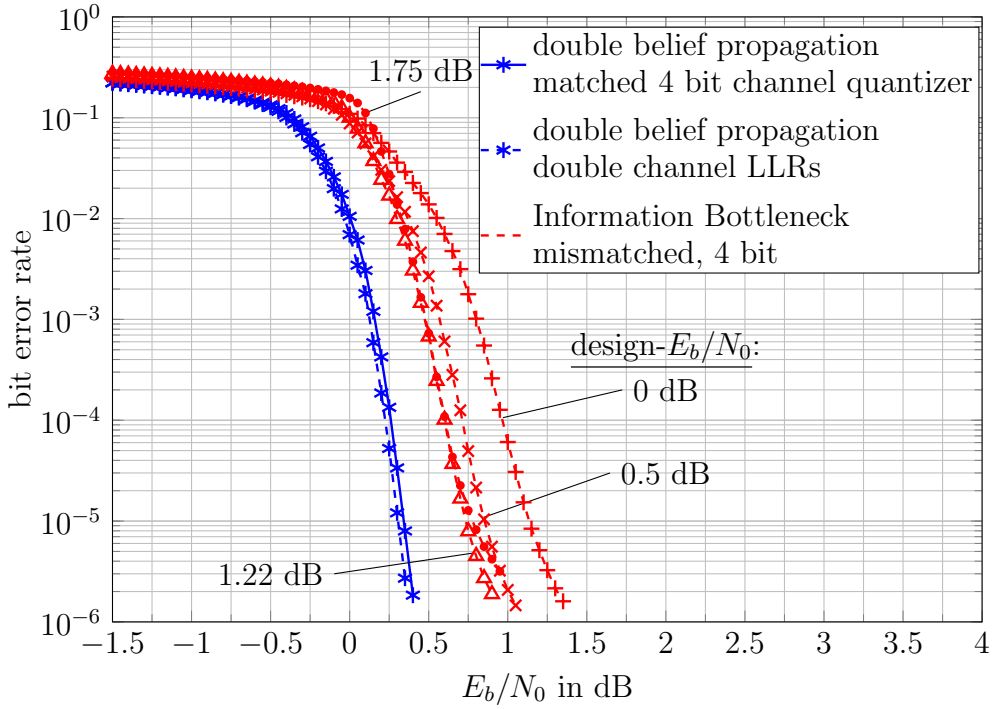


(a) Bit error rate of *irregular code c)* ($R = 0.75$) with $i_{\max} = 40$ decoding iterations.

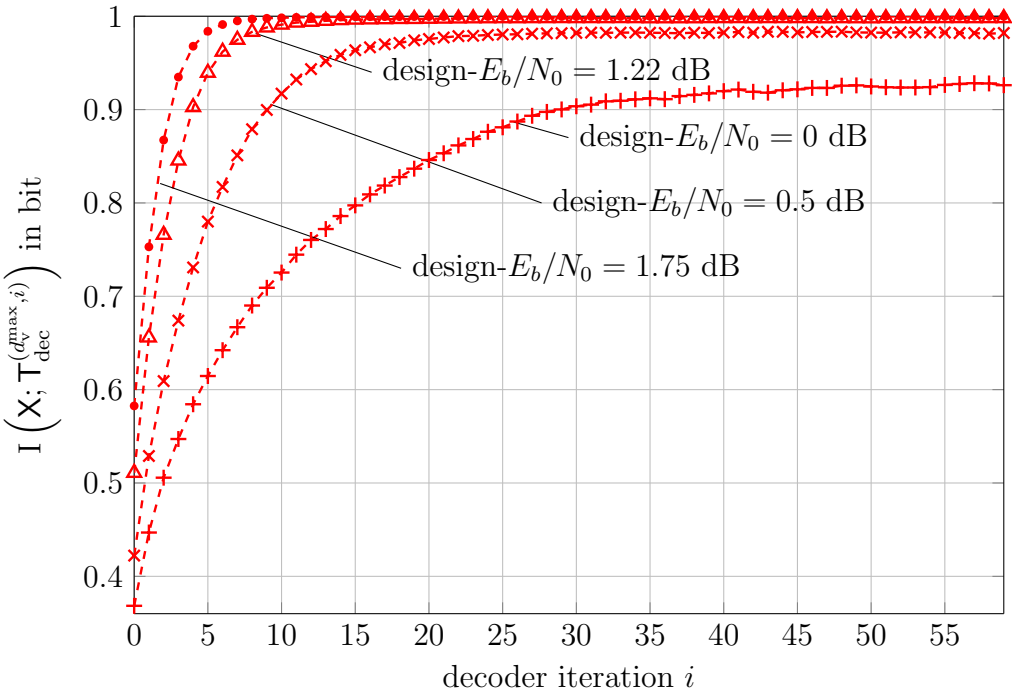


(b) Maximum variable node output information for the corresponding ensemble.

Figure 5.38: Analysis of *irregular code c)* with BPSK modulation under AWGN over E_b/N_0. Information Bottleneck decoders with 4 bit messages and $i_{\max} = 40$ iterations are applied.



(a) Bit error rate of *irregular code d)* ($R = 0.25$) with $i_{\max} = 60$ decoding iterations.



(b) Maximum variable node output information for the corresponding ensemble.

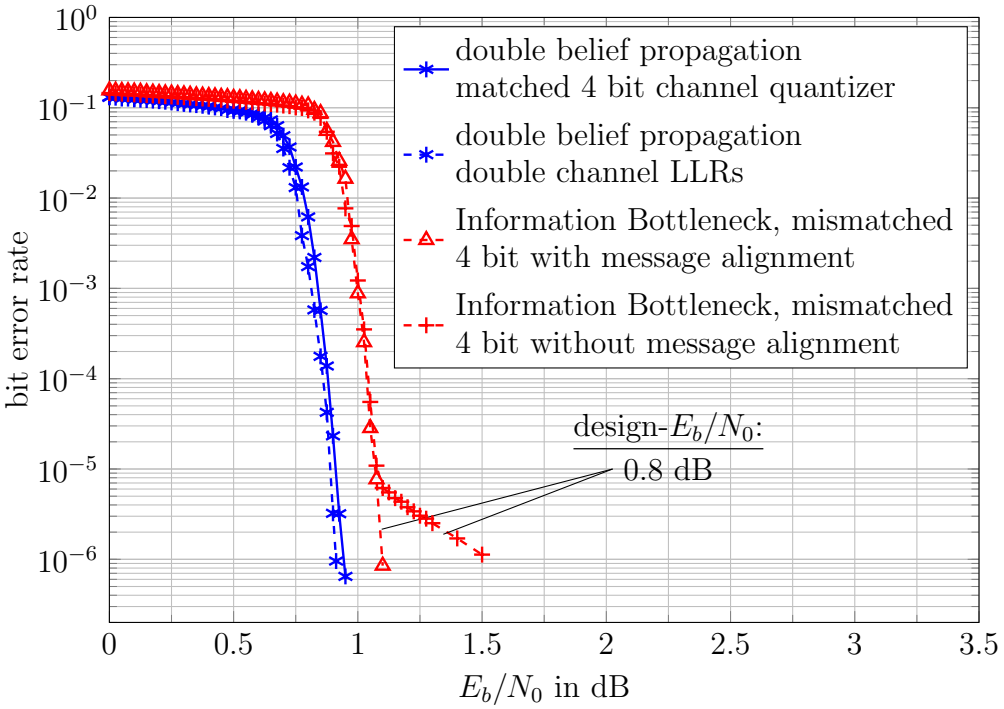
Figure 5.39: Analysis of *irregular code d)* with BPSK modulation under AWGN over E_b/N_0. Information Bottleneck decoders with 4 bit messages and $i_{\max} = 60$ iterations are applied.

0.1 dB over E_b/N_0. For the low rate $R = 0.25$ code *irregular code d)*, the respective gap of the best found Information Bottleneck decoder with design-$E_b/N_0 = 1.22$ dB is around 0.3 dB over E_b/N_0 at a bit error rate of 10^{-4}. It even increases a little bit for lower bit error rates. Obviously, the low code rate $R = 0.25$ of *irregular code d)* makes it more challenging to approach the performance of the double precision belief propagation decoder with a 4 bit Information Bottleneck decoder. However, it will be shown soon that a 5 bit Information Bottleneck decoder with message alignment can significantly reduce the remaining gap for the investigated low rate *irregular code d)*.

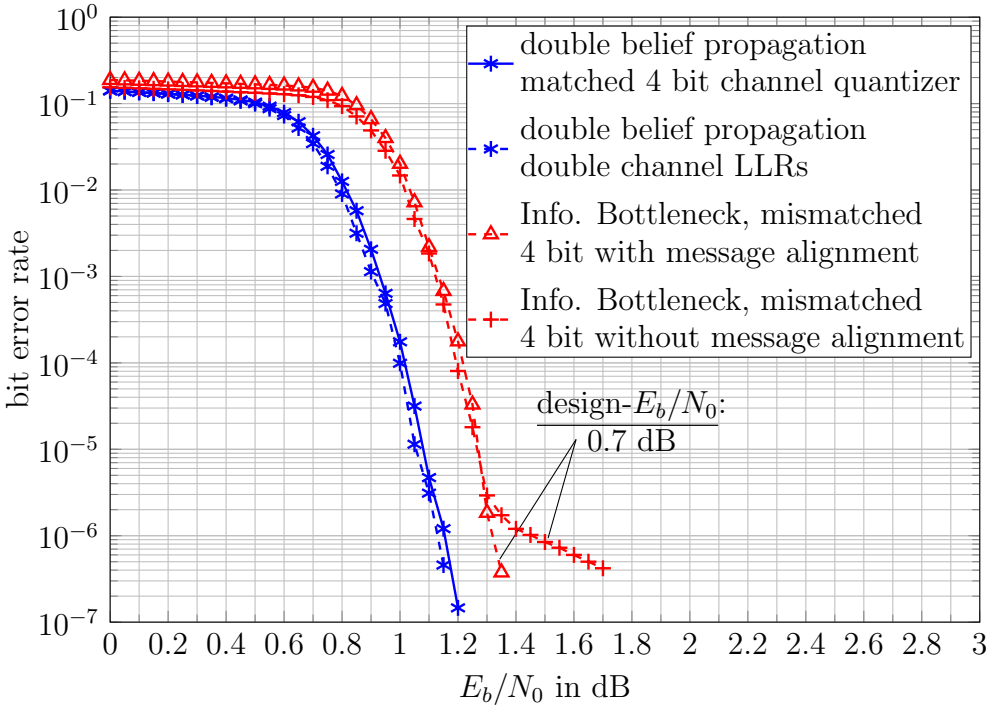
In summary, all investigated codes show that using the proposed decoder construction method from Section 5.4 which includes message alignment allows to construct Information Bottleneck decoders for irregular LDPC codes with 4 bit messages which approach to the performance of the double precision belief propagation decoding algorithm up to some small fraction of a decibel over E_b/N_0. The presented results also indicate that 4 bit Information Bottleneck decoders come closer to the performance of a double precision belief propagation performance for irregular LDPC codes with high code rates than for irregular codes with low code rates.

Performance Influence of Message Alignment

The previous investigation has shown that 4 bit Information Bottleneck decoders with message alignment can be constructed which offer performance very close to that of double precision belief propagation decoding. A question remaining is whether or not message alignment is a key technique to achieve this performance. To show that this is the case, Figures 5.40 and 5.41 investigate the bit error rate performances of 4 bit Information Bottleneck decoders with and without message alignment for *irregular codes a), b), c)* and *d)*. The number of decoder iterations was chosen identical to the previous investigation from Section 5.4.3 for each code and is mentioned again in the caption of each respective bit error rate plot. Figure 5.41b also includes curves for 5 bit Information Bottleneck decoders with and without message alignment, to illustrate that the observed performance gap of Information Bottleneck decoders with

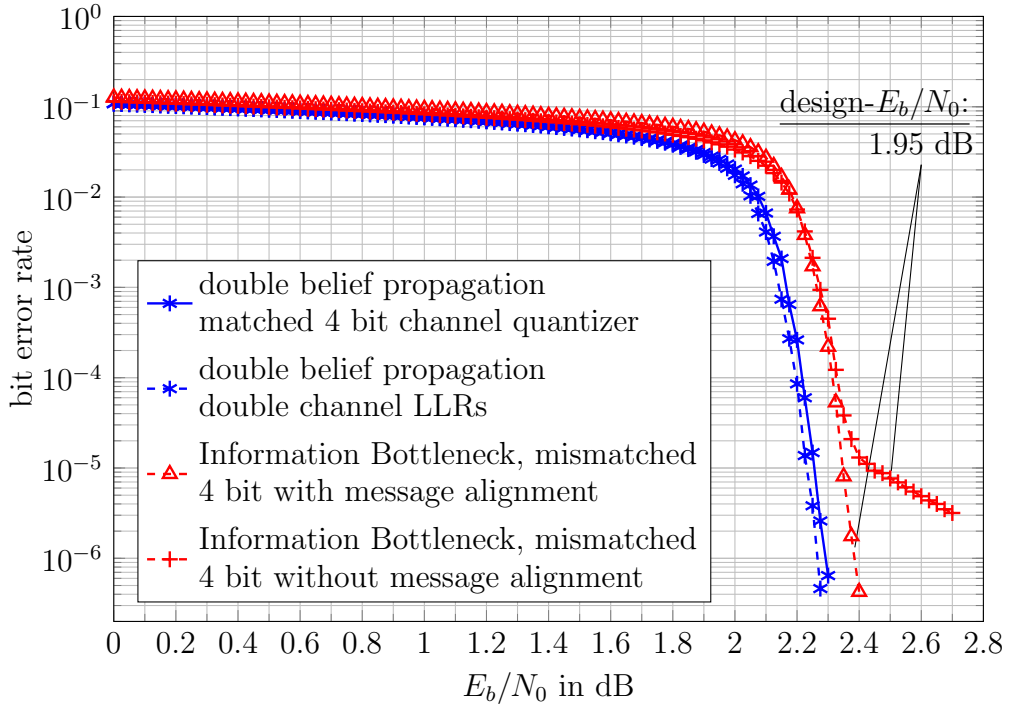


(a) Bit error rate of *irregular code a)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.

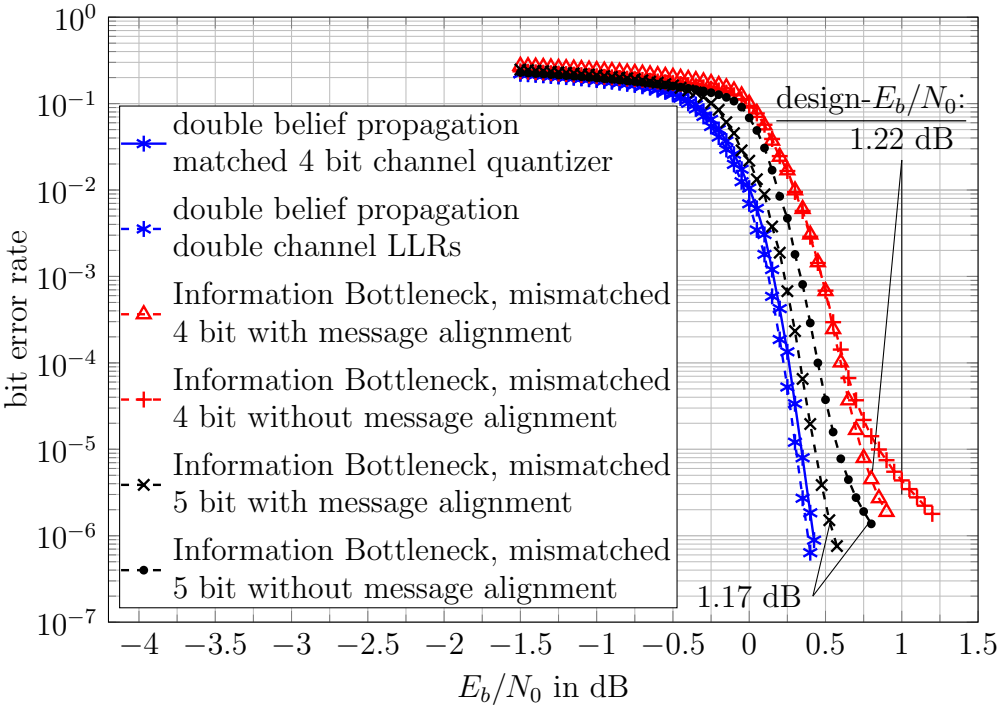


(b) Bit error rate of *irregular code b)* ($R = 0.5$) with $i_{\max} = 70$ decoding iterations.

Figure 5.40: Influence of message alignment on the bit error rates of Information Bottleneck decoders for *irregular codes a)* and *b)*.



(a) Bit error rate of *irregular code c)* ($R = 0.75$) with $i_{\max} = 40$ decoding iterations.



(b) Bit error rate of *irregular code d)* ($R = 0.25$) with $i_{\max} = 60$ decoding iterations.

Figure 5.41: Influence of message alignment on the bit error rates of Information Bottleneck decoders for *irregular codes c)* and *d)*.

respect to double precision belief propagation decoding for *irregular code d)* can be significantly reduced by a 5 bit Information Bottleneck decoder.

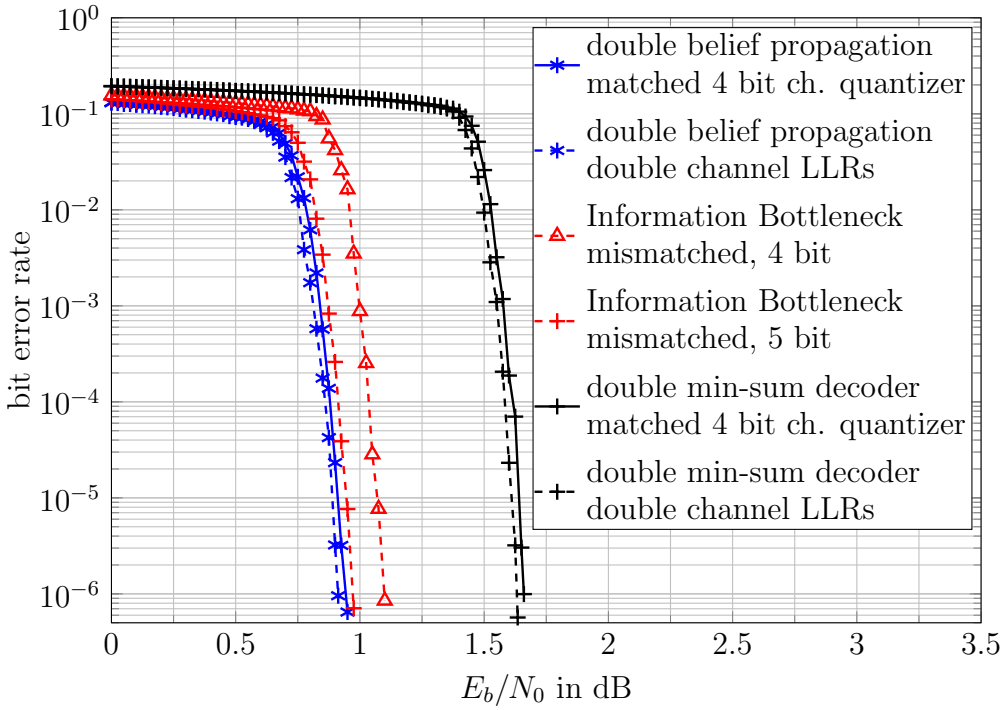
In the decoders without message alignment, the integers $t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}$ and $t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}$ were passed into the decoding graph directly, which corresponds to designing the decoder as it has been described in Section 5.4, but using only identity mappings $p\left(z_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}|t_{\mathrm{v}\to\mathrm{c}}^{(d_\mathrm{v},i)}\right)$ and $p\left(z_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}|t_{\mathrm{c}\to\mathrm{v}}^{(d_\mathrm{c},i)}\right)$ instead of adequately designed message alignment mappings.

The results for all four investigated codes show that the decoders without message alignment suffer from an early flattening of the bit error rate curves resulting in an early error floor. This error floor can be mitigated by using message alignment for all investigated codes and decoders. For *irregular code d)* and 5 bit Information Bottleneck decoding, a slight loss of not using message alignment can already be observed in the cliff region in addition to the premature flattening of the bit error rate curve without message alignment. As a result, message alignment can be considered to be a key technique for the design of Information Bottleneck LDPC decoders for irregular LDPC codes.

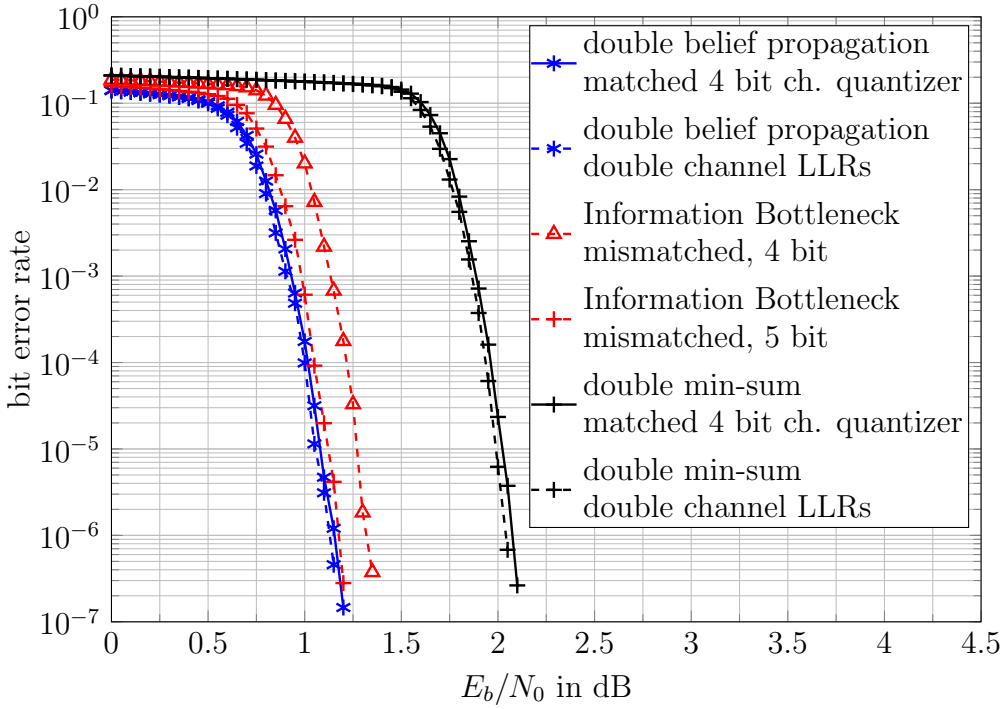
Comparison with Min-Sum Decoding

Figure 5.42 illustrates the bit error rate performances of the best found Information Bottleneck decoders with 4 and 5 bit messages for *irregular code a)* and *irregular code b)* with a maximum of $i_{\max} = 50$ and $i_{\max} = 70$ decoding iterations, respectively and compares them to the performances of min-sum decoders with and without channel output quantization. Figure 5.43 similarly illustrates the bit error rate performances of the best found Information Bottleneck decoders with 4 and 5 bit messages for *irregular code c)* and *irregular code d)* with a maximum of $i_{\max} = 40$ and $i_{\max} = 60$ decoding iterations. The included curves referring to double precision belief propagation shall serve as reference curves.

For *irregular codes a), b)* and *c)*, the min-sum decoder is strongly outperformed by the 4 bit Information Bottleneck decoders as it is revealed Figures 5.42a, 5.42b and 5.43a. The 5 bit Information Bottleneck decoders come enourmously close to the double belief propagation reference curves for these codes. For *irregular code a)* and a bit error rate of 10^{-5} the double precision min-sum decoder looses around 0.5 dB in comparison to the Information Bottleneck

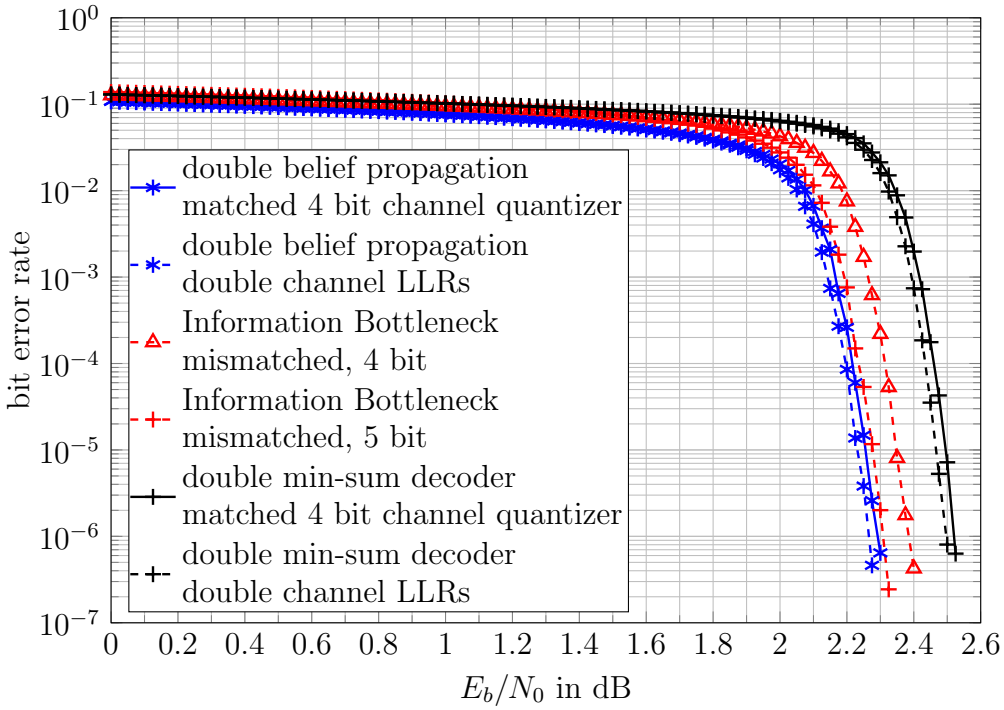


(a) Bit error rate of *irregular code a)* ($R = 0.5$) with $i_{\max} = 50$ decoding iterations.

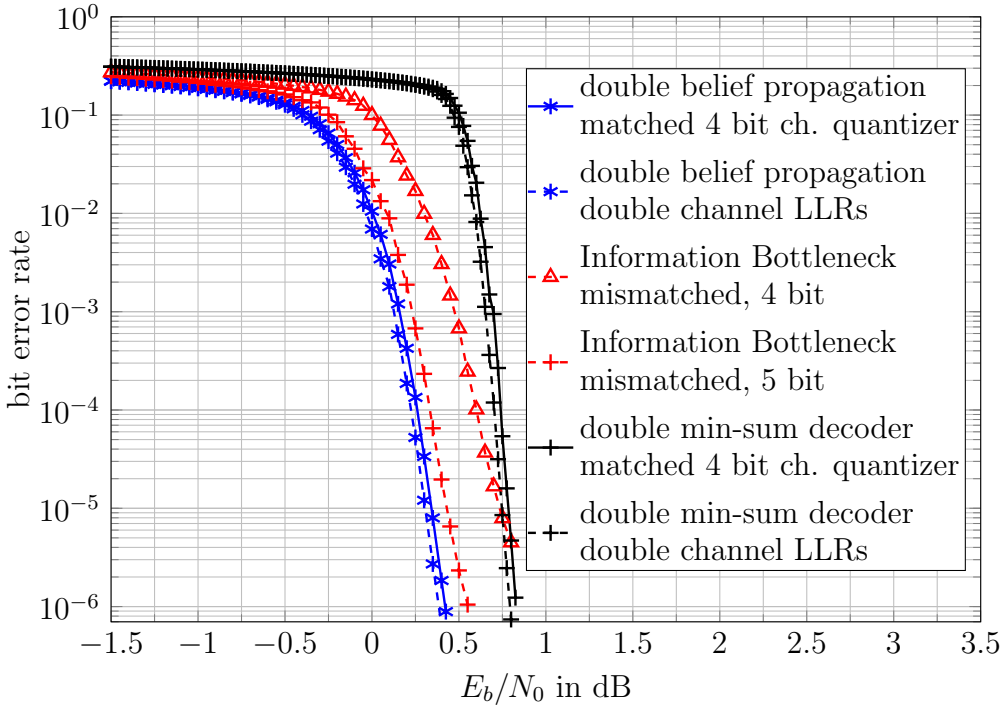


(b) Bit error rate of *irregular code b)* ($R = 0.5$) with $i_{\max} = 70$ decoding iterations.

Figure 5.42: Bit error rates of various decoders for *irregular code a)* and *irregular code b)* with BPSK modulation under AWGN over E_b/N_0.



(a) Bit error rate of *irregular code c)* ($R = 0.75$) with $i_{\max} = 40$ decoding iterations.



(b) Bit error rate of *irregular code d)* ($R = 0.25$) with $i_{\max} = 60$ decoding iterations.

Figure 5.43: Bit error rates of various decoders for *irregular code c)* and *irregular code d)* with BPSK modulation under AWGN over E_b/N_0.

decoder with 4 bit messages. The Information Bottleneck decoder with 5 bit messages approaches the decoding performance of a double precision belief propagation decoder up to some fractions of tenths of a decibel in the waterfall region of *irregular code a)*. The same effects can be observed for *irregular codes b)* and *c)*. For *irregular code b)*, the min-sum decoder with quantized channel output looses around 0.7 dB over E_b/N_0 in comparison to the 4 bit Information Bottleneck decoder at a bit error rate of $3 \cdot 10^{-5}$. For *irregular code c)*, the respective gap is around 0.15 dB over E_b/N_0 at a bit error rate of 10^{-5}.

Figure 5.43b shows that the best found 4 bit Information Bottleneck decoder for the low rate *irregular code d)* outperforms the min-sum decoder in the cliff region, but falls short at low bit error rates below 10^{-5}. Increasing the message bit width to 5 bit, however, brings the performance of the Information Bottleneck decoder close to the double precision reference curves also for the very low rate *irregular code d)*. The gap of the min-sum decoder with channel output quantization to the 5 bit Information Bottleneck decoder is around 0.25 dB at a bit error rate of $3 \cdot 10^{-6}$. It is larger for higher bit error rates in the cliff region.

5.4.4 Summary

In this section Information Bottleneck LDPC decoders for irregular codes have been constructed. For this purpose, a message alignment step was included in the decoder construction process and the decoding. This was done to overcome the impairment that integer messages from nodes with different degrees could represent different meanings for the corresponding codeword bits due to the independent design of the lookup tables for the different node degrees.

Including message alignment in the decoder construction enables to build Information Bottleneck LDPC decoders with 4 bit messages that have performance very close to double precision belief propagation decoding for the investigated irregular codes with code rates $R \geq 0.5$. The investigation of a code with $R = 0.25$ has shown that such low code rates seem to require 5 bit Information Bottleneck decoders to obtain performance very close to that of double precision belief propagation decoding.

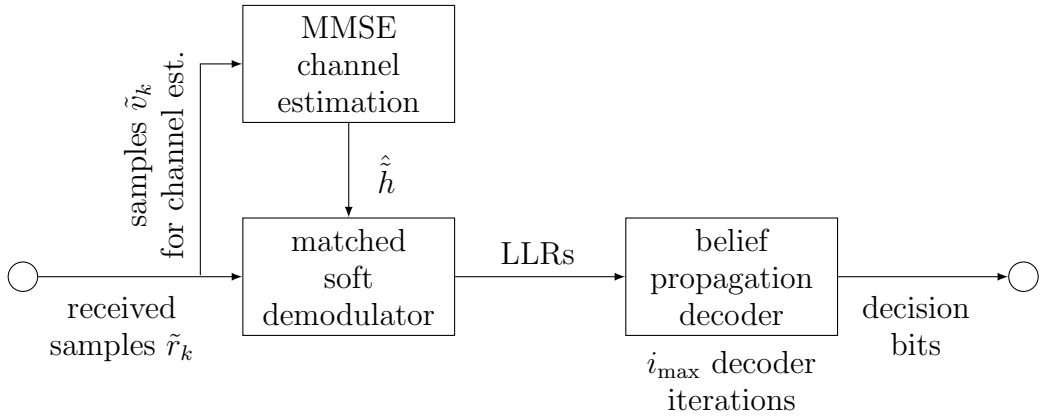
Similarly as it could be observed for regular LDPC codes in Section 5.2.5, the design-E_b/N_0 influences the error floor performance of the constructed LDPC decoders. While choosing it too low typically results in an early error floor, choosing it too large tends to harm the performance in the waterfall region. Not using message alignment also results in an early error floor. One can conclude that message alignment is a key technique for the construction of Information Bottleneck LDPC decoders for irregular LDPC codes.

5.5 A Simple Information Bottleneck Detection Scheme for Flat Fading Channels

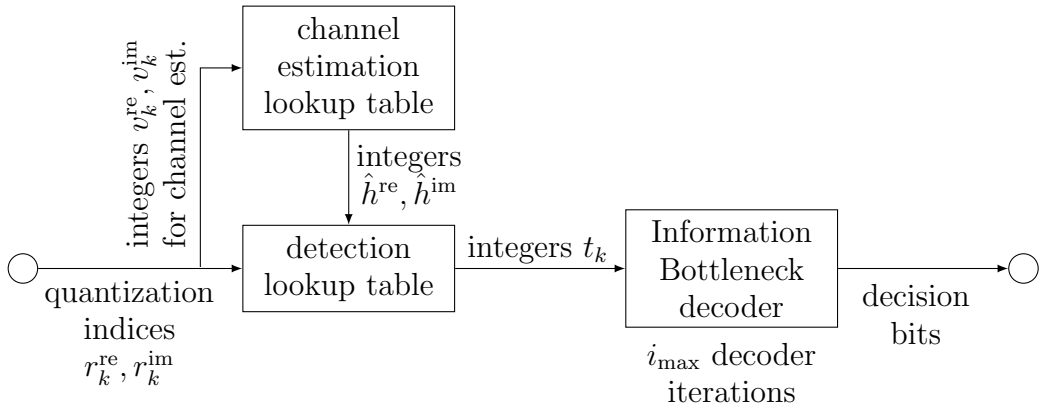
In many applications, the detection and decoding problem is more complex than for the AWGN channel model. Consider LDPC encoded data transmission over frequency flat block fading channel with block length B which has been introduced in Section 5.1.3. The considered transmitter encodes an LDPC code and uses BPSK modulation. The transmitter also inserts $P < B$ pilot symbols $s_0^{\mathrm{p}}, s_1^{\mathrm{p}}, \ldots, s_{P-1}^{\mathrm{p}}$ periodically into the modulated codeword with a distance of $B - P$ codeword symbols.

Coherent detection at the output of the fading channel additionally to correcting errors caused by the AWGN, requires to estimate and track the unknown channel coefficient $\tilde{h}_k$. The known pilot symbols enable the receiver to estimate the unknown channel coefficient. The obtained estimate can then be used to perform soft demodulation which provides LLRs to be processed by a conventional LDPC decoding algorithm. Figure 5.44a shows a conventional receiver chain which performs these signal processing steps. It uses minimum mean squared error (MMSE) channel estimation, soft demodulation and belief propagation decoding.

In the following, a simple coherent Information Bottleneck detection scheme for an LDPC encoded transmission over a frequency flat fading channel is presented and investigated. This scheme has been developed in the scope of this thesis and published in [63] and [76]. A schematic overview of the considered signal processing chain is shown in Figure 5.44b. Novel results included in this section show that Information Bottleneck receivers can be



(a) Conventional receiver chain.



(b) Information Bottleneck receiver chain.

Figure 5.44: Comparison of a conventional receiver chain and an Information Bottleneck receiver chain. In the latter, all signal processing steps required for channel estimation, detection and decoding are implemented using Information Bottleneck lookup tables.

constructed which even perform better than the ones investigated in [63] and [76].

The first signal processing step at the receiver is quantization of the received signal to obtain q bit quantization indices used for the further signal processing. The Information Bottleneck quantizers presented in Section 5.1.3 are used for this purpose. From this point, all signal processing steps required to make Information Bottleneck LDPC decoders applicable in the considered scenario are developed. An integer based Information Bottleneck equivalent to channel estimation will be the starting point of this process. Afterwards, a lookup

table based detection scheme will be presented to obtain integers that can be fed to an Information Bottleneck LDPC decoder. Hence, as depicted in Figure 5.44b, the aim is construction of a complete receiver-sided signal processing chain which includes channel estimation, detection and LDPC decoding, but only uses lookup operations in tables designed with the Information Bottleneck method. The design of all required signal processing blocks shown in Figure 5.44b will be explained in the following.

5.5.1 Information Bottleneck Channel Estimation

The transmitter inserts P pilot symbols into its transmit burst with a distance of $B-P$ symbols. Let $\tilde{v}_0, \tilde{v}_1, \ldots, \tilde{v}_{P-1}$ denote the *continuous* received samples corresponding to the inserted pilot symbols. Let us, moreover, consider a certain realization $\tilde{h}_k$ of the channel coefficient which is modeled constant for B symbol durations, such that the time index k can be dropped for the moment.

The considered conventional receiver uses the knowledge of the P pilot symbols s_k^{p} to calculate an MMSE channel estimate. This channel estimate is given by

$$\hat{\tilde{h}} = \frac{1}{P + \sigma_{\tilde{n}}^2/\sigma_{\tilde{h}}^2} \sum_{k=0}^{P-1} \tilde{v}_k s_k^{\mathrm{p}}. \tag{5.50}$$

It is obvious, that the evaluation of Equation (5.50) requires high precision arithmetic. In any digital receiver a high quantization resolution is required such that the quantized received samples adequately approximate the continuous $\tilde{v}_k$ to obtain a good estimate with high accuracy.

In the considered Information Bottleneck receiver, however, the aim is to directly process the q bit integer indices $v_k^{\mathrm{re}} = f_{\mathrm{Q}}\left(\tilde{v}_k^{\mathrm{re}}\right)$ and $v_k^{\mathrm{im}} = f_{\mathrm{Q}}\left(\tilde{v}_k^{\mathrm{im}}\right)$. These quantization indices are from the set $\{0, 1, \ldots, 2^q - 1\}$ and they do neither approximate the received samples, nor can they be used to obtain a channel estimate with an arithmetic rule like Equation (5.50).

Essentially, the task of channel estimation is to extract information on the unknown channel coefficient $\tilde{h}$ from P pairs of integers $\left(v_k^{\mathrm{re}}, v_k^{\mathrm{im}}\right)$. Please note that due to the considered channel model and the fact that the pilot symbols are BPSK symbols, it is possible to handle v_k^{re} and v_k^{im} independently. In the

following, the signal processing of v_k^{im} will not be discussed, as it is equivalent to the one of v_k^{re}.

Consider the quantized real part h^{re} of channel coefficient $\tilde{h} = \tilde{h}^{\mathrm{re}} + j\tilde{h}^{\mathrm{im}}$. The integer h^{re} is the index of the quantization region the continuous $\tilde{h}^{\mathrm{re}}$ falls into, that is, $h^{\mathrm{re}} = f_{\mathrm{Q}}\left(\tilde{h}^{\mathrm{re}}\right)$. The quantizer $f_{\mathrm{Q}}(.)$ is considered to be designed with the Information Bottleneck method, as discussed in Section 5.1.3. From an information theoretical perspective, the task of a channel estimator is hence to map the unsigned integers $v_0^{\mathrm{re}}, v_1^{\mathrm{re}}, \ldots, v_{P-1}^{\mathrm{re}}$ onto $\hat{h}^{\mathrm{re}}$, such that the mutual information $\mathrm{I}\left(\hat{\mathsf{H}}^{\mathrm{re}}; \mathsf{H}^{\mathrm{re}}\right)$ is maximized. Please note, however, that $\hat{h}^{\mathrm{re}}$ does not approximate h^{re} in general, as one might expect from the usual notion of channel estimation. Both realizations can even be from different sets with different cardinalities. The compressed $\hat{h}^{\mathrm{re}}$ is just an integer sharing a desired huge amount of mutual information with h^{re}. Realizations h^{re} and $\hat{h}^{\mathrm{re}}$ are solely linked by $p\left(h^{\mathrm{re}}|\hat{h}^{\mathrm{re}}\right)$.

An intuitive approach to design an Information Bottleneck equivalent to channel estimation would be to design an Information Bottleneck compression mapping $p\left(\hat{h}^{\mathrm{re}}|v_0^{\mathrm{re}}, v_1^{\mathrm{re}}, \ldots, v_{P-1}^{\mathrm{re}}\right)$ which aims to maximize $\mathrm{I}\left(\hat{\mathsf{H}}^{\mathrm{re}}; \mathsf{H}^{\mathrm{re}}\right)$. However, implementing this node would typically result in prohibitive complexity, since the number of possible input configurations is $|\mathcal{R}^{\mathrm{re}}|^P$. For example, with 5 bit quantization and $P = 6$ pilot symbols, this would result in a lookup table with more than 10^9 entries. As a result, the node needs to be opened.

Figure 5.45 shows an opened node structure in an Information Bottleneck graph which makes implementing the node $p\left(\hat{h}^{\mathrm{re}}|v_0^{\mathrm{re}}, v_1^{\mathrm{re}}, \ldots, v_{P-1}^{\mathrm{re}}\right)$ practically feasible. Obviously, a strong similarity to the check and variable node operations in Information Bottleneck LDPC decoders exists. Just as it has been done for the implementation of the node operations in LDPC decoders, the lookup of $\hat{h}^{\mathrm{re}}$ based on P inputs is split into $(P-1)$ two-input operations, all designed with the Information Bottleneck method.

All two-input lookup tables appearing inside the opened node structure shall preserve information on the same variable H^{re}. This makes the Information Bottleneck graph shown in Figure 5.45 very similar to the one from Figure 5.18 which describes the variable node operation in an Information Bottleneck LDPC decoder. In fact, more similarities between both Information Bottleneck graphs can be identified, when considering how the joint distributions required

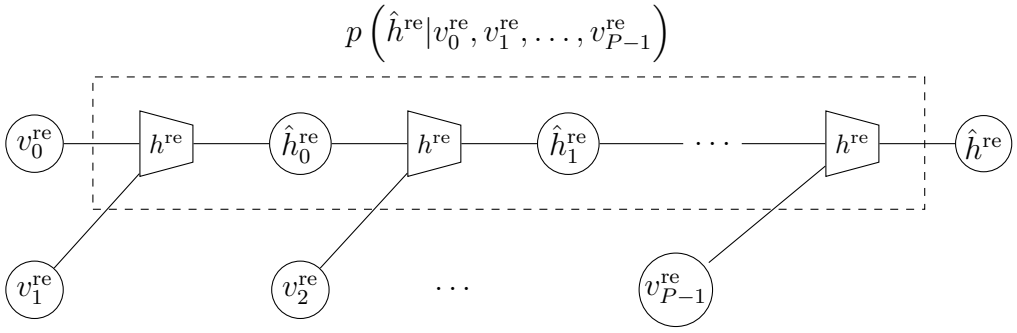


Figure 5.45: Information Bottleneck graph of a channel estimation lookup table with opened node $p\left(\hat{h}^{\mathrm{re}}|v_0^{\mathrm{re}}, v_1^{\mathrm{re}}, \ldots, v_{P-1}^{\mathrm{re}}\right)$. The output of the node aims to preserve information on the quantized real part of the unknown channel coefficient.

to design the nodes $p\left(\hat{h}_0^{\mathrm{re}}|v_0^{\mathrm{re}}, v_1^{\mathrm{re}}\right)$ and $p\left(\hat{h}_m^{\mathrm{re}}|\hat{h}_{m-1}^{\mathrm{re}}, v_{m+1}^{\mathrm{re}}\right)$ for $m \geq 1$ with the Information Bottleneck method are obtained. These distributions are given by

$$p\left(h^{\mathrm{re}}, v_0^{\mathrm{re}}, v_1^{\mathrm{re}}|s_0^{\mathrm{p}}, s_1^{\mathrm{p}}\right) = p\left(h^{\mathrm{re}}, v_0^{\mathrm{re}}|s_0^{\mathrm{p}}\right) p\left(h^{\mathrm{re}}, v_1^{\mathrm{re}}|s_1^{\mathrm{p}}\right) \frac{1}{p\left(h^{\mathrm{re}}\right)} \tag{5.51}$$

and

$$p\left(h^{\mathrm{re}}, \hat{h}_{m-1}^{\mathrm{re}}, v_{m+1}^{\mathrm{re}}|s_0^{\mathrm{p}}, s_1^{\mathrm{p}}, \ldots, s_{m+1}^{\mathrm{p}}\right) = p\left(h^{\mathrm{re}}, \hat{h}_{m-1}^{\mathrm{re}}|s_0^{\mathrm{p}}, s_1^{\mathrm{p}}, \ldots, s_m^{\mathrm{p}}\right) p\left(h^{\mathrm{re}}, v_{m+1}^{\mathrm{re}}|s_{m+1}^{\mathrm{p}}\right) \frac{1}{p\left(h^{\mathrm{re}}\right)} \text{ for } m \geq 1. \tag{5.52}$$

It is not strictly necessary to mention the condition on $s_0^{\mathrm{p}}, s_1^{\mathrm{p}}, \ldots, s_{m+1}^{\mathrm{p}}$ explicitly, as these pilot symbols are modeled to not be random. However, in this notation it becomes clear that the joint distributions involved in the design and hence also the resulting lookup tables depend on the particular choice of pilot symbols.

The Equations (5.51) and (5.52) obviously compare to Equations (5.29) and (5.30) which have been used for the design of the variable node operations in an Information Bottleneck LDPC decoder as they have some very similar structure. This is not a coincidence, but can be explained as follows. In an LDPC decoder, the variable nodes impose an equality constraint on the bit represented by each incoming message, that is, the incoming messages are modeled to be independent given the codeword bit and all incoming messages represent the same codeword bit. Similarly, in the considered Information

Bottleneck graph for channel estimation, the incoming quantization indices v_k^{re} are modeled to be independent given h^{re} and s_k^{p} and all provide information on the same h^{re} which is modeled constant over a block of B symbol durations. As a result, we again deal with an equality constraint.

An open question still is how to obtain the joint distributions $p\left(h^{\mathrm{re}}, v_m^{\mathrm{re}} | s_m^{\mathrm{p}}\right)$ needed to evaluate Equations (5.51) and (5.52). Recall that h^{re} and v_m^{re} are quantization indices of $\tilde{h}^{\mathrm{re}}$ and $\tilde{v}_m^{\mathrm{re}}$, respectively. Hence, $h^{\mathrm{re}} = f_{\mathrm{Q}}\left(\tilde{h}^{\mathrm{re}}\right)$ and $v_m^{\mathrm{re}} = f_{\mathrm{Q}}\left(\tilde{v}_m^{\mathrm{re}}\right)$, where $f_{\mathrm{Q}}(.)$ characterizes the threshold decisions of an Information Bottleneck quantizer from Section 5.1.3. The quantizer $f_{\mathrm{Q}}(.)$ for $\tilde{h}^{\mathrm{re}}$ and $\tilde{v}^{\mathrm{re}}$ can also be described by $p\left(h^{\mathrm{re}} | \tilde{h}^{\mathrm{re}}\right)$ and $p\left(v_m^{\mathrm{re}} | \tilde{v}_m^{\mathrm{re}}\right)$. This allows to express the joint distribution $p\left(h^{\mathrm{re}}, v_m^{\mathrm{re}} | s_m^{\mathrm{p}}\right)$ as

$$p\left(h^{\mathrm{re}}, v_m^{\mathrm{re}} |\, s_m^{\mathrm{p}}\right) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p\left(\tilde{h}^{\mathrm{re}}, \tilde{v}_m^{\mathrm{re}} | s_m^{\mathrm{p}}\right) p\left(h^{\mathrm{re}} | \tilde{h}^{\mathrm{re}}\right) p\left(v_m^{\mathrm{re}} | \tilde{v}_m^{\mathrm{re}}\right) \,\mathrm{d}\,\tilde{h}^{\mathrm{re}}\,\mathrm{d}\,\tilde{v}_m^{\mathrm{re}}. \quad (5.53)$$

In this integral, the factors $p\left(h^{\mathrm{re}} | \tilde{h}^{\mathrm{re}}\right)$ and $p\left(v_m^{\mathrm{re}} | \tilde{v}_m^{\mathrm{re}}\right)$ gather all probability mass of the continuous $\tilde{h}^{\mathrm{re}}$ and $\tilde{v}_m^{\mathrm{re}}$ which are mapped onto the same pair $(h^{\mathrm{re}}, v_m^{\mathrm{re}})$ by application of the quantizer on these variables. Therefore, they determine the integration area for a particular pair $(h^{\mathrm{re}}, v_m^{\mathrm{re}})$. The distribution $p\left(\tilde{h}^{\mathrm{re}}, \tilde{v}_m^{\mathrm{re}} | s_m^{\mathrm{p}}\right)$ for the considered channel model is a multivariate Gaussian distribution. Both components have zero mean. As a result, the multivariate Gaussian distribution $p\left(\tilde{h}^{\mathrm{re}}, \tilde{v}_m^{\mathrm{re}}, | s_m^{\mathrm{p}}\right)$ is fully characterized by the covariance matrix

$$\mathbf{C}_{\tilde{\mathsf{H}}^{\mathrm{re}}, \tilde{\mathsf{V}}_m^{\mathrm{re}} | \mathsf{S}_m^{\mathrm{p}}} = \frac{\sigma_{\tilde{h}}^2}{2} \begin{bmatrix} 1 & s_m^{\mathrm{p}} \\ s_m^{\mathrm{p}} & 1 \end{bmatrix} + \frac{\sigma_{\tilde{n}}^2}{2} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}. \quad (5.54)$$

For a given covariance matrix and integration limits determined by a particular quantizer, [77] describes a very efficient method to numerically solve the integrals over probability distributions of multivariate Gaussian random variables of the form from Equation (5.53). Application of this method allows to obtain the joint distributions $p\left(h^{\mathrm{re}}, v_m^{\mathrm{re}} |\, s_m^{\mathrm{p}}\right)$ and, hence, to construct the Information Bottleneck nodes from Figure 5.45 using Equations (5.51) and (5.52) with any Information Bottleneck algorithm. In this thesis, the KL-means algorithm is applied to construct the proposed Information Bottleneck channel estimator.

This choice was motivated by the inherent possibility for parallelization of this algorithm described in Section 3.2.3 which results in runtime benefits.

The design process of the Information Bottleneck channel estimator delivers the joint distribution $p\left(\hat{h}^{\mathrm{re}}, h^{\mathrm{re}}\right) = p\left(h^{\mathrm{re}}|\hat{h}^{\mathrm{re}}\right) p\left(\hat{h}^{\mathrm{re}}\right)$ which has to be used further, to construct an Information Bottleneck detection lookup table. The output cardinalities of the Information Bottleneck algorithm used to design the Information Bottleneck nodes in Figure 5.45 determine the bit widths required to store the intermediate results $\hat{h}_m^{\mathrm{re}}$ and the final result $\hat{h}^{\mathrm{re}}$. For simplicity, all Information Bottleneck algorithms applied to node design use the same output bit width q^{ce}, such that all intermediate results $\hat{h}_m^{\mathrm{re}}$ and the final output $\hat{h}^{\mathrm{re}}$ are from the same set $\left\{0, 1, \ldots, 2^{q^{\mathrm{ce}}} - 1\right\}$. However, again this is a design choice which leaves a lot of freedom for the design of the opened node structure.

Finally, please note that due to the identical statistical properties of h^{re} and h^{im}, the obtained Information Bottleneck channel estimator can also be used to obtain $\hat{h}^{\mathrm{im}}$.

5.5.2 Information Bottleneck Detection and Decoding

In a conventional receiver with high precision, the obtained MMSE channel estimate $\hat{\tilde{h}}$ from Equation (5.50) can be used for soft demodulation. For this purpose, a so called *matched* soft demodulator as proposed in [78] calculates LLRs

$$L^{\mathrm{ch}}(c_k) = \log \frac{p\left(\tilde{r}_k | \mathsf{C}_k = 0, \hat{\tilde{h}}\right)}{p\left(\tilde{r}_k | \mathsf{C}_k = 1, \hat{\tilde{h}}\right)}. \tag{5.55}$$

The demodulator is called *matched* because it already takes into account, that the used MMSE channel estimate $\hat{\tilde{h}}$ is not a perfect estimate, but instead results from P known pilot symbols. Following the derivation in [78], the LLRs of a matched soft demodulator are given by

$$L^{\mathrm{ch}}(c_k) = \frac{1}{1 + \frac{1}{P + \sigma_{\tilde{n}}^2 / \sigma_{\tilde{h}}^2}} \cdot \frac{4 \operatorname{Re}\left\{\hat{\tilde{h}}^* \tilde{r}_k\right\}}{\sigma_{\tilde{n}}^2}. \tag{5.56}$$

Using these channel LLRs, it is easy to apply a conventional LLR based decoding algorithm.

When the Information Bottleneck channel estimator from the preceding section is applied, it delivers the pair $\left(\hat{h}^{\mathrm{re}}, \hat{h}^{\mathrm{im}}\right)$ of unsigned integers. Moreover,

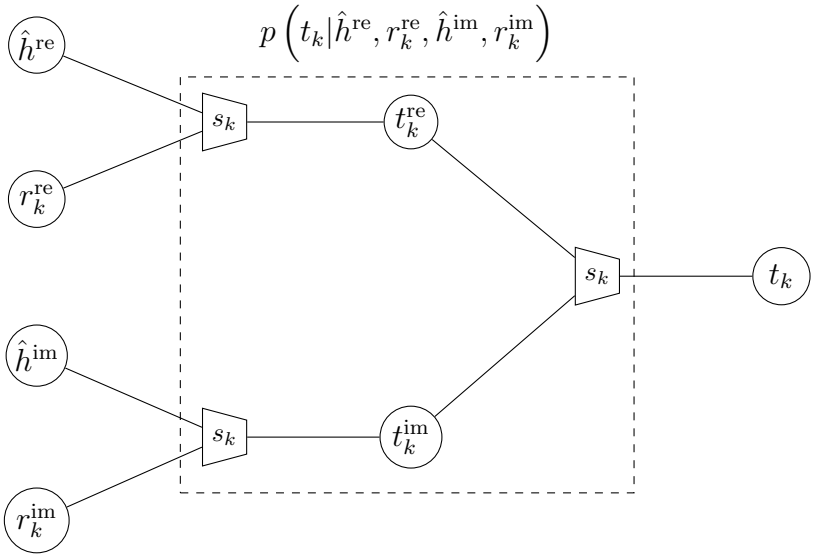


Figure 5.46: Information Bottleneck graph of a coherent BPSK detector for a frequency flat fading channel.

the quantizer delivers a pair $\left(r_k^{\mathrm{re}}, r_k^{\mathrm{im}}\right)$ for each received symbol s_k. The task of an Information Bottleneck demodulator equivalent for the considered channel is extracting relevant information on s_k from $\left(\hat{h}^{\mathrm{re}}, \hat{h}^{\mathrm{im}}\right)$ and $\left(r_k^{\mathrm{re}}, r_k^{\mathrm{im}}\right)$. Figure 5.46 shows an Information Bottleneck graph performing this task. The shown opened node $p\left(t_k|\hat{h}^{\mathrm{re}}, r_k^{\mathrm{re}}, \hat{h}^{\mathrm{im}}, r_k^{\mathrm{im}}\right)$ first aims to successively combine $\hat{h}^{\mathrm{re}}, r_k^{\mathrm{re}}$ and $\hat{h}^{\mathrm{im}}, r_k^{\mathrm{im}}$ to obtain t_k^{re} and t_k^{im}. The lookup tables $p\left(t_k^{\mathrm{re}}|\hat{h}^{\mathrm{re}}, r_k^{\mathrm{re}}\right)$ and $p\left(t_k^{\mathrm{im}}|\hat{h}^{\mathrm{im}}, r_k^{\mathrm{im}}\right)$ are identical, as their inputs are identically distributed. To design $p\left(t_k^{\mathrm{re}}|\hat{h}^{\mathrm{re}}, r_k^{\mathrm{re}}\right)$ with an Information Bottleneck algorithm, one needs the joint probability distribution

$$p\left(s_k, \hat{h}^{\mathrm{re}}, r_k^{\mathrm{re}}\right) = \sum_{h^{\mathrm{re}}=0}^{2^q-1} p\left(h^{\mathrm{re}}, r_k^{\mathrm{re}}|s_k\right) \underbrace{\frac{p\left(h^{\mathrm{re}}, \hat{h}^{\mathrm{re}}\right)}{p\left(h^{\mathrm{re}}\right)}}_{p\left(\hat{h}^{\mathrm{re}}|h^{\mathrm{re}}\right)} p(s_k). \tag{5.57}$$

In Equation (5.57), the distributions $p\left(h^{\mathrm{re}}, \hat{h}^{\mathrm{re}}\right)$ and $p\left(h^{\mathrm{re}}\right)$ are known from the design of the channel estimation lookup table. The distribution $p\left(h^{\mathrm{re}}, r_k^{\mathrm{re}}|s_k\right)$ again is an integral of a multivariate Gaussian distribution, that is,

$$p\left(h^{\mathrm{re}}, r_k^{\mathrm{re}}|\, s_k\right) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p\left(\tilde{h}^{\mathrm{re}}, \tilde{r}_k^{\mathrm{re}}|s_k\right) p\left(h^{\mathrm{re}}|\tilde{h}^{\mathrm{re}}\right) p\left(r_k^{\mathrm{re}}|\tilde{r}_k^{\mathrm{re}}\right) \,\mathrm{d}\,\tilde{h}^{\mathrm{re}}\,\mathrm{d}\,\tilde{r}_k^{\mathrm{re}}. \tag{5.58}$$

The integral again can be solved using the method from [77]. The required covariance matrix is obtained analogously to Equation (5.54).

The node $p\left(t_k|t_k^{\mathrm{re}}, t_k^{\mathrm{im}}\right)$ shown in Figure 5.46 combines the information on s_k from t_k^{re} and t_k^{im} in a single output t_k. Taking the independence of t_k^{re} and t_k^{im} into account, the joint distribution for its design is given by

$$p\left(s_k, t_k^{\mathrm{re}}, t_k^{\mathrm{im}}\right) = p\left(s_k, t_k^{\mathrm{re}}\right) p\left(s_k, t_k^{\mathrm{im}}\right) \frac{1}{p\left(s_k\right)}. \tag{5.59}$$

Again, the joint distributions $p\left(s_k, t_k^{\mathrm{re}}\right)$ and $p\left(s_k, t_k^{\mathrm{im}}\right)$ are known from the Information Bottleneck algorithm applied to design the ancestor nodes $p\left(t_k^{\mathrm{re}}|\hat{h}^{\mathrm{re}}, r_k^{\mathrm{re}}\right)$ and $p\left(t_k^{\mathrm{im}}|\hat{h}^{\mathrm{im}}, r_k^{\mathrm{im}}\right)$.

In this thesis, the KL-means algorithm is also used to design detection lookup tables due to its short runtime. For simplicity, all Information Bottleneck nodes appearing in the opened node from Figure 5.46 use the same output bit width q^{det}, such that $t_k \in \left\{0, 1, \ldots, 2^{q^{\mathrm{det}}} - 1\right\}$. The integers t_k can directly be used as input integers for an Information Bottleneck LDPC decoder. As a result, they form the incoming messages $y_n^{(0)}$ of the check nodes in the first decoding iteration. The design of the node $p\left(t_k|t_k^{\mathrm{re}}, t_k^{\mathrm{im}}\right)$ delivers $p\left(s_k, t_k\right)$ as a side product. This distribution steps into the position of $p\left(b_n, y_n^{(0)}\right)$ in the design of the Information Bottleneck decoder. The BPSK symbol s_k has to be relabeled to b_n and t_k to $y_n^{(0)}$. Since $b_n \in \{0, 1\}$ and $s_k \in \{-1, +1\}$, this step involves the reversion of the bit-to-symbol mapping of the applied BPSK modulation. The design of the Information Bottleneck decoder then is equivalent to Section 5.2.

The knowledge of the joint distribution $p\left(s_k, t_k\right)$ from the design of the node $p\left(t_k|t_k^{\mathrm{re}}, t_k^{\mathrm{im}}\right)$ again allows to pair a conventional LLR based LDPC decoder with the Information Bottleneck detection scheme from Figure 5.46 because it allows to calculate channel LLRs from the integers t_k. This again enables a fair comparison which solely focusses on the difference between the conventional LDPC decoders and the Information Bottleneck decoders. However, for the LLR based decoders, all lookup tables have to be matched to the E_b/N_0 in order to provide meaningful LLRs.

5.5.3 Performance and Discussion

This section investigates the bit error rate performance of the proposed Information Bottleneck detection and decoding scheme for a frequency flat block fading channel. The investigated scenario has the following parameters:

1. The channel coefficients $\tilde{h}_k$ of the block fading channel are constant for $B = 38$ symbol durations.

2. $P = 6$ pilot symbols are inserted into the BPSK modulated codeword periodically with a distance of $B - P = 32$ codeword symbols.

3. The applied LDPC code is the length $8,000$ $(3, 6)$ regular LDPC code 8000.4000.3.483 from [68] (*regular code a)*) with code rate $R = 0.5$. The maximum number of decoder iterations is $i_{\max} = 50$ for all applied decoders.

These parameters were chosen for exemplary illustration of the proposed receiver design method. The chosen values allow to analyze the performance and the interaction of all developed Information Bottleneck signal processing components, but can be adapted easily.

Figure 5.47 shows the bit error rate performances of several detection and decoding schemes in the considered scenario. A non-quantized conventional system with belief propagation decoding and MMSE channel estimation as shown in Figure 5.44a serves as a performance bound for the quantized receivers (dashed blue curve, $*$ markers).

In order to investigate the performance of the proposed Information Bottleneck channel estimation and detection scheme without the influence of the Information Bottleneck decoder, a curve which pairs the Information Bottleneck channel estimation and detection with belief propagation decoding is included. In the corresponding receiver, a $q = 6$ bit quantizer, $q^{\text{ce}} = 8$ bit channel estimation lookup tables, and $q^{\text{det}} = 5$ bit detection lookup tables which were all designed with the Information Bottleneck method have been used. The corresponding bit error rate curve (solid blue curve, $*$ markers) is labeled with the respective bit widths $(q, q^{\text{ce}}, q^{\text{det}}) = (6, 8, 5)$. It shows that the loss in comparison to the non-quantized system with belief propagation decoding is about 0.1 dB over E_b/N_0 in the waterfall region of the code. This

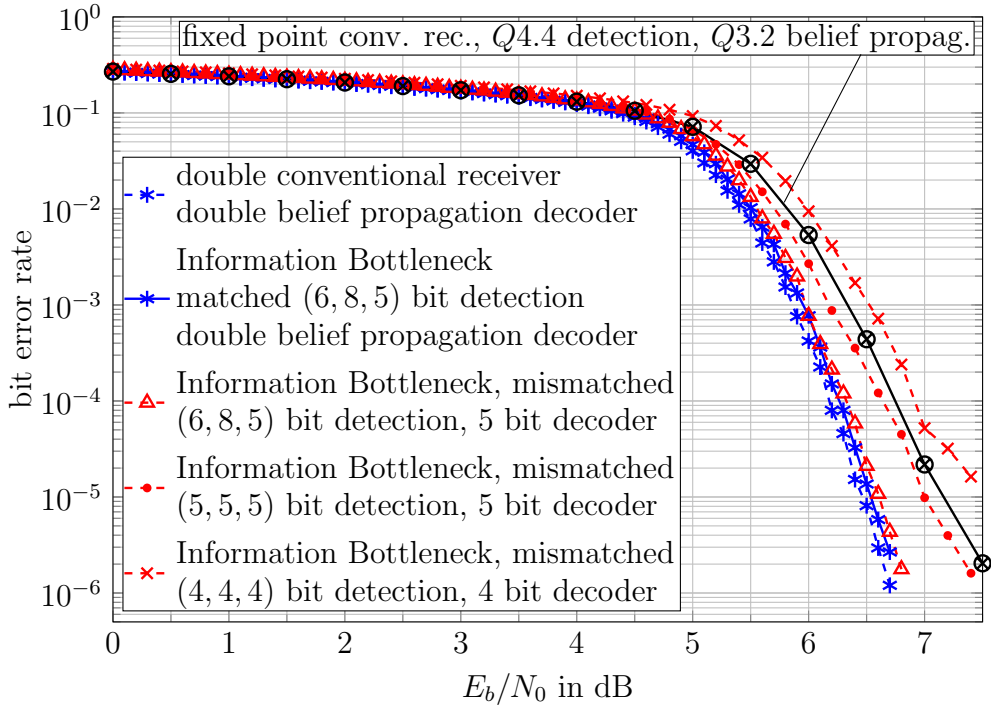


Figure 5.47: Bit error rates of various detection and decoding schemes for *regular code a)* with BPSK modulation on a block fading channel over E_b/N_0.

is particularly interesting because the entire channel estimation and detection stages of this receiver only use lookup operations and are quantized. However, the receiver requires to match the lookup tables used for channel estimation and detection to the E_b/N_0 on the channel, to provide meaningful LLRs to the belief propagation decoder.

The red curve with $\triangle$ markers in Figure 5.47 shows the bit error rate of the proposed fully lookup table based Information Bottleneck receiver chain which implements channel estimation, detection and LDPC decoding only using lookup tables. The bit widths in the detection and channel estimation stages were chosen as $(q, q^{\mathrm{ce}}, q^{\mathrm{det}}) = (6, 8, 5)$ again. The applied Information Bottleneck LDPC decoder used $q^{\mathrm{dec}} = q^{\mathrm{det}} = 5$ bit messages. All components inside this receiver were used mismatched. The design-E_b/N_0 chosen in this setting was 5.09 dB and found by hand optimization. It shall be mentioned that due to the very large number of Information Bottleneck algorithms involved in the receiver design, it was observed that multiple runs of the receiver construction with the same design-E_b/N_0 sometimes resulted in different receivers with

different performances, depending on the initial seed of the random number generator. This is caused by the fact that each applied Information Bottleneck algorithm starts from some random initial clustering and could find different lookup tables, with slightly different amounts of preserved relevant information, depending on this initial clustering. Therefore, the receiver construction was run multiple times and also each Information Bottleneck algorithm was run with many different initial clusterings. Once an entire receiver had been constructed, the analysis of the decoder output information $\mathrm{I}\left(\mathsf{X};\mathsf{T}_{\mathrm{dec}}^{(i)}\right)$ described in Section 5.2.4 was used to identify potentially promising receivers. The shown results refer to the best found receiver from this method.

The loss of this quantized and mismatched receiver is just about 0.2 dB over E_b/N_0 in comparison to the non-quantized receiver with double precision and smaller than 0.1 dB in comparison to the reference system with Information Bottleneck detection and channel estimation, LLR based belief propagation decoding and matched lookup tables. It needs to be emphasized that the receivers with belief propagation decoding had perfect knowledge of the channel noise variance $\sigma_{\tilde{n}}^2$ to calculate LLRs for the belief propagation decoders. The Information Bottleneck receiver, in contrast, did not require any knowledge of the channel noise variance because all its lookup tables were being used mismatched to the actual E_b/N_0.

It was found that it is important to use bit widths of at least 6 and 8 bits for the quantizer and the channel estimation scheme to achieve performance this close to the conventional receiver. To illustrate this fact, the red curve with dot markers shows the performance of the Information Bottleneck receiver with all components using $(q, q^{\mathrm{ce}}, q^{\mathrm{det}}) = (5, 5, 5)$ bit quantization and an Information Bottleneck decoder with $q^{\mathrm{dec}} = q^{\mathrm{det}} = 5$ bit messages. The design-E_b/N_0 chosen in this setting was 5.22 dB. The loss of this quantized and mismatched 5 bit receiver is about 0.5 dB over E_b/N_0 in comparison to the non-quantized receiver with double precision at a bit error rate of 10^{-5}. The dashed red curve with $\times$ markers illustrates the performance of the Information Bottleneck receiver with all components using 4 bit integers. Its design-E_b/N_0 was 5.3 dB. As expected, the four bit Information Bottleneck receiver suffers from a greater loss over E_b/N_0 with respect to the double precision reference than the five bit Information Bottleneck receiver.

Finally, also a curve showing the performance of a receiver with fixed point precision is included in Figure 5.47 (solid black curve, $\otimes$ markers). This receiver used 8 bit fixed point arithmetic in the detector and the channel estimation processing and 5 bit fixed point messages in the belief propagation decoder. The $Qa.b$ notation is used here as in [79] to describe the formats of used fixed point numbers. It expresses, how many bits are allocated to the integer and the fractional part of a signed fixed point number. The range of a signed $Qa.b$ fixed point number is from $-2^{(a-1)}$ to $+2^{(a-1)} - 2^{-b}$ with a uniform precision of 2^{-b}. The considered receiver used the 8 bit $Q4.4$ number format in the detection and channel estimation stages and the five bit $Q3.2$ fixed point format to represent the exchanged LLRs in the LDPC decoder. For a fair comparison, also all other combinations of $Qa.b$ number formats with these bit widths in the detector and the channel decoder were tested, but this choice offered the best performance in this scenario. The results show that the conventional fixed point receiver is outperformed by the Information Bottleneck receiver with $(6, 8, 5)$ bit detection and 5 bit decoding and also by the Information Bottleneck receiver with 5 bit processing in all signal processing blocks.

An open question still is, whether it is practical to store the lookup tables required to implement the Information Bottleneck receiver or not. Table 5.6 lists the memory requirements of the proposed Information Bottleneck receivers with the chosen parameters. The formulas to calculate the memory demands of the respective components and their derivation can be found in Appendix A.7.1. The memory requirements of the best found receiver structure with performance comparable to that ouf double precision processing lies in the order of several hundred kilobytes which is no challenge to handle in practice. The table also illustrates that the Information Bottleneck design approach allows to flexibly construct smaller receiver components by reducing the bit widths of the components involved, if required. However, such a reduction comes at the expense of a performance degradation.

5.5.4 Summary

In this section a complete Information Bottleneck based coherent detection scheme for a frequency flat fading channel has been developed, investigated

Table 5.6: Memory demands of receiver components

$(6, 8, 5)$ bit detection, 5 bit decoder:

Component	Output bit width	Memory requirements
channel estimation	$q^{\mathrm{ce}} = 8$	69.63 kilobytes
detection	$q^{\mathrm{det}} = 5$	10.88 kilobytes
LDPC decoder	$q^{\mathrm{dec}} = 5$	224.0 kilobytes
total		304.51 kilobytes

$(5, 5, 5)$ bit detection, 5 bit decoder:

Component	Output bit width	Memory requirements
channel estimation	$q^{\mathrm{ce}} = 5$	3.2 kilobytes
detection	$q^{\mathrm{det}} = 5$	1.28 kilobytes
LDPC decoder	$q^{\mathrm{dec}} = 5$	224.0 kilobytes
total		228.48 kilobytes

$(4, 4, 4)$ bit detection, 4 bit decoder:

Component	Output bit width	Memory requirements
channel estimation	$q^{\mathrm{ce}} = 4$	0.64 kilobytes
detection	$q^{\mathrm{det}} = 4$	0.256 kilobytes
LDPC decoder	$q^{\mathrm{dec}} = 4$	44.8 kilobytes
total		45.7 kilobytes

notation: 1.0 kilobyte $= 1,000$ bytes

and compared to a conventional receiver chain. For this purpose, first an integer based equivalent to conventional channel estimation has been developed. Afterwards, a detection scheme to handle integer indices from this channel estimation scheme and quantization indices from a quantizer was presented. Both were modeled and visualized using Information Bottleneck graphs. The resulting detection scheme allows for application of an Information Bottleneck LDPC decoder. As a result, an Information Bottleneck receiver chain has been developed which performs all signal processing only using lookup tables constructed with the Information Bottleneck method. The best found receiver chain uses bit widths between 5 and 8 bits in its concatenated Information Bottleneck lookup tables and offers performance very close to that of a con-

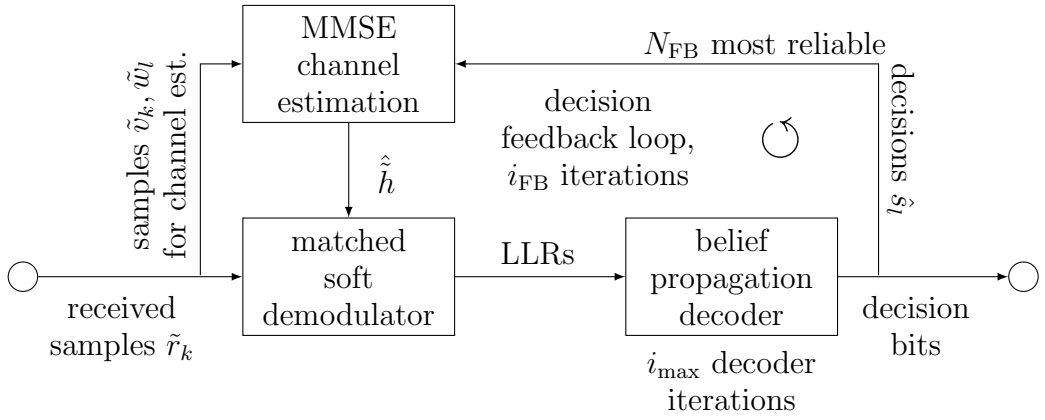
ventional receiver chain with double precision. Moreover, it outperforms a fixed point receiver with identical bit widths.

It was found that Information Bottleneck receivers used mismatched to their design-E_b/N_0 can offer remarkable bit error rate performance. As a result, only a single set of static lookup tables has to be constructed and stored to implement an entire receiver chain. The memory demand of the lookup tables required to implement the best investigated Information Bottleneck receiver lies in the order of several hundred kilobytes. It was also shown that the proposed receiver construction scheme allows to construct receivers with considerably smaller lookup tables by reducing the bit widths of the receiver components at the expense of a performance degradation. Therefore, the proposed receiver construction scheme allows to systematically trade implementation complexity for bit error rate performance.

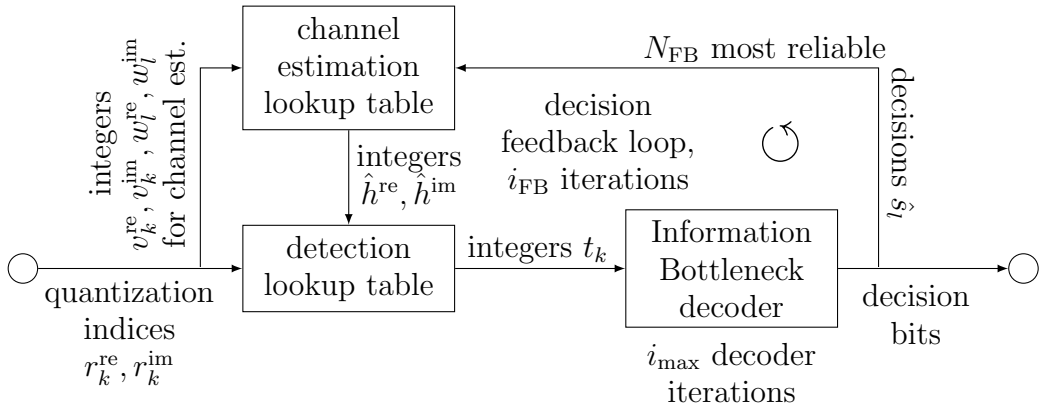
5.6 An Iterative Detection Scheme with Decision Feedback Channel Estimation

The insertion of pilot symbols into the transmitted signal introduces a pilot overhead and reduces the spectral efficiency. A well known technique to reduce the number of required pilot symbols in a coded transmission scheme is using decision feedback from the decoder to virtually increase the number of available pilot symbols at the receiver. Even very sophisticated soft decision feedback channel estimation schemes exist [80].

A conventional receiver chain for LDPC encoded transmission over a frequency flat fading channel with hard decision feedback is shown in Figure 5.48a. The processing of the depicted receiver chain was inspired by the one described in [81]. The considered receiver feeds back the BPSK symbols $\hat{s}_l$ which correspond to the most reliable bit decisions from the channel decoder to the MMSE channel estimator to virtually increase the number of available pilot symbols. In the considered scheduling the receiver first obtains initial channel estimates from a small number of pilot symbols. Based on this, soft demodulation is performed to obtain channel LLRs. The belief propagation decoder afterwards performs a maximum of $i_{\max}$ decoder iterations based on these LLRs. Then hard decisions are performed on the most reliable output



(a) Conventional receiver chain with decision feedback.



(b) Information Bottleneck receiver chain with decision feedback.

Figure 5.48: Comparison of a conventional receiver chain and an Information Bottleneck receiver chain with decision feedback. The feedback decisions are used to virtually increase the number of available pilot symbols for channel estimation.

bits of the decoder. These hard decisions are fed back to the channel estimator and used together with the known pilot symbols to determine refined channel estimates which are again delivered to the demodulator. This decision feedback loop is repeated i_{FB} times.

In this section, a receiver is presented which similarly applies decision feedback aided channel estimation in the lookup table based Information Bottleneck detection and decoding scheme from Section 5.5. This receiver is shown schematically in Figure 5.48b. Despite the structure of the receiver shown in Figure 5.48b looks identical to one of the conventional receiver above, the ma-

jor difference is again that all signal processing entities are designed with the Information Bottleneck method and implemented as simple lookup tables. The considered Information Bottleneck receiver has been developed in the scope of this thesis and was published in [82].

5.6.1 Feedback Aided Information Bottleneck Channel Estimation

The initial channel estimation, detection and decoding principles of the considered receiver are identical to the one from the previous section. After pilot based Information Bottleneck channel estimation has been performed using the Information Bottleneck graph from Figure 5.45, one has access to the integer index $\hat{h}^{\mathrm{re}}$. Since in the considered system one needs to distinguish between the pilot based channel estimation and a feedback based channel estimation to be presented next, the index FW abbreviating *forward* is added to $\hat{h}^{\mathrm{re}}$. The applied channel estimation scheme consists of the pilot based forward channel estimation scheme delivering $\hat{h}^{\mathrm{re}}_{\mathrm{FW}}$ and, additionally, a feedback (FB) aided channel estimator which makes use of the decisions on the most reliable codeword bits and delivers $\hat{h}^{\mathrm{re}}_{\mathrm{FB}}$.

If an LLR based decoder is used for decoding, the absolute magnitudes of its decision LLRs tell the reliability of the bit decisions. The output integers of the Information Bottleneck decoder similarly indicate a reliability information. This is caused by the fact that Algorithm 4 which is used to design the node operations in the Information Bottleneck LDPC decoder inherently sorts the clusters by their LLRs in the design process of the decoder. As a result, the smallest possible output integer $t^{(i)}_{\mathrm{dec}} = 0$ indicates a very reliable decision on the corresponding codeword bit being 1 and the largest possible $t^{(i)}_{\mathrm{dec}}$ indicates a very reliable decision on 0. It is, therefore, possible to identify the N_{FB} most reliable decisions of the Information Bottleneck LDPC decoder analogous to the LLR based decoding algorithms.

Let w^{re}_l, $l \in \{0, 1, \ldots, N_{\mathrm{FB}}-1\}$ denote the quantized integer indices r^{re}_k which correspond to the most reliable bit decisions in the decoded codeword. Let, moreover, $\hat{s}_l$ denote the respective hard decision BPSK symbols corresponding to these decision bits. The task of an Information Bottleneck feedback channel

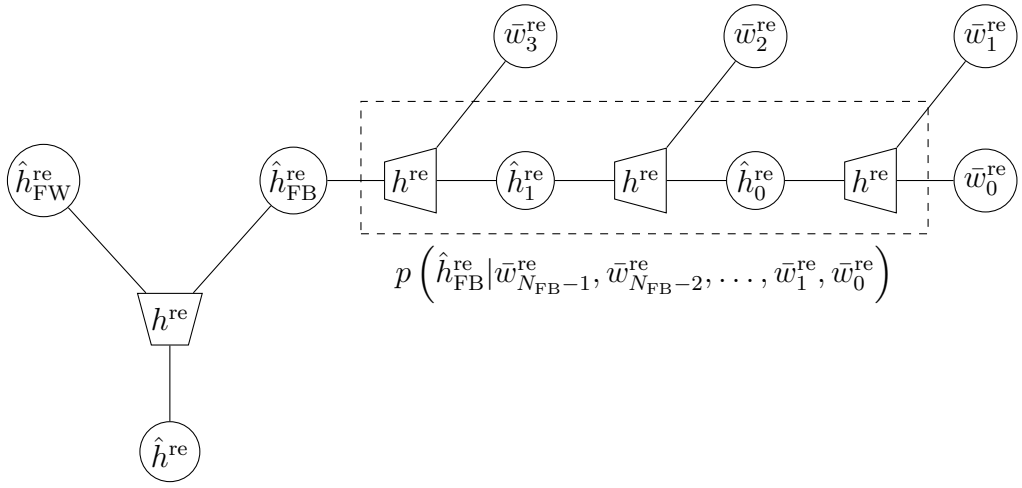


Figure 5.49: Information Bottleneck graph of a feedback channel estimation lookup table with opened node $p\left(\hat{h}_{\mathrm{FB}}^{\mathrm{re}}|\bar{w}_{N_{\mathrm{FB}}-1}^{\mathrm{re}}, \bar{w}_{N_{\mathrm{FB}}-2}^{\mathrm{re}}, \ldots, \bar{w}_1^{\mathrm{re}}, \bar{w}_0^{\mathrm{re}}\right)$ for $N_{\mathrm{FB}} = 4$. The output of the node aims to preserve information on the quantized real part of the unknown channel coefficient.

estimator is to extract information on h^{re} from $\hat{s}_l$ and w_l^{re}. Please note that the transformation

$$\bar{w}_l^{\mathrm{re}} = \begin{cases} w_l^{\mathrm{re}} & \hat{s}_l = +1 \\ 2^q - w_l^{\mathrm{re}} - 1 & \hat{s}_l = -1. \end{cases} \tag{5.60}$$

undoes the influence of $\hat{s}_l$ and makes all $\bar{w}_l^{\mathrm{re}}$ look like quantization indices received for the transmitted symbol $s_l = +1$, if the decision $\hat{s}_l$ is correct.

Figure 5.49 shows an Information Bottleneck graph of a feedback based Information Bottleneck channel estimation scheme which processes the transformed $\bar{w}_l^{\mathrm{re}}$ for an exemplarily number of $N_{\mathrm{FB}} = 4$ feedback symbols. The indices $\bar{w}_l^{\mathrm{re}}$ are inputs of two-input lookup tables inside the opened node $p\left(\hat{h}_{\mathrm{FB}}^{\mathrm{re}}|\bar{w}_0^{\mathrm{re}}, \bar{w}_1^{\mathrm{re}}, \ldots, \bar{w}_{N_{\mathrm{FB}}-1}^{\mathrm{re}}\right)$. This node is designed completely equivalent to the forward channel estimator from Figure 5.45 with the only difference that one assumes that the transmitted symbol $s_l = +1$ due to the transformation from Equation (5.60). Please note that this transformation is just a simple method to avoid the need of feeding $\hat{s}_l$ to the feedback channel estimator lookup tables and, therefore, to keep these lookup tables as small as possible.

The node $p\left(\hat{h}_{\mathrm{FB}}^{\mathrm{re}}|\bar{w}_{N_{\mathrm{FB}}-1}^{\mathrm{re}}, \ldots, \bar{w}_1^{\mathrm{re}}, \bar{w}_0^{\mathrm{re}}\right)$ delivers the integer $\hat{h}_{\mathrm{FB}}^{\mathrm{re}}$ which gathers the knowledge on h^{re} from the feedback path. The remaining task is the

combination of the knowledge on h^{re} from the forward and the feedback paths. This task is performed by the Information Bottleneck node $p\left(\hat{h}^{\mathrm{re}}|\hat{h}^{\mathrm{re}}_{\mathrm{FW}},\hat{h}^{\mathrm{re}}_{\mathrm{FB}}\right)$ in Figure 5.49. The joint distribution for its design is given by

$$p\left(h^{\mathrm{re}},\hat{h}^{\mathrm{re}}_{\mathrm{FW}},\hat{h}^{\mathrm{re}}_{\mathrm{FB}}\right)=p\left(h^{\mathrm{re}},\hat{h}^{\mathrm{re}}_{\mathrm{FW}}\right)p\left(h^{\mathrm{re}},\hat{h}^{\mathrm{re}}_{\mathrm{FB}}\right)\frac{1}{p\left(h^{\mathrm{re}}\right)}. \tag{5.61}$$

The joint distributions $p\left(h^{\mathrm{re}},\hat{h}^{\mathrm{re}}_{\mathrm{FW}}\right)$ and $p\left(h^{\mathrm{re}},\hat{h}^{\mathrm{re}}_{\mathrm{FB}}\right)$ are known from the Information Bottleneck design of the forward and the feedback channel estimation scheme, respectively. The distribution $p\left(h^{\mathrm{re}}\right)$ can be obtained by marginalization of one of these distributions.

For simplicity, it is assumed that the compression cardinalities of all Information Bottleneck algorithms applied to design the nodes shown in Figure 5.49 are chosen as $2^{q^{\mathrm{ce}}}$, such that the forward and the feedback channel estimation lookup tables both deliver q^{ce} bit integers.

After the first round of decision feedback in the conventional receiver shown in Figure 5.48a, typically a much more reliable channel estimate $\hat{\hat{h}}$ than in the initial decoding step will be available. Similarly, in the Information Bottleneck receiver, the mutual information $\mathrm{I}\left(\mathsf{H}^{\mathrm{re}};\hat{\mathsf{H}}^{\mathrm{re}}\right)$ should fulfil $\mathrm{I}\left(\mathsf{H}^{\mathrm{re}};\hat{\mathsf{H}}^{\mathrm{re}}\right)>\mathrm{I}\left(\mathsf{H}^{\mathrm{re}};\hat{\mathsf{H}}^{\mathrm{re}}_{\mathrm{FW}}\right)$.

Since the integer $\hat{h}^{\mathrm{re}}$ is used for detection in the Information Bottleneck graph from Figure 5.46 in the next feedback iteration, the Information Bottleneck nodes inside the detector have to be re-designed and matched to the more informative $\hat{h}^{\mathrm{re}}$. Also an updated LDPC decoder could be designed for the next decision feedback iteration by making use of the output distribution from the design of the adapted detector.

However, storing an updated LDPC decoder and an updated detector for each feedback iteration is complex. Therefore, the Information Bottleneck detector is only adapted to the first feedback iteration. As a result, one has to store one detector for the initial decoding round and another one for the feedback iterations. The LDPC decoder is held static for all feedback iterations to further reduce complexity. Hence, only a single Information Bottleneck LDPC decoder has to be stored. Also the feedback channel estimator from Figure 5.49 is held constant in all decision feedback iterations. Again, the idea is to construct all lookup tables offline and to use them mismatched to the

actual E_b/N_0 on the channel. The bit error rate performance of this approach is investigated in the following.

5.6.2 Performance and Discussion

This section investigates the bit error rate performance of the proposed feedback aided Information Bottleneck receiver chain for a frequency flat block fading channel and compares it to conventional receivers. For this purpose, the following scenario is considered:

1. The channel coefficients $\tilde{h}_k$ of the block fading channel are constant for $B = 35$ symbol durations.

2. $P = 3$ pilot symbols are inserted into the BPSK modulated codeword periodically with a distance of $B - P = 32$ codeword symbols.

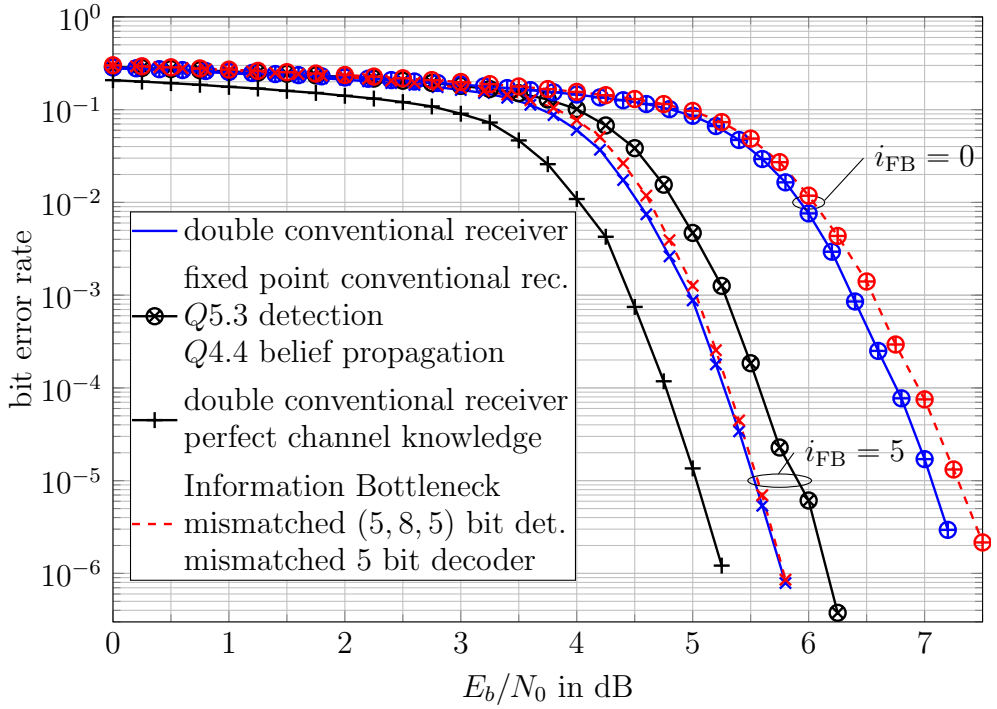
3. The applied LDPC code is the length $8,000$ $(3, 6)$ regular LDPC code 8000.4000.3.483 from [68] (*regular code a)*) with code rate $R = 0.5$. The maximum number of decoder iterations is $i_{\max} = 25$ for all applied decoders in each decision feedback iteration.

4. The $N_{\mathrm{FB}} = 20$ most reliable symbol decisions $\hat{s}_l$ in each fading block of length $B = 35$ are fed back to improve the channel estimate valid for the respective block.

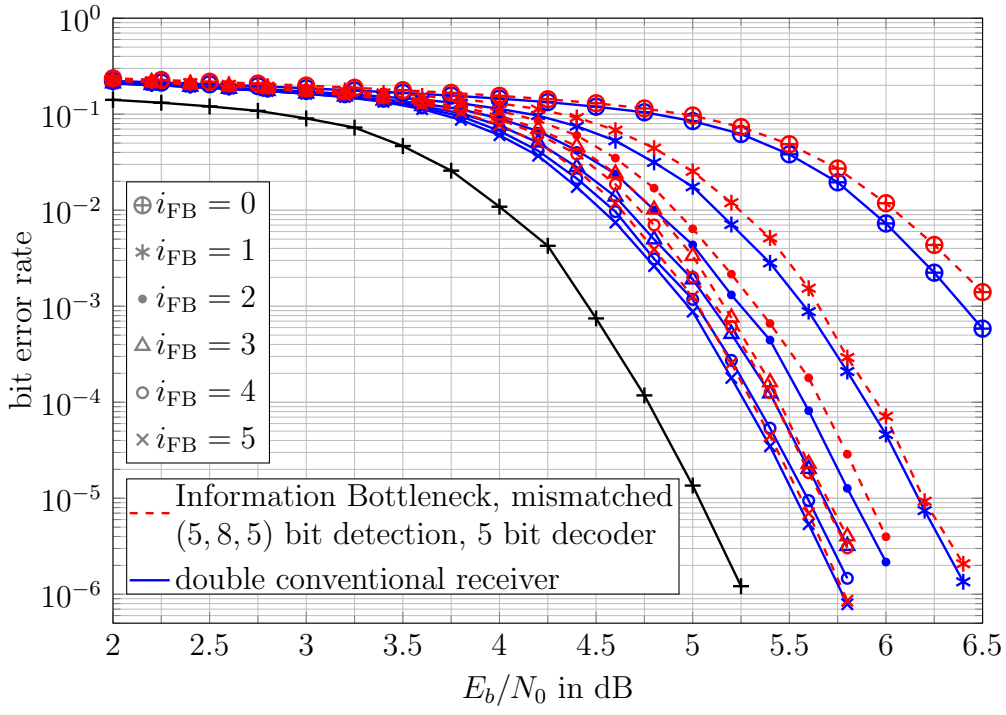
Again, the aforementioned parameters were chosen for exemplary illustration of the proposed receiver design method, but can be adapted easily.

Figure 5.50a shows the bit error rate of the Information Bottleneck receiver from Figure 5.48b with bit widths $(q, q^{\mathrm{ce}}, q^{\mathrm{det}}) = (5, 8, 5)$ used for channel estimation and detection and an Information Bottleneck LDPC decoder with 5 bit messages. Curves are provided for $i_{\mathrm{FB}} = 0$ and $i_{\mathrm{FB}} = 5$ outer iterations of the decision feedback loop (red dashed curves with $\oplus$ markers for $i_{\mathrm{FB}} = 0$ and $\times$ markers for $i_{\mathrm{FB}} = 5$).

The conventional receiver chain from Figure 5.48a serves as the most important reference (blue solid curves, with $\oplus$ markers for $i_{\mathrm{FB}} = 0$ and $\times$ markers for $i_{\mathrm{FB}} = 5$). Please note that this receiver does not use any quantization beyond the limits of floating point arithmetic with double precision. Anyway,



(a) Bit error rates for several iterative receivers with $i_{\mathrm{FB}} = 0$ and $i_{\mathrm{FB}} = 5$



(b) Bit error rates for several iterative receivers with $i_{\mathrm{FB}} = 0$ to $i_{\mathrm{FB}} = 5$.

Figure 5.50: Comparison of detection and decoding schemes for *regular code a)* with BPSK modulation on a block fading channel over E_b/N_0.

for $i_{\mathrm{FB}} = 5$ the loss of the proposed Information Bottleneck receiver is less than 0.1 dB over E_b/N_0. In fact, it is almost non-existent.

As another reference, the bit error rate of an 8 bit fixed point implementation of the conventional receiver is included in Figure 5.50a for $i_{\mathrm{FB}} = 5$ (solid black curve, $\otimes$ markers). This receiver used the $Q5.3$ fixed point format in the detection and channel estimation stages and the $Q4.4$ fixed point format for the messages in the belief propagation decoder. In order to not disadvantage the conventional receiver chain, also all other possible combinations of 8 bit fixed point formats were tested, but the chosen choice offered the best performance. The 8 bit fixed point conventional receiver is clearly outperformed by the Information Bottleneck receiver. This is particularly interesting because the Information Bottleneck receiver only used 8 bit integers in its channel estimation stage and smaller 5 bit integers in all other detection and decoding stages.

In the case without decision feedback, where $i_{\mathrm{FB}} = 0$, the Information Bottleneck receiver looses about 0.2 dB at a bit error rate around 10^{-5} in comparison to the double precision reference system. The observed gap is is a little larger than for the receiver with $(q, q^{\mathrm{ce}}, q^{\mathrm{det}}) = (6, 8, 5)$ bit detection and 5 bit decoding from Figure 5.47 because here the design-E_b/N_0 was tuned to yield the best performance with $i_{\mathrm{FB}} = 5$ feedback iterations and $q = 5$ instead of $q = 6$ bit channel output quantization has been used. The chosen design-E_b/N_0 was 5.65 dB and found by hand optimization.

It is remarkable that in both cases, with and without decision feedback, a receiver with $q = 5$ bit channel output quantization, $q^{\mathrm{ce}} = 8$ bit channel estimation, $q^{\mathrm{det}} = 5$ bit detection and 5 bit LDPC decoding almost achieves identical performance as a conventional receiver with double precision. This is even more fascinating when recalling that all lookup tables used in the Information Bottleneck receiver chain were designed offline and used mismatched to the actual channel quality. As a byproduct, other than in the conventional receiver chains, it is not even required to estimate the noise variance in the Information Bottleneck receiver.

Finally, the black curve with $+$ markers in Figure 5.50a shows the performance of the conventional receiver chain without feedback iterations, but perfect channel knowledge. This curve shall serve as a performance bound.

Table 5.7: Memory demands of receiver components

Component	Output bit width	Memory requirements
forward channel estimation	$q^{\mathrm{ce}} = 8$	9.22 kilobytes
feedback channel estimation	$q^{\mathrm{ce}} = 8$	214.02 kilobytes
initial detector	$q^{\mathrm{det}} = 5$	5.76 kilobytes
detector feedback iterations	$q^{\mathrm{det}} = 5$	5.76 kilobytes
LDPC decoder	$q^{\mathrm{dec}} = 5$	112.0 kilobytes
total		346.76 kilobytes

notation: 1.0 kilobyte = 1,000 bytes

So far, Figure 5.50a has shown that the performance of the Information Bottleneck receiver is very close to that of the conventional receiver with double precision, if $i_{\mathrm{FB}} = 5$ feedback iterations are performed in both receivers. An open question, however, still is whether or not the same Information Bottleneck receiver falls short, if fewer feedback iterations are performed. Therefore, Figure 5.50b shows the bit error rates of both receivers for all $i_{\mathrm{FB}} \in \{0, 1, 2, 3, 4, 5\}$. The black curve with + markers in Figure 5.50b again refers to the conventional receiver chain without feedback iterations, but perfect channel knowledge. The figure clearly shows that the performance of the Information Bottleneck receiver can compete with the double precision conventional receiver for all investigated numbers of feedback iterations i_{FB}. The figure also indicates that the gains of the feedback loop are very significant in the first three feedback iterations for both receivers and then become less significant. It was found that using more than five feedback iterations hardly improves the bit error rate performance of any investigated receiver.

The memory demands required to store the Information Bottleneck receiver components are listed in Table 5.7. Formulas to calculate the size of the receiver components for a given set of parameters can be found in Appendix A.7.2. Again the total memory amount required to store the lookup tables lies in the order of several hundred kilobytes.

5.6.3 Summary

In this section the Information Bottleneck based coherent detection scheme for frequency flat fading channels from Section 5.5 has been improved by including a feedback aided iterative channel estimation scheme. The feedback channel estimation scheme was developed and described in an Information Bottleneck graph. The decision feedback loop was shown to greatly improve the bit error rate performances of all applied receiver chains.

Comparable to the case without decision feedback, the bit error performance of the designed Information Bottleneck receiver with 5 bit channel output quantization, 8 bit channel estimation, 5 bit detection and 5 bit decoding is very close to that of a conventional feedback aided receiver with double precision MMSE channel estimation, soft demodulation and belief propagation decoding. Furthermore, the designed Information Bottleneck receiver outperforms a conventional feedback aided receiver with 8 bit fixed point arithmetic. The memory demand to store the lookup tables for the Information Bottleneck receiver lies in the order of several hundred kilobytes and still is easy to handle in practice.

The presented results show that it is possible to adapt advanced signal processing techniques like decision feedback from conventional receivers to Information Bottleneck receivers. Interestingly, the resulting Information Bottleneck receivers can compete with their conventional counterparts with high precision while using low bit widths and very simple lookup operations. It is also very interesting that the investigated Information Bottleneck receiver showed very good performance, although its lookup tables were designed offline and used mismatched to the design-E_b/N_0.

Chapter 6

Implementation and Evaluation of Information Bottleneck LDPC Decoders on a Digital Signal Processor

In the previous chapter, it was shown that the Information Bottleneck approach to signal processing offers surprisingly good performance with strongly quantized signal representations. All the signal processing operations in the resulting signal processing blocks become lookup operations in static tables. It is intuitive that lookup operations have low complexity, as the only arithmetical operation required to implement a lookup table is the calculation of memory addresses to determine the outcome of the operation. Furthermore, signal representation with a few bits is desirable from an implementation perspective. An open question, however, still is which practical advantages in comparison to state-of-the-art signal processing the proposed technique offers in a specific implementation on a hardware platform. This chapter tries to close this gap. Its most important aim is to give practical evidence of such advantages of Information Bottleneck signal processing over state-of-the-art signal processing techniques and to quantify them. The focus of this investigation is on LDPC decoding. Practical implementations of Information Bottleneck LDPC decoders are presented. These implementations have been developed in the scope of this thesis and published in [83, 84]. The belief propagation and the min-sum decoding algorithms presented in Section 2.3.3 were also implemented for comparison. Related works by other authors which focus on implementation aspects of LDPC decoders with a similar decoding principle are [30, 34]. These works focus on dedicated chip implementations of the decoder. In contrast, this work considers decoder implementations on a DSP.

A DSP implementation is of special interest for SDR applications. The driving idea of SDRs is to implement elementary parts of a radio signal processing chain in software which is run on a general purpose signal processing platform [85, 86]. Using this approach, crucial design parameters of a communications chain, such as the channel coding/decoding and modulation/demodulation schemes can be adapted with a simple software update. This of course offers great flexibility. However, SDR development comprises mapping signal processing algorithms to the available signal processing resources of a platform and is, therefore, challenging from an engineering perspective.

In the following, several specific implementation aspects of Information Bottleneck LDPC decoders and state-of-the-art decoders on a DSP will be con-

sidered. Practical results on the bit error rates, the memory demands and the decoding throughputs of the distinct decoders will be presented and discussed.

6.1 Implementation Scope, Hardware Properties and Parameters

In order to develop an optimized decoder implementation on a DSP, one has to tailor the decoder implementation to the parameters of a specific LDPC code and the DSP's resources to be able to utilize the resources of the platform as good as possible. The following scope is considered here:

1. The implementations to be presented focus on $(3, 6)$ regular codes. The length $8,000$ code *regular code a)* and the length $2,640$ code *regular code b)* will be used for performance evaluation of all implemented decoders.

2. The applied modulation scheme is BPSK and the considered transmission channel is an AWGN channel.

3. The focus lies on a q bit decoder implementation of the Information Bottleneck decoder, where all message alphabets $\mathcal{Y}_{\text{chan}}, \mathcal{T}_{\text{var}}$ and $\mathcal{T}_{\text{chk}}$ are chosen identical as $\{0, 1, \ldots, 2^q - 1\}$. In particular using $q = 4$ bit messages is considered.

4. All considered decoders perform a maximum of $i_{\max} = 50$ decoder iterations.

The target DSP is a Texas Instruments TMS320C6474 DSP [87]. This chip will be termed C6474 in the following. A complete specification of the DSP and details on its architecture can be found in [87]. The C6474 was embedded on an eInfochips TMDSEVM6474L evaluation board [88] and clocked with a main frequency of 1 GHz.

An Information Bottleneck decoder, a belief propagation decoder which implements the function $f_c(x)$ from Equation (2.55) using a small lookup table and a min-sum decoder were implemented on this device. All software run on the DSP was written in C++ programming language and compiled using the DSP vendor's compiler.

6.1.1 Determining Hardware Parameters

Most importantly, the C6474 is a fixed point DSP. Since the chip does not offer native floating point operations, they have to be emulated in software, if required. This emulation is much slower than native fixed point operations. Therefore, all decoder implementations presented here use only fixed point and integer arithmetic.

Especially for Information Bottleneck LDPC decoders, but also for all conventional LDPC decoders, memory access is one of the main performance factors. In all LDPC decoders, the messages have to be read from and written back into the memory. In an Information Bottleneck LDPC decoder, additionally the lookup tables lying in the memory have to be accessed extremely often.

In our setup, a four layer memory architecture was present: The C6474 provides 32 registers with a width of 32 bit each for one of two functional units. These registers are accessible by the arithmetical and logical unit (ALU) without any stall cycles. As it will be discussed later, the registers can be utilized to store intermediate results in the decoding process. Moreover, the C6474 provides a layered cache which is separated into a very fast level one (L1) cache and a slightly slower level two (L2) cache. The size of the L1 data (L1D) cache was 32 kilobyte. L1D access can be considered to be one of the most determining operations in the Information Bottleneck LDPC decoder because segments of the lookup tables have to be accessed by all variable and all check nodes in a particular decoder iteration. Therefore, cache hits and misses strongly influence the throughput of the decoder. The accessible L2 cache was 1024 kilobyte. The complete caching process is mostly invisible to the DSP programmer, but should be considered inherently during programming. Finally, the chosen evaluation board provides 256 megabytes of DDR2 RAM which is clocked with 66 MHz and is very slow in comparison to the mentioned caches.

The C6474 has several built-in functions which can be utilized in the LDPC decoding process in conventional and Information Bottleneck LDPC decoders. These functions are so called *intrinsics* and can be executed in a single CPU cycle [89]. For example, there are several minima intrinsics allowing to find the minimum of two fixed point numbers. These intrinsics are useful and

were used in the min-sum and the belief propagation decoder. Another very important intrinsic is bit field extraction. Bit field extraction allows to extract an arbitrary subgroup of neighboring bits from a longer word in the memory by a left-right shift combination of a register. Using bit field extraction, it is possible to utilize standard $8, 16$ and 32 bit data types to embed several 4 bit integers. This can be used to significantly reduce the memory footprint of an Information Bottleneck LDPC decoder, as it will be discussed soon.

6.2 DSP Implementation Aspects of Information Bottleneck LDPC Decoders

Information Bottleneck LDPC decoders are special because their only required arithmetical operations are address calculations in lookup tables. In the following, the implementation of lookup tables in DSP implementations of Information Bottleneck decoders is discussed. Afterwards, two distinct options for its practical realization are presented. Finally, an implementation optimization which reduces the number of required lookup operations for generation of the outgoing messages in the decoding will be considered.

6.2.1 DSP Implementation View on Information Bottleneck Lookup Tables

Despite the design process of an Information Bottleneck LDPC decoder requires some sophisticated information theory, from an implementation perspective on a DSP, the resulting check and variable node operations are very simple. First, consider an arbitrary two-input lookup table in any opened node structure, as it is depicted on the top of Figure 6.1. This table inputs incoming integers $y_0^{(i)}, y_1^{(i)} \in \{0, 1, \ldots, 2^q - 1\}$ and delivers the outgoing integer $t_m^{(i)} \in \{0, 1 \ldots, 2^q - 1\}$ in iteration i by evaluation of the function $f_m^{(i)}\left(y_0^{(i)}, y_1^{(i)}\right)$. Below the illustration of the lookup table, a view on its implementation on a DSP is provided which will be explained further in the following.

The logic of the function implemented in the lookup table is essentially stored in a static vector $\mathbf{lut}_m^{(i)}$. The programming equivalent to a static vector is an array lying in the static memory. An array is a connected memory region

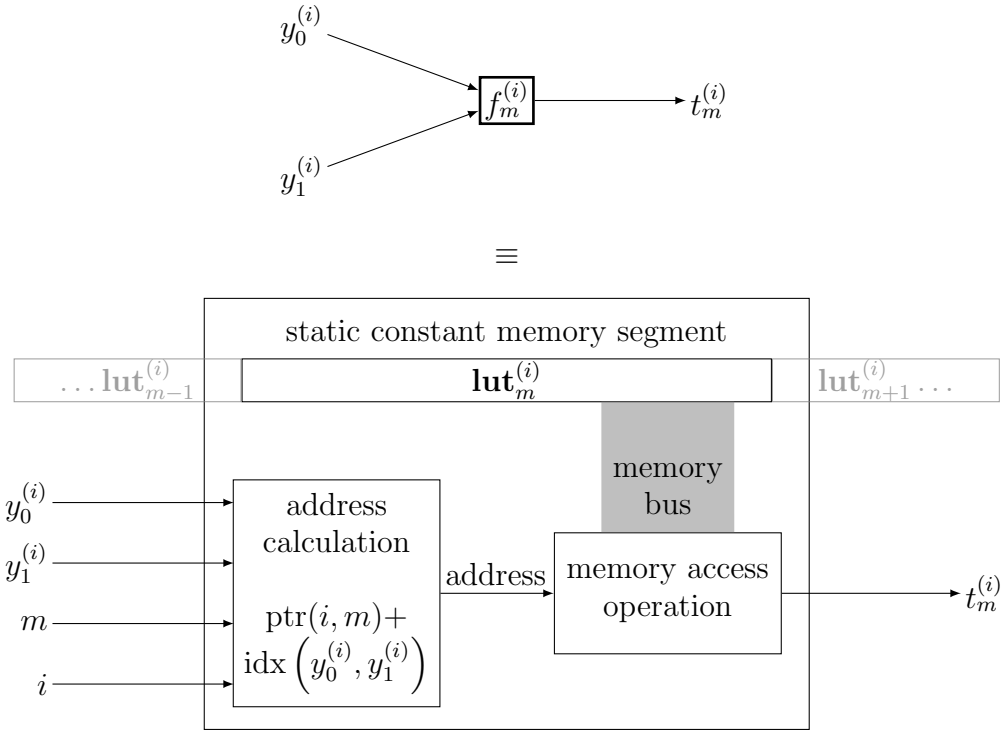


Figure 6.1: DSP implementation view on a lookup table implementing the function $f_m^{(i)}\left(y_0^{(i)}, y_1^{(i)}\right)$ which is designed with the Information Bottleneck method.

consisting of multiple cells that all hold data of the same data type. Access to an array is done using indexing. In the simplest case, each memory cell in this array corresponds to one entry of the vector. The length of this vector corresponds to the number of possible input configurations of $\left(y_0^{(i)}, y_1^{(i)}\right)$ which exist. For $y_0^{(i)}, y_1^{(i)} \in \{0, 1, \ldots, 2^q - 1\}$, this number is $(2^q)^2 = 2^{2q}$. The vector $\mathbf{lut}_m^{(i)}$ holds each outcome $t_m^{(i)}$ of the two-input operation at a particular position. This position has to be determined from $y_0^{(i)}$ and $y_1^{(i)}$ by making use of an indexing function which maps each pair onto a unique position. This function is denoted idx $\left(y_0^{(i)}, y_1^{(i)}\right)$. An important design aim is choosing an indexing function which is very simple to evaluate because the complexity of index calculation mainly influences the complexity of the lookup table implementation. A function meeting this requirement is

$$\text{idx}\left(y_0^{(i)}, y_1^{(i)}\right) = y_0^{(i)} \cdot 2^q + y_1^{(i)}. \tag{6.1}$$

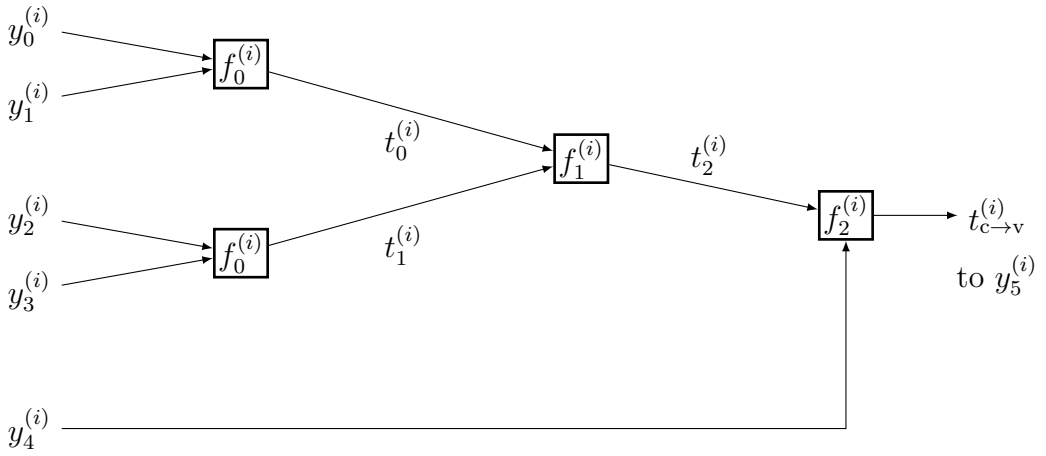


Figure 6.2: Exemplary two-input decomposition of a check node lookup table for a degree $d_\mathrm{c} = 6$ check node operation.

The multiplication of $y_0^{(i)}$ by 2^q in this function can be performed as a simple bit shift operation. The binary representation of $y_0^{(i)}$ has to be shifted q bits to the left. This operation zeroizes the lower q bits which are filled up with the binary representation of $y_1^{(i)}$ by the addition of $y_1^{(i)}$ in Equation (6.1). Therefore, evaluating $\mathrm{idx}\left(y_0^{(i)}, y_1^{(i)}\right)$ from Equation (6.1) just corresponds to writing the binary q bit representations of $y_0^{(i)}$ and $y_1^{(i)}$ next to each other in a memory cell which holds $2q$ bits. Once the position of $t_0^{(i)}$ in vector $\mathbf{lut}_m^{(i)}$ has been determined, the final step is reading it from the memory. The outcome of the two-input operation is determined as

$$t_m^{(i)} = \mathbf{lut}_m^{(i)}\left[\mathrm{idx}\left(y_0^{(i)}, y_1^{(i)}\right)\right]. \tag{6.2}$$

On a DSP, this memory access operation involves the memory bus and also the caches, as it is illustrated in Figure 6.1. After the read operation has been executed, $t_m^{(i)}$ is available in a register and can be processed further by the ALU.

As it has already been discussed in Section 5.2.1, in an Information Bottleneck decoder any node operation is decomposed into several concatenated two-input lookup tables in an opened node structure. As a result, the described mechanism is executed several times at every node in a particular decoder iteration i. Let us consider the decomposition of a degree $d_\mathrm{c} = 6$ check node operation shown in Figure 6.2 exemplarily. Letting M denote the number of *distinct* two-input lookup tables in a node decomposition, one finds that $M = 3$

for this decomposition because $f_0^{(i)}\left(y_0^{(i)}, y_1^{(i)}\right)$ and $f_0^{(i)}\left(y_2^{(i)}, y_3^{(i)}\right)$ can use the same lookup table due to their identically distributed inputs. Implementing all two-input operations of a particular opened node structure on a DSP is easy by a simple concatenation of the vectors $\mathbf{lut}_m^{(i)}$ for $m \in \{0, 1, \ldots, M-1\}$ which determine the respective two-input lookup tables $f_m^{(i)}$. This is also illustrated in Figure 6.1. Each two-input lookup table $f_m^{(i)}$ works on a certain memory region which holds the corresponding vector $\mathbf{lut}_m^{(i)}$. It is natural to also concatenate all two-input lookup table vectors $\mathbf{lut}_0^{(i)}, \mathbf{lut}_1^{(i)}, \ldots, \mathbf{lut}_{M-1}^{(i)}$ for all decoder iterations $i \in \{0, 1, \ldots, i_{\max} - 1\}$. In doing so, one obtains a long vector

$$\mathbf{LUT} = \Big[\mathbf{lut}_0^{(0)}, \mathbf{lut}_1^{(0)}, \ldots, \mathbf{lut}_{M-1}^{(0)}, \mathbf{lut}_0^{(1)}, \mathbf{lut}_1^{(1)}, \ldots, \mathbf{lut}_{M-1}^{(1)}, \ldots, \mathbf{lut}_0^{(i_{\max}-1)}, \mathbf{lut}_1^{(i_{\max}-1)}, \ldots, \mathbf{lut}_{M-1}^{(i_{\max}-1)}\Big]. \quad (6.3)$$

which characterizes all two-input operations of the check or the variable nodes in all decoder iterations. Please note that one of such vectors **LUT** is required for the variable nodes and different one is required for the check nodes.

With the vector **LUT**, any two-input lookup table $f_m^{(i)}$ can calculate its output w under inputs u, v in iteration i as

$$w = \mathbf{LUT}\left[\mathrm{ptr}(i, m) + \mathrm{idx}(u, v)\right]. \quad (6.4)$$

The function $\mathrm{ptr}(i, m)$ points to first entry of $\mathbf{lut}_m^{(i)}$ in vector **LUT**. It is calculated as

$$\mathrm{ptr}(i, m) = iM \cdot 2^{2q} + m \cdot 2^{2q}. \quad (6.5)$$

The evaluation of the right-hand side of Equation (6.4) essentially is the only required operation in an Information Bottleneck decoder for the variable and for the check nodes. It fetches an element from a certain position of the respective **LUT** vector.

6.2.2 Lookup Table Reduction

In Section 5.2.5 it was shown that $q = 4$ bit messages are sufficient to provide performance very close to that of double precision belief propagation decoding. Most DSPs natively support data types consisting of several bytes (8 bits). As it has already been mentioned above, from a programming perspective, it is

completely natural to store each **LUT** vector in an array. The smallest standard data type in C++ which is suitable to store a $q = 4$ bit message is a *uint8* integer, where *uint* abbreviates *unsigned integer* and the appended 8 denotes the wordlength of 8 bits. Reserving a *uint8* memory cell for each lookup table entry, however, wastes the upper half word of the cell because when storing a number from $\{0, 1, \ldots, 2^4 - 1\}$ in this cell, the upper halfword is always 0000 in the binary representation. A very simple idea to reduce the memory footprint of the decoder is to use the upper halfword and the lower halfword to store two consecutive lookup table entries. This of course halves the size of the lookup table arrays in comparison to the described straightforward allocation of a *uint8* array. Accessing an entry in the lookup table, however, requires extra operations.

In Equation (6.4), the vector **LUT** is accessed at position

$$\mathrm{addr}(i, m, u, v) = \mathrm{ptr}(i, m) + \mathrm{idx}(u, v). \tag{6.6}$$

If only one table entry per element is stored in the array holding this vector, accessing the array at index $\mathrm{addr}(i, m, u, v)$ automatically delivers the desired table entry. If, in contrast, two table entries are held concatenatedly in the upper and the lower halfword of each *uint8* memory cell, $\mathrm{addr}(i, m, u, v)$ needs to be divided by 2 according to integer division rules prior to accessing the array. Array access at $\lfloor \mathrm{addr}(i, m, u, v)/2 \rfloor$ then fetches 8 bits, that is, *two consecutive* table entries in one byte from the memory. Thus, depending on whether $\mathrm{addr}(i, m, u, v)$ is odd or even, the upper or the lower halfword needs to be extracted by an additional bit field extraction. The C6474 offers a very fast intrinsic operation for this purpose. The proposed technique reduces the memory required to store the lookup tables in an Information Bottleneck decoder. As a result, it will be termed *lookup table reduction* in the following.

Including the caches, the C6474 makes up a complex system. There is no simple answer to the question of the best choice of optimization towards a reduced memory footprint or a reduced number of operations for the lookup table implementation. In the following a technique is presented which results in *larger* lookup tables in the memory that need a reduced number of operations for the lookup table access during decoding. Both techniques will be compared in practice in Section 6.3.

Finally, it shall be emphasized that the principle of storing several $q = 4$ bit messages in a single memory cell holding a longer base data type is not limited to the table entries of the lookup tables, but can also be applied to reduce the memory required to store the the exchanged messages. Consider, for example a degree $d_{\mathrm{v}} = 3$ variable node which has to process a message from the channel and $d_{\mathrm{v}} = 3$ incoming messages for its bit decision. All four incoming messages of the node can be stored in a single *uint16* integer by sequentially shifting the binary entries of the memory cell 4 bits to the left and adding another 4 bit message after each shift.

6.2.3 Lookup Table Expansion

In the following it is assumed that a straightforward allocation of one array cell per lookup table entry is performed. The concatenation of two-input lookup tables appearing in an opened node structure then allows for a very interesting implementation optimization of Information Bottleneck LDPC decoders on a DSP. This optimization is applicable to variable and to check nodes. Please recall the check node decomposition shown in Figure 6.2, exemplarily.

In this node decomposition, intermediate results $t_0^{(i)}$ and $t_1^{(i)}$ are being processed by the concatenated lookup table $f_1^{(i)}\left(t_0^{(i)}, t_1^{(i)}\right)$. The intermediate result $t_2^{(i)}$ delivered by the concatenated lookup table is fed forward to the next concatenated lookup table $f_2^{(i)}\left(t_2^{(i)}, y_4^{(i)}\right)$. As a result, one can summarize that the intermediate results appearing at the distinct two-input lookup tables are fed forward and serve as inputs to other two-input lookup tables. In the end, however, one only needs the final outcome $t_{\mathrm{c}\to\mathrm{v}}^{(i)}$ for decoding.

Expanding the position in vector **LUT** required to determine the outcome of $f_1^{(i)}\left(t_0^{(i)}, t_1^{(i)}\right)$ gives an interesting insight. This position is

$$\mathrm{addr}\left(i, 1, t_0^{(i)}, t_1^{(i)}\right) = \underbrace{iM \cdot 2^{2q} + 1 \cdot 2^{2q} + t_0^{(i)} \cdot 2^{q}}_{\bar{t}_0^{(i)}} + t_1^{(i)} = \bar{t}_0^{(i)} + t_1^{(i)}. \tag{6.7}$$

The braced part in Equation (6.7) only depends on the iteration index i, the index $m = 1$ of the lookup table $f_1^{(i)}\left(t_0^{(i)}, t_1^{(i)}\right)$ and its first input $t_0^{(i)}$ which is the outcome of the previous operation. As a result, instead of storing $t_0^{(i)}$ at position $\mathrm{ptr}(i, 0) + \mathrm{idx}\left(y_0^{(i)}, y_1^{(i)}\right)$ in **LUT**, from where $f_0^{(i)}\left(y_0^{(i)}, y_1^{(i)}\right)$ reads its

output $t_0^{(i)}$, one could also directly store the braced part of Equation (6.7) at this position. This braced part is denoted by $\bar{t}_0^{(i)}$.

This modification has the advantage that the next concatenated lookup table $f_0^{(i)}\left(t_0^{(i)}, t_1^{(i)}\right)$ can calculate its next accessed index in **LUT** simply using one integer addition as

$$\text{addr}\left(\bar{t}_0^{(i)}, t_1^{(i)}\right) = \bar{t}_0^{(i)} + t_1^{(i)}. \tag{6.8}$$

This greatly reduces the computational complexity of the address calculation in comparison to evaluation of Equation (6.6) from $t_0^{(i)}$ and $t_1^{(i)}$.

Please recall that $t_0^{(i)}$ and $t_1^{(i)}$ are read from the same subvector $\mathbf{lut}_0^{(i)}$ in **LUT** in the opened node structure shown in Figure 6.2 because both operations $f_0^{(i)}\left(y_0^{(i)}, y_1^{(i)}\right)$ and $f_0^{(i)}\left(y_2^{(i)}, y_3^{(i)}\right)$ have identically distributed inputs and, therefore, the same subscript $m = 0$. As a result, also a modified $\bar{t}_1^{(i)}$ which includes offsets for the subtable index m and the iteration number i would be read from this vector, if the lookup tables were implemented in the proposed manner. In order to evaluate the right-hand side of Equation (6.8), however, $t_1^{(i)} \neq \bar{t}_1^{(i)}$ is required. Fortunately, $t_1^{(i)}$ can easily be reobtained vom $\bar{t}_1^{(i)}$ by making use of the relation

$$t_1^{(i)} = \underbrace{\left(iM \cdot 2^{2q} + (0+1)2^{2q} + t_1^{(i)} \cdot 2^q\right)}_{\bar{t}_1^{(i)} \text{ read from } \mathbf{lut}_0^{(i)} \text{ in } \mathbf{LUT}} / 2^q \bmod 2^q. \tag{6.9}$$

The technique described can be applied similarly to most of the two-input lookup tables in arbitrary node decompositions, such that the address calculation in the concatenated structure is greatly simplified. Only the last two-input lookup table which provides the final outcome of the opened node structure has to deliver the original integer index $t_{\mathrm{c}\to\mathrm{v}}^{(i)} \in \{0, 1, \ldots, 2^q - 1\}$ in Figure 6.2. Therefore, the subvectors $\mathbf{lut}_{M-1}^{(i)}$ stored for this table stay unchanged for all decoder iterations i.

The modified table entries $\bar{t}_m^{(i)}$ are no longer $q = 4$ bit integers because they include offsets for the iteration number i and the table index m originally obtained using the function $\text{ptr}(i, m)$. As a result, an elementary data type with a bit width of 8 bits is not sufficient to store the modified $\bar{t}_m^{(i)}$ in an array

which represents the vector **LUT**. The largest possible $\bar{t}_m^{(i)}$ which can appear is

$$\max_{i,m} \bar{t}_m^{(i)} = 2^q \cdot \underbrace{(2^q - 1)}_{\max\limits_{i,m} t_m^{(i)}} + \max_{i,m} \mathrm{ptr}(i,m) =$$

$$(i_{\max} - 1)M \cdot 2^{2q} + (M-1) \cdot 2^{2q} + 2^q(2^q - 1) =$$

$$i_{\max} M \cdot 2^{2q} - 2^q. \quad (6.10)$$

As a result, the required bit width of the elements in the array holding the vector **LUT** is

$$\left\lceil \log_2 \left(\max_{i,m} \bar{t}_m^{(i)} \right) \right\rceil =$$

$$\left\lceil \log_2 \left(i_{\max} M \cdot 2^{2q} - 2^q \right) \right\rceil \leq \left\lceil 2q + \log_2(i_{\max} M) \right\rceil . \quad (6.11)$$

With a number of $i_{\max} = 50$ maximum decoder iterations, $M = 3$ distinct two-input lookup tables in the degree $d_c = 6$ check node decomposition shown in Figure 6.2 and $q = 4$ bit messages, this yields 16 bits per element in **LUT**. The corresponding base data type in C++ to be used is the *uint16* type.

In comparison to the lookup table reduction technique presented in the preceding section, where two $q = 4$ bit lookup table entries were stored in a single *uint8* memory cell, this greater base data type increases the size of the **LUT** arrays by a factor of four. As a result, the considered lookup table implementation technique is termed *lookup table expansion*. The memory amounts of Information Bottleneck decoders with lookup table reduction and lookup table expansion as well as the ones of state-of-the-art decoders will be compared in Section 6.3.

6.2.4 Reusing Intermediate Results

Each variable and each check node has to generate outgoing messages for all its connected edges during the iterative decoding process. Recall that Figure 6.2 depicts the generation of the outgoing message to be passed along the target edge which delivered $y_5^{(i)}$ as $y_5^{(i)}$ is excluded on the input side on the left. Message generation for any other target edge requires to exchange any input message on the left with $y_5^{(i)}$ to generate the message for the edge which

delivered the respective replaced message. Afterwards, one has to re-evaluate the two-input operations depicted, if required. For example, if one replaced $y_0^{(i)}$ by $y_5^{(i)}$, one would have to re-evaluate $f_0^{(i)}\left(y_5^{(i)}, y_2^{(i)}\right)$ followed by $f_1^{(i)}\left(t_0^{(i)}, t_1^{(i)}\right)$ and $f_2^{(i)}\left(t_2^{(i)}, y_4^{(i)}\right)$. However, in this case the intermediate result $t_1^{(i)}$ does not change and, therefore, one can *reuse* it and dismiss the re-evaluation of $f_0\left(y_2^{(i)}, y_3^{(i)}\right)$. Obviously, replacing $y_0^{(i)}$ by $y_5^{(i)}$ to determine the message to be propagated along the edge which delivered $y_0^{(i)}$ is not the best idea. In contrast, replacing $y_4^{(i)}$ by $y_5^{(i)}$ to determine the message to be propagated along the edge which delivered $y_4^{(i)}$ results in the fact that only $f_2\left(y_5^{(i)}, t_2^{(i)}\right)$ has to be re-evaluated for determination of the new outgoing message. In this case, all intermediate results $t_m^{(i)}$ stay unchanged.

If the principle of reusing intermediate results is applied for the generation of all d_c outgoing messages, it offers considerable savings in the number of required operations. Figure 6.3 shows a node decomposition which makes extensive use of this principle for a degree $d_\mathrm{c} = 6$ check node. As it can be seen there, the shown decomposition enables to generate all $d_\mathrm{c} = 6$ outgoing messages using only twelve two-input operations. Without reusing any intermediate results, this process would require d_c evaluations of $d_\mathrm{c} - 2$ two-input operations and, hence, $d_\mathrm{c}(d_\mathrm{c}-2)$ operations in total which is 24 in this example.

The decomposition shown in Figure 6.3 inherently contains the one already depicted in Figure 6.2 as it is highlighted using dashed edges. Therefore, the decomposition from Figure 6.3 also requires to store only $M = 3$ distinct vectors $\mathbf{lut}_m^{(i)}$ for implementation of the entire node operation in an Information Bottleneck decoder. Finding optimum node decompositions with the maximum reuse potential of intermediate results *and additionally* a minimum number M of distinct two-input lookup tables for a given node degree is a complex combinatorial problem. In fact, the one depicted in Figure 6.3 is the best found for the parameters considered here. The optimum choice of a node decomposition could also depend on the target DSP because one has to be able to store all intermediate results appearing close to the ALU, such that they can be accessed without an undesirable delay. As it has already been mentioned, the C6474 offers 32 registers which are accessible by the ALU without any stall cycles. The six intermediate results $t_m^{(i)}$ appearing in Figure 6.2 easily fit into these registers without blocking all of them.

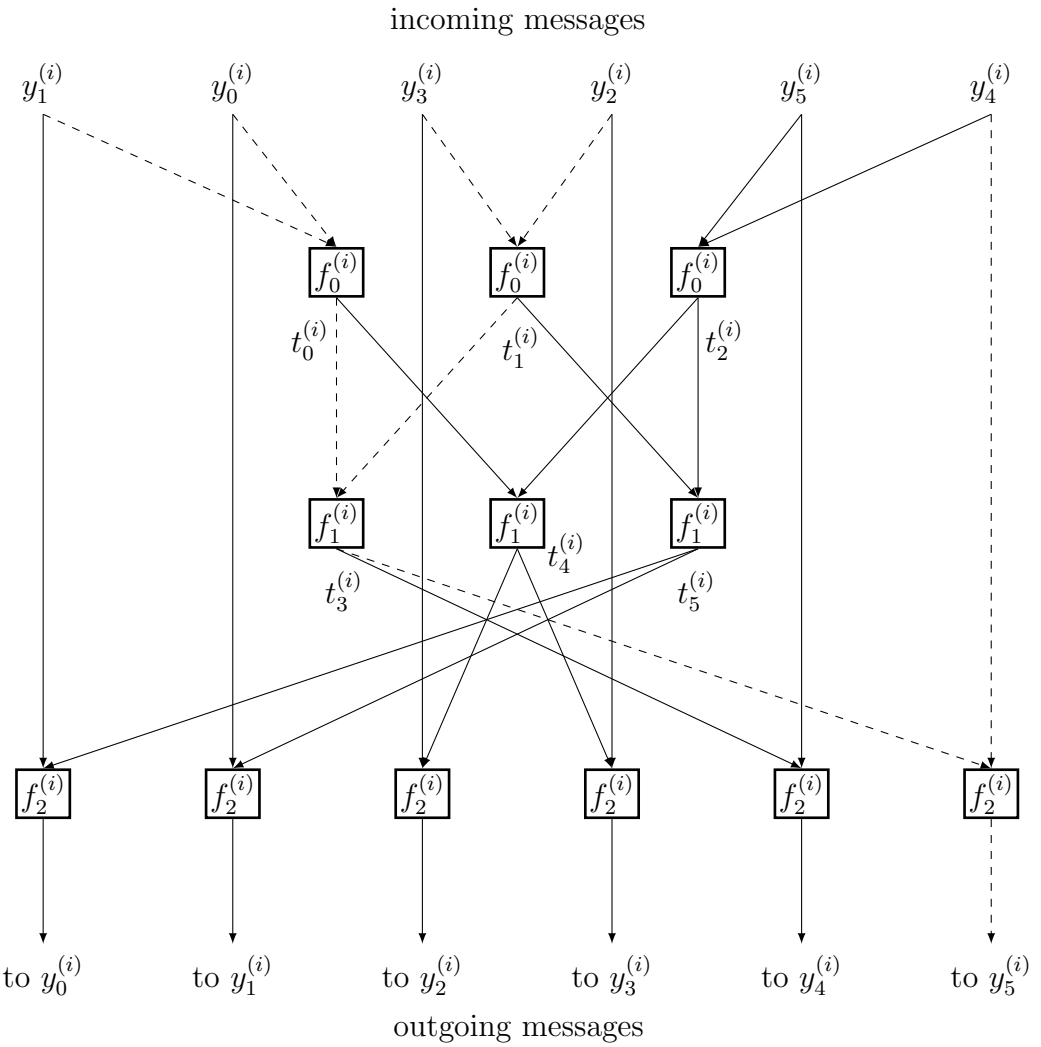


Figure 6.3: Message generation of a degree $d_\mathrm{c} = 6$ check node which utilizes reuse of intermediate results.

Reusing intermediate results does not affect the possibility to use lookup table reduction and lookup table expansion described in Sections 6.2.2 and 6.2.3. As a consequence, these techniques can be combined with the principle of reusing intermediate results. Moreover, it is irrelevant whether the appearing two-input operations are lookups of integers or two-input operations of LLRs. Therefore, the same method can also be applied in any conventional LDPC decoder using the box-plus or the min-sum operation. Since the conventional decoders also profit from performing as few operations as possible, this is done in this work.

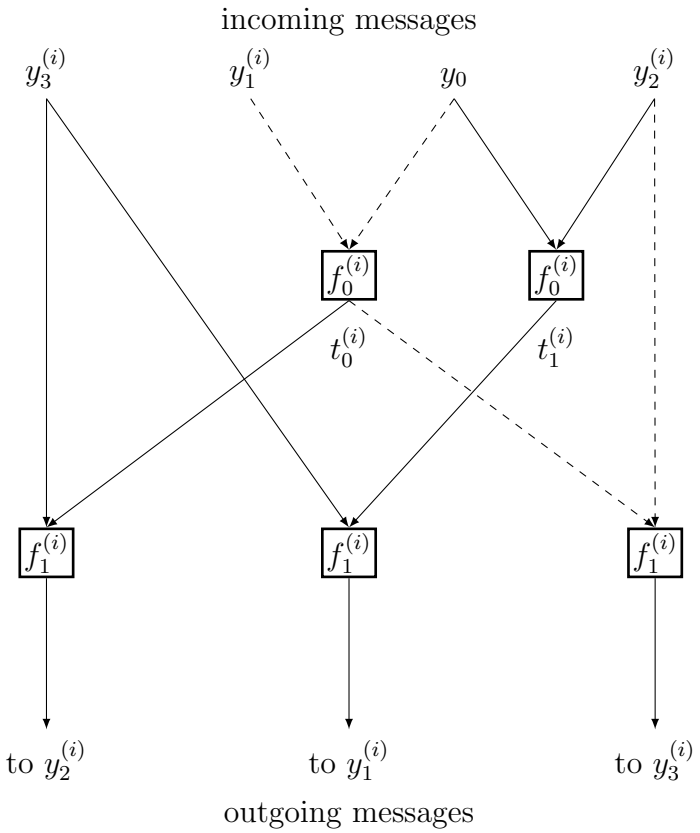


Figure 6.4: Message generation of a degree $d_\mathrm{v} = 3$ variable node which utilizes reuse of intermediate results.

Of course, also for the variable nodes a decomposition into two-input lookup tables which results in maximum reuse potential and a minimum number of distinct required lookup tables should be chosen. For the relatively small variable node degree $d_\mathrm{v} = 3$, however, no more favorable decomposition than the one originally proposed by Kurkoski *et al.* in [28] and discussed in Section 5.2.2 could be found. Also this decomposition enables to reuse one intermediate result $t_0^{(i)}$ when considering message generation for all edges connected to a degree $d_\mathrm{v} = 3$ variable node, as it is depicted in Figure 6.4. As shown there, it requires $M = 2$ different vectors $\mathbf{lut}_m^{(i)}$ to implement $f_0^{(i)}$ and $f_1^{(i)}$. All $d_\mathrm{v} = 3$ outgoing messages are determined using five two-input operations. Please note that another lookup table is required to perform the final bit decision which is not depicted in Figure 6.4. Again, the shown decomposition can also be applied for the variable nodes in a state-of-the art decoder which uses the addition operation of LLRs at the variable nodes.

6.3 Results and Discussion

This section presents the practical results from the DSP implementation of Information Bottleneck decoders and state-of-the-art decoders on the C6474. The conventional decoding algorithms investigated here used fixed point arithmetic. Recall that the range of a signed $Qa.b$ fixed point number is from $-2^{(a-1)}$ to $+2^{(a-1)} - 2^{-b}$ with a uniform precision of 2^{-b}. In order to not disadvantage the conventional decoding algorithms by a bad choice of parameters, several bit widths and fixed point formats were tested on the device to find the parameters which offer best performance. Finally, eight bit fixed point number representation with four bits designated to the fractional part was used for the conventional decoders corresponding to the $Q4.4$ fixed point format.

It was found that using a larger bit width, for example, 16 bit fixed point arithmetic, did not improve the decoding performance of the conventional decoders, but slowed down their decoding significantly.

In the following, the memory requirements of the implemented decoders are compared. Afterwards, their bit error rate performance on the DSP is investigated. Finally, the net decoding throughputs of the distinct decoders on the C6474 will be compared.

6.3.1 Comparison of the Memory Requirements

First, the memory requirements of the distinct decoders on the DSP platform are investigated. To do so, Table 6.1 lists the memory amounts required for the components of the implemented decoders. The upper part of the table lists the memory demands required to store the graph structure of the parity-check matrices of the applied codes *regular code a)* and *regular code b)*. These memory demands are identical for all implemented decoders and only depend on the code. They are determined by the number of edges in the respective Tanner graph of the parity-check matrix which is identical to the number of ones in this matrix.

The center part of Table 6.1 shows the memory amounts required for the lookup tables used to implement the node opertations of the distinct decoders. As it can be seen there, the min-sum decoder does not use a lookup table at all, resulting in a total size of 0 kilobyte. The belief propagation decoder uses a

Table 6.1: Comparison of memory requirements of distinct decoders

Decoder	Code	Memory for graph structure
all	*regular code a)*	48.0 kilobytes
all	*regular code b)*	15.84 kilobytes

Decoder	Memory for lookup tables
min-sum decoder	0 kilobyte
belief propagation decoder	0.022 kilobyte
Information Bottleneck, lookup table reduction	38.4 kilobytes
Information Bottleneck, lookup table expansion	153.6 kilobytes

Code	Decoder	Memory for messages
a)	min-sum decoder	32.0 kilobytes
a)	belief propagation decoder	32.0 kilobytes
a)	Information Bottleneck	16.0 kilobytes
b)	min-sum decoder	10.56 kilobytes
b)	belief propagation decoder	10.56 kilobytes
b)	Information Bottleneck	5.28 kilobytes

notation: 1.0 kilobyte = 1,000 bytes

very small lookup table with a size of only 22 bytes for implementation of $f_c(x)$ from Equation (2.55). The size was chosen that small because it turned out that a total of 22 lookup table entries was sufficient to achieve the best possible bit error rate performance of the fixed point belief propagation decoder with the chosen eight bit fixed point format.

The Information Bottleneck decoders naturally need more memory to store the lookup tables characterizing all two-input operations in all decoder iterations. Both variants, the one with lookup table reduction and with lookup table expansion, require several kilobytes of memory to store the lookup tables.

Finally, the bottom part of Table 6.1 compares the memory demands required to store the messages exchanged in the distinct decoders. Here the Information Bottleneck LDPC decoders can profit from a shorter $q = 4$ bit representation of the messages which halves this memory demand in comparison to the state-of-the-art decoders. Profiting from the shorter $q = 4$ bit width representation is possible in the DSP implementation by storing several $q = 4$

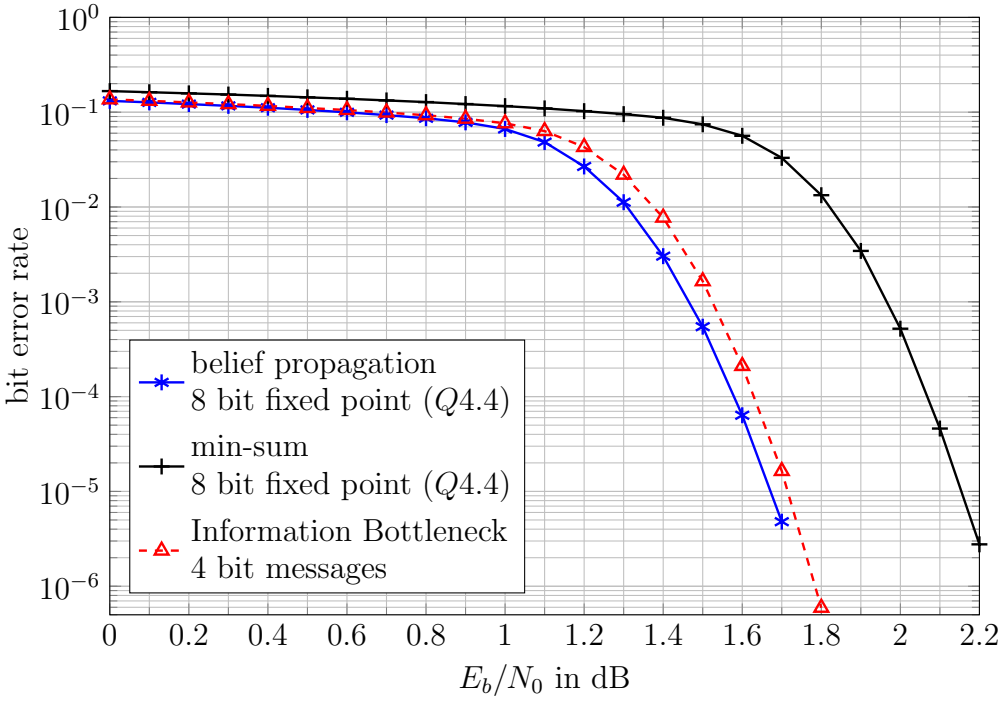
bit integers in a longer base data type, as it has been explained at the end of Section 6.2.2.

In summary, Information Bottleneck decoders require larger memory amounts than the conventional decoders. Anyway, they are still moderate. For example, a fixed point belief propagation decoder for *regular code a)* requires around 80 kilobytes of memory, while an Information Bottleneck decoder with lookup table reduction needs 102.4 kilobytes. When lookup table expansion is used, the Information Bottleneck decoder needs 217.6 kilobytes. However, we will see that lookup table expansion offers considerable gains in the decoding throughput. The investigation above tells that all decoders easily fit into the memory of the applied platform.

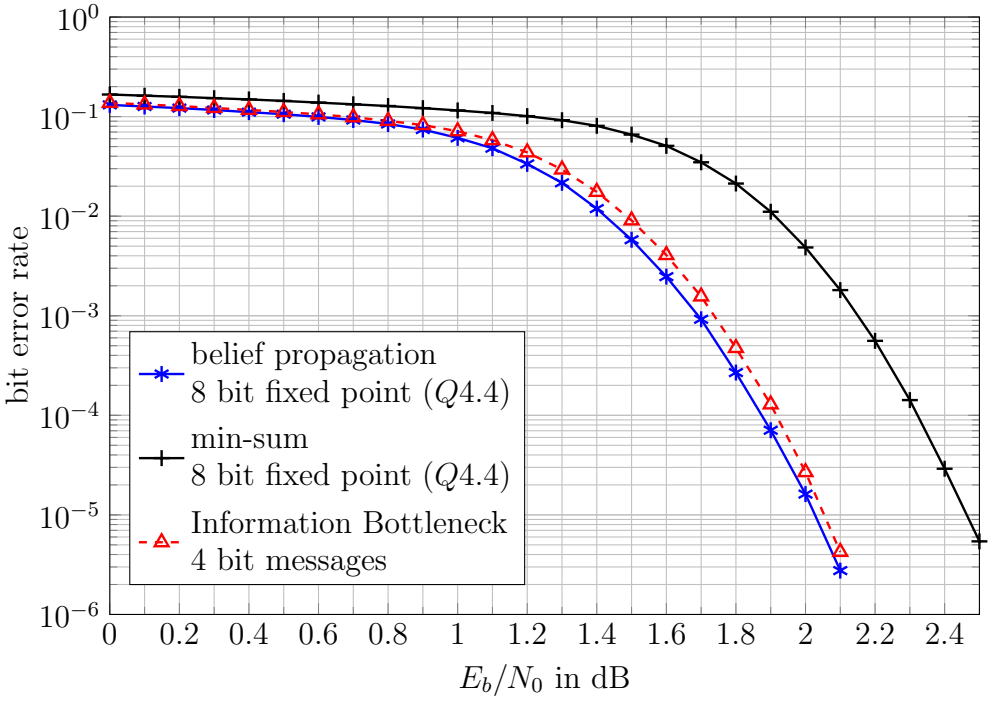
6.3.2 Comparison of the Bit Error Rate Performances

In order to evaluate the bit error rate performances of the different decoders, LDPC encoded transmission schemes over an AWGN channel using *regular code a)* and *regular code b)* with BPSK modulation were implemented on the physical layer. At the receiving end, the received signal was quantized using a $q = 4$ bit Information Bottleneck quantizer designed as described in Section 5.1.2. For the conventional decoders, LLRs were calculated in fixed point precision and passed to the decoder. The applied Information Bottleneck decoder directly processed the quantization indices from the channel output quantizer. All decoders performed a maximum of $i_{\max} = 50$ decoding iterations or stopped decoding as soon as all their parity-checks were satisfied. Figure 6.5 shows the bit error rates of the implemented fixed point belief propagation decoder, an Information Bottleneck decoder and a min-sum decoder on the DSP. The upper part of the Figure corresponds to *regular code a)* and the lower part corresponds to *regular code b)*. Please note that the bit error rate performance of the Information Bottleneck decoder is independent of its respective lookup table implementation, that is, lookup table reduction or lookup table expansion described in Section 6.2.2 and Section 6.2.3.

The effects that the Information Bottleneck LDPC decoder outperforms the min-sum decoder and comes very close to the performance of the belief propagation decoder that could be observed in the simulations from Section 5.2.5 are also validated in the fixed point DSP implementations of the respective



(a) Bit error rate of *regular code a)* for with $i_{\max} = 50$ iterations.



(b) Bit error rate of *regular code b)* with $i_{\max} = 50$ iterations.

Figure 6.5: Comparison of the bit error rate performance of various decoders on the C6474 DSP.

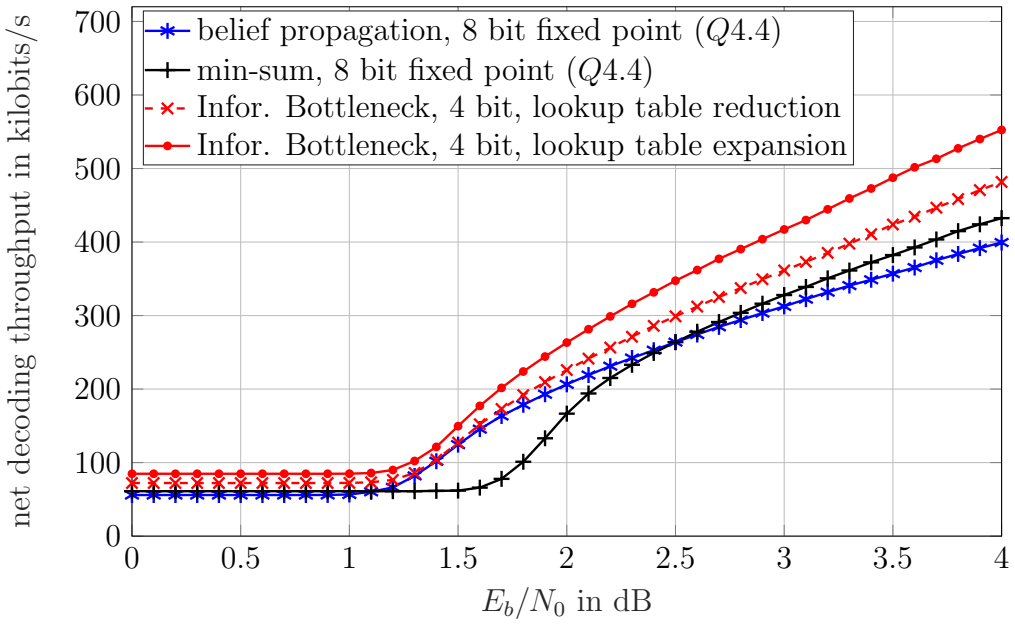
decoders. As the most important result, the Information Bottleneck decoder offers almost identical performance to the belief propagation decoder and it outperforms the min-sum decoder.

6.3.3 Comparison of the Net Decoding Throughputs

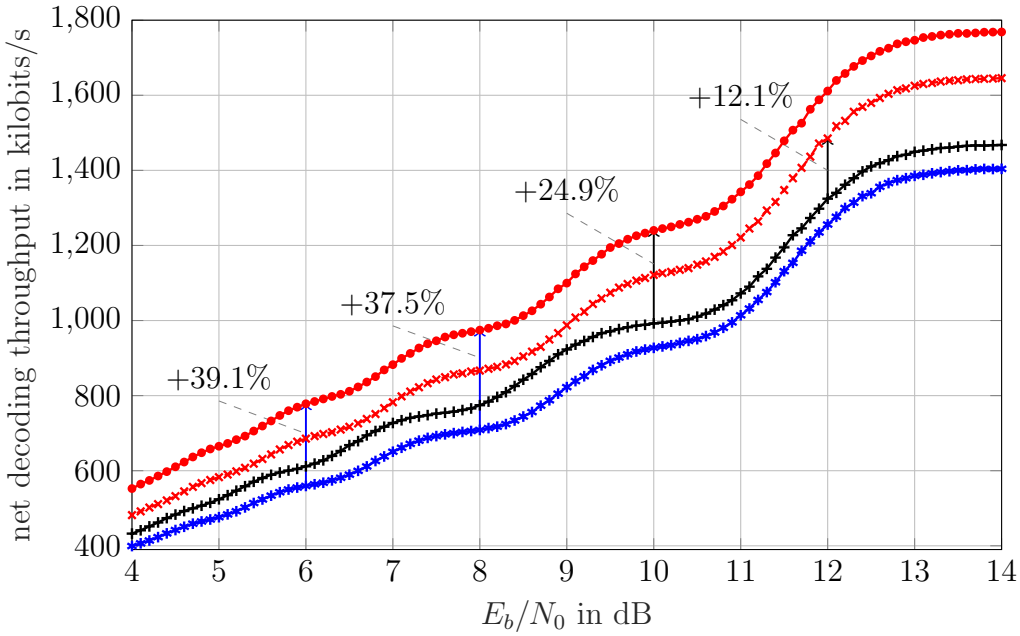
The results from Section 6.3.2 motivate application of Information Bottleneck decoders from a performance perspective because they offer error correction capability very close to that of a belief propagation decoder and they outperform the min-sum decoder. The question remaining is if Information Bottleneck decoders offer higher or lower net decoding throughput on the C6474. In order to determine the net decoding throughput, the average number of decoded information bits per second has been measured on the DSP for $E_b/N_0 \in [0; 14]$ dB.

Figures 6.6 and 6.7 show the results for *regular code a)* and *regular code b)*, respectively. For both codes the E_b/N_0 axis has been split into two parts from 0 to 4 dB shown in the upper part of the respective figure and from 4 dB to 14 dB in the lower part. This was done to enlarge the curves in the low E_b/N_0 regime, where they show some interesting behaviour. All shown curves rise, as all decoders tend to perform fewer iterations on average as the E_b/N_0 increases. For very high E_b/N_0, the decoding normally terminates after the initial parity-check, as errors in the received word are very unlikely. As a result, the curves saturate on the right of Figures 6.6b and 6.7b and tell the maximum achievable throughputs.

The main result holding for both codes is that both variants of the Information Bottleneck LDPC decoders offer the highest net decoding throughputs among all implemented decoders over the complete shown range of E_b/N_0. This is especially important when recalling the bit error rate performances of the fixed point belief propagation decoder and the Information Bottleneck decoder over E_b/N_0 shown in Figure 6.5. While the bit error rates of both decoders are almost identical over E_b/N_0, the Information Bottleneck decoders run significantly faster on the DSP. The lookup table expansion technique offers the highest decoding throughput at the expense of larger lookup tables. Interestingly, there is an E_b/N_0 interval from approximately 1.1 dB to 1.5 dB for both codes, where the fixed point belief propagation decoder comes close

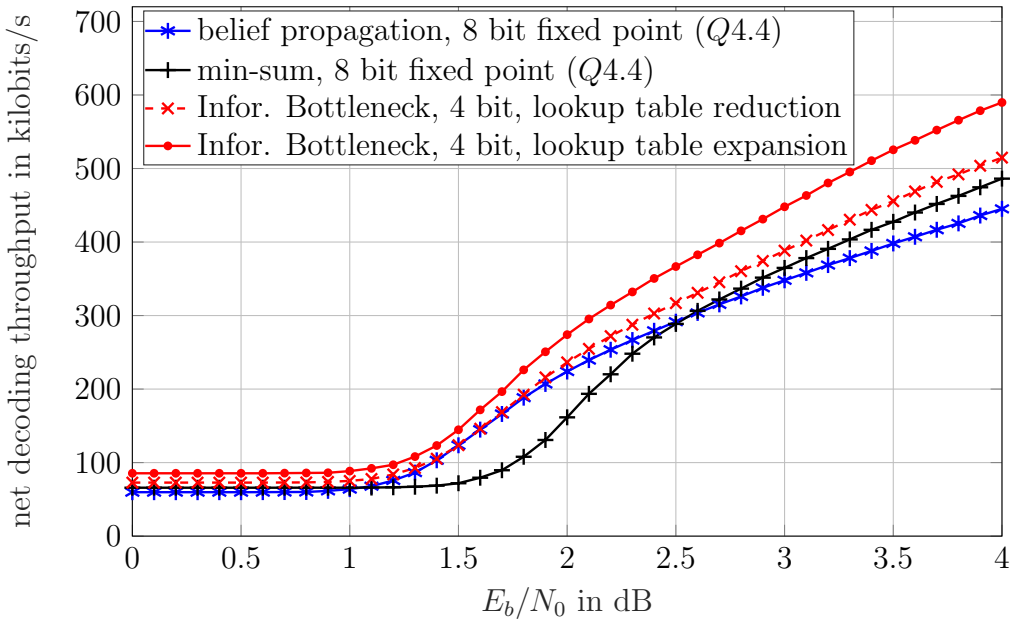


(a) Comparison of net decoding throughputs on the C6474 DSP.

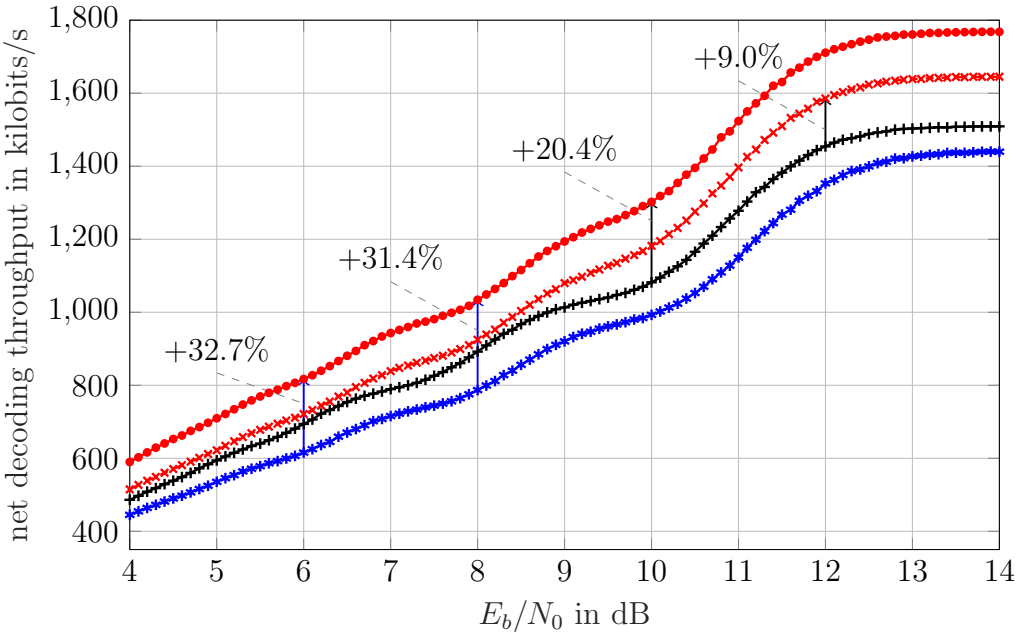


(b) Comparison of net decoding throughputs on the C6474 DSP.

Figure 6.6: Net decoding throughputs of distinct LDPC decoders on the C6474 for *regular code a)*. Information Bottleneck decoders use $q = 4$ bit messages. The conventional decoders use 8 bit fixed point arithmetic with 4 bits allocated to the fractional part.



(a) Comparison of net decoding throughputs on the C6474 DSP.



(b) Comparison of net decoding throughputs on the C6474 DSP.

Figure 6.7: Net decoding throughputs of distinct LDPC decoders on the C6474 for *regular code b)*. Information Bottleneck decoders use $q = 4$ bit messages. The conventional decoders use 8 bit fixed point arithmetic with 4 bits allocated to the fractional part.

Table 6.2: Net decoding throughputs on the C6474 DSP

Net decoding throughput, *regular code a)* in kilobits/s:

E_b/N_0 in dB	0	4	8	12
belief propagation decoder	55.9	399.3	708.5	1255.7
Information Bottleneck, lookup table reduction	72.2	481.7	866.7	1485.1
Information Bottleneck, lookup table expansion	84.8	552.5	974.5	1611.4
min-sum decoder	61.1	432.5	774.5	1324.5
gain lookup table exp. w.r.t. belief propagation	51.7%	38.4%	37.5%	28.3%
gain lookup table expansion w.r.t. min-sum	38.8%	27.7%	25.8%	21.7%

Net decoding throughput, *regular code b)* in kilobits/s:

E_b/N_0 in dB	0	4	8	12
belief propagation decoder	59.8	445.1	786.8	1352.9
Information Bottleneck, lookup table reduction	72.8	514.8	924.7	1585.5
Information Bottleneck, lookup table expansion	85.5	589.8	1034.2	1711.0
min-sum decoder	65.8	486.2	892.2	1454.2
gain lookup table exp. w.r.t. belief propagation	43.0%	32.5%	31.4%	26.5%
gain lookup table expansion w.r.t. min-sum	29.9%	21.3%	15.9%	17.7%

to the Information Bottleneck decoder with lookup table reduction. This effect is caused by a slightly smaller number of iterations of the conventional decoder that have to be performed in this interval on average due to its slighty better error correction performance (cf. Figure 6.5). Anyway, both curves for the Information Bottleneck decoders always lie above the ones for the belief propagation decoder.

Concerning the state-of-the-art algorithms, it is very interesting that the curves for the belief propagation decoder and the min-sum decoder intersect in Figures 6.6a and 6.7a. This is due to the fact that the belief propagation decoder always requires extra resources to evaluate its lookup table for $f_c(x)$, but the gains in terms of the required number of iterations in comparison to the simpler min-sum operation become smaller with increasing E_b/N_0.

To quantify the throughput gains of the Information Bottleneck decoders they are added in percent values between the curves referring to the distinct decoders in Figures 6.6b and 6.7b for exemplary E_b/N_0. Moreover, Table 6.2

shows some exemplary data points from the curves in Figures 6.6 and 6.7. The upper part of this table corresponds to *regular code a)* and the lower part corresponds to *regular code b)*. At the bottom of each part, the gains of the fastest Information Bottleneck decoder with lookup table expansion with respect to the belief propagation decoder and the min-sum decoder are listed.

6.4 Summary

This chapter investigated and quantified potential gains of Information Bottleneck signal processing in comparsion to state-of-the-art signal processing in a specific implementation on a real hardware platform. In particular, a belief propagation decoder, a min-sum decoder and Information Bottleneck decoders were implemented on a DSP and compared comprehensively. First, the bit error rate performance of Information Bottleneck decoders in comparison to the considered state-of-the-art decoding algorithms could be verified on the DSP platform, showing that Information Bottleneck decoders which offer almost the same bit error rate performance as belief propagation decoders and which outperform min-sum decoders can be built on a DSP in practice.

By analyzing the memory requirements of the distinct decoders, it was shown that the memory amounts required to implement Information Bottleneck LDPC decoders are reasonable and available on a typical SDR platform.

A specific matter about DSP development is that one always has to carefully map the signal processing algorithms to the available platform resources. As a consequence, several ideas for the lookup table implementation on a DSP platform have been developed and presented, resulting in techniques called lookup table reduction and lookup table expansion. The practical results have shown that while lookup table reduction allows to reduce the memory footprint of an Information Bottleneck decoder, lookup table expansion provides significant gains in net decoding throughput at the expense of larger arrays in the memory. Application of these techniques, therefore, allows to flexibly trade lookup table memory for decoding speed in an Information Bottleneck decoder. Furthermore, it has been explained how specific two-input decompositions of the node operations in an LDPC decoder allow to profit from reusing intermediate results for several edges connected to a node in the decoding process. This

technique is applicable to Information Bottleneck LDPC decoders and to the state-of-the-art decoders.

The provided investigation of the net decoding throughputs on the DSP showed that Information Bottleneck decoders offered the highest net decoding throughputs among all implemented decoders. Information Bottleneck decoders with lookup table expansion by far outperformed the competitors in terms of net decoding throughput. This is particularly important when recalling that the bit error rate performances of the Information Bottleneck decoders are better than the one of the min-sum decoder. Therefore, the Information Bottleneck decoders beat this reference decoder in terms of both, error correction performance and net decoding throughput. They also beat the belief propagation decoder in terms of decoding throughput while offering almost identical bit error correction capabilities.

From the results presented in this chapter, one can draw the conclusion that Information Bottleneck LDPC decoders are very intersting candidates for SDR applications which require to implement the channel decoder on a DSP.

Chapter 7

The Information Bottleneck Method as a General Concept for Receiver Design

Chapter 5 illustrated that the Information Bottleneck method can be used to design components of receiver signal processing chains using many practical examples. These examples reached from quantizer design over the decoding of binary LDPC codes under various modulation schemes to channel estimation and detection components. Specific advantages of the resulting receiver components in comparison to state-of-the-art receivers are that they only require low bit widths in their implementation and, moreover, replace all complex arithmetical signal processing operations by simple lookup operations. Hence, while achieving excellent performance, virtually the same as double precision benchmark algorithms, developing a receiver with the Information Bottleneck method inherently tackles two important bottlenecks in the implementation of modern digital receivers at the same time.

Chapter 6 has exemplarily revealed specific advantages resulting from the Information Bottleneck design in comparison to state-of-the-art signal processing approaches in an implementation on a DSP based signal processing platform.

This chapter aims to summarize some insights which could be gained from the application of the Information Bottleneck method to various receiver-sided signal processing problems carried out so far. Moreover, it discusses advantages of the proposed Information Bottleneck receiver design and summarizes some of its remaining challenges.

7.1 Extracting Relevant Information from the Received Signal

A very general view on the Information Bottleneck approach to receiver design is illustrated in Figure 7.1. The transmitter on the left encodes and modulates user data x to obtain the transmitted signal s for transmission over a physical transmission channel. At the output of the channel, the receiver has access to the observed realization $\tilde{y}$ of the received signal. The vital task of a digital receiver now essentially is to extract *relevant information* on the transmitted data x from the received signal $\tilde{y}$, thus motivating to use the Information Bottleneck method for receiver design. Doing so inherently leads from a signal

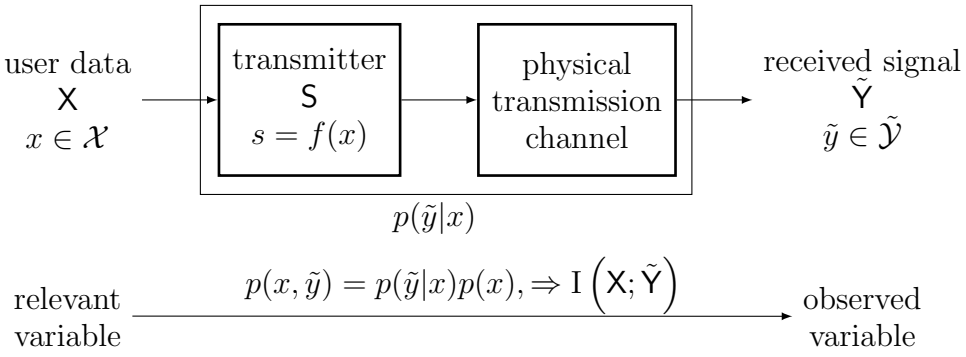


Figure 7.1: General view on the Information Bottleneck problem of building a receiver chain. The receiver has to extract information on the user data from the received signal.

processing perspective to an information processing perspective on the detection and decoding problems of a communication receiver.

In order to apply the Information Bottleneck method, one essentially only needs to determine the joint probability distribution $p(x, \tilde{y})$ of the relevant random variable X and the observed variable $\tilde{\mathsf{Y}}$. The distribution $p(x, \tilde{y})$ is fully determined by the source statistics $p(x)$ and the transition probability $p(\tilde{y}|x)$ of the effective transmission channel. In theory, one just has to feed this distribution to an Information Bottleneck algorithm to construct a receiver which delivers an output that is highly informative about the user data x.

The first burden, however, is that the received signal $\tilde{\mathsf{Y}}$ from a physical transmission channel typically is continuous. As a result, the design of a scalar quantizer for the received signal which maps the continuous received samples onto a discrete representation normally has to be the first step of receiver design with the Information Bottleneck method. After application of a channel output quantizer, the effective channel has a discrete output with outputs that can simply be enumerated using unsigned integers. From this point, one has to figure out a meaningful way to extract information on X from the outputs Y of the channel output quantizer in a receiver chain.

Unfortunately, channel coding and modulation applied at the transmitter have to spread the information on a single bit in the user data over a large block of transmitted symbols to protect the transmitted data against transmission errors on the physical channel. As a result, X, S and Y are multivariate variables with very large dimensionalities in practice. A simple and straightfor-

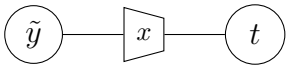
ward application of an Information Bottleneck algorithm on $p(x, y)$ to design a digital receiver is, therefore, not feasible in most cases because the cardinalities of the variables involved are too large. In contrast, one has to split up the problem of extracting information on X at the receiving end into channel decoding, detection and estimation tasks performed by concatenated signal processing blocks which exchange messages, just as it is typically done in a conventional receiver.

A special characteristic of a receiver which is entirely made of blocks constructed with the Information Bottleneck method is that it has no need to represent the quantized signals or the messages exchanged using real or complex representation values. Instead, it can just use unsigned integer indices for this purpose. The reason is that all its concatenated lookup tables are designed with the knowledge of the probability distributions involved and that these distributions determine the inherent mutual information.

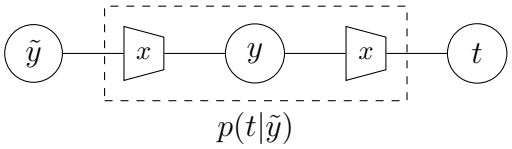
7.2 Modelling Receiver Chains in Information Bottleneck Graphs

Designing a signal processing chain using the Information Bottleneck method is often a very complex task which typically involves various estimation, detection and decoding steps. In the receiver design with the Information Bottleneck method these steps correspond to designing concatenated Information Bottleneck nodes which locally aim to maximize their relevant information and to concatenate them in a meaningful way. Hence, one has to design a *flow of relevant information* through the signal processing entities. To do so, one needs a tool which allows to efficiently describe the information relations of various variables involved in the receiver processing. Information Bottleneck graphs were found to be particularly useful in the design of such complex receiver chains, as they express very compactly which variable compresses which and which variable shall be informative about which other.

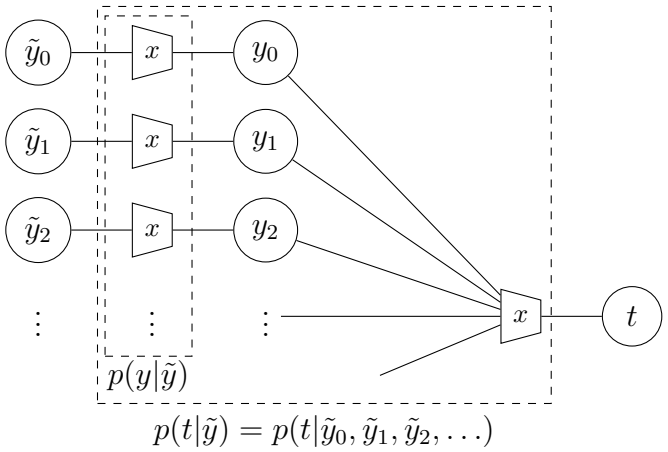
To recall this feature of Information Bottleneck graphs note that the problem of extracting relevant information on the user data x from the received signal $\tilde{y}$ which has just been considered in the previous section can be described very compactly using an Information Bottleneck graph, as it is illustrated in



(a) Very high abstraction level model of an Information Bottleneck receiver.



(b) More detailed model of an Information Bottleneck receiver with an opened node.



(c) Even more detailed model of an Information Bottleneck receiver.

Figure 7.2: Simple example for the usage of Information Bottleneck graphs in receiver design. Information Bottleneck graphs allow to adapt the abstraction level of receiver components easily by opening and closing nodes.

Figure 7.2a. Moreover, Information Bottleneck graphs allow for hierarchical modelling by opening and closing Information Bottleneck nodes. This feature enables to easily adapt the abstraction level of a considered receiver model. It is, for example, very easy to include a quantizer $p(y|\tilde{y})$ which maps $\tilde{y}$ onto y in this model by opening the node $p(t|\tilde{y})$, as it is illustrated underneath in Figure 7.2b. Please note that, if $\tilde{y} = (\tilde{y}_0, \tilde{y}_1, \tilde{y}_2, \ldots)$ is multivariate, the quantization node could again be opened, for example, into a decomposed Information Bottleneck node which consists of many scalar quantizers internally, as it is shown exemplarily in Figure 7.2c. The node $p(t|y_0, y_1, y_2, \ldots)$ in this example could again be opened, for example to decompose it into Information Bottleneck

nodes that perform detection, channel estimation and channel decoding. Like this, one can dive deeper and deeper into the design of local signal processing components with the Information Bottleneck method in a systematic manner. With this ability to model complex information relations between huge numbers of variables hierarchically, Information Bottleneck graphs are a key to Information Bottleneck based receiver modelling and receiver design.

7.3 The Role of Message Alignment

In an Information Bottleneck graph which consists of concatenated Information Bottleneck nodes, typically output joint distributions $p(x,t)$ of the Information Bottleneck algorithms applied to the node design serve as inputs to the Information Bottleneck algorithms applied for the design of other concatenated nodes. Since in a receiver chain, one can deterministically resolve all connections between the nodes involved, the design of any concatenated scheme of Information Bottleneck nodes should be possible in theory, if one is able to obtain all joint probability distributions $p(x,y)$ to design all included nodes.

In some applications, however, modelling and tracking the particular connections between all nodes involved and their respective associated joint probility distributions has prohibitive complexity. This problem appears, if the number of nodes involved is very large and their connections are quasi random, as it has been observed, for example, in the Tanner graph structure of an irregular LDPC code. The problem is also visualized in a generalized manner in Figure 7.3 which describes the quasi random graph structure using the Π symbol. In such scenarios, the joint output distributions $p(x,t_n)$ of the Information Bottleneck algorithms applied to design the Information Bottleneck nodes $p(t_n|y_n)$ of different nodes with an identifying index n on the left of the figure depend on some local configuration of the particular nodes involved. As a result, a receiving node of an unsigned integer message t on the right of the figure cannot resolve the originally intended meaning $p(x|t_n)$ of this integer message for the connected relevant random variable X without knowing the particular index n and the corresponding $p(x|t_n)$.

It was shown that in such situations, message alignment helps to determine a common *language* spoken by the different, independently designed Infor-

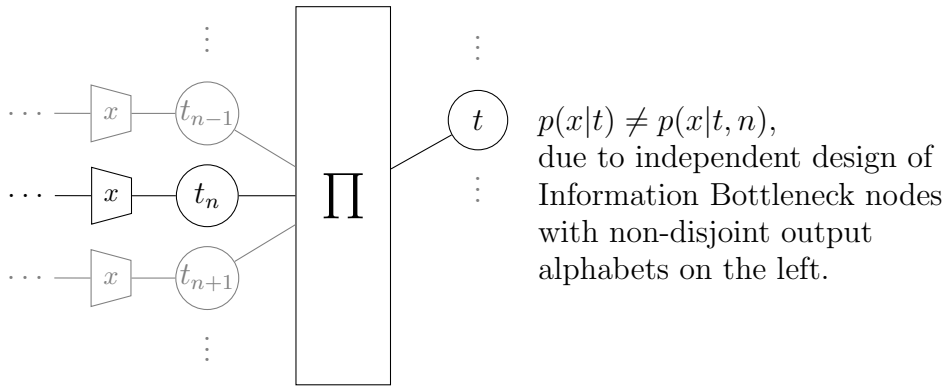


Figure 7.3: General illustration of the message alignment problem. Due to the independent design of the Information Bottleneck nodes on the left, the distribution $p(x|t,n)$ depends on the properties of the transmitting node with index n on the left, but the index n cannot be resolved on the right.

mation Bottleneck nodes $p(t_n|y_n)$ in the Information Bottleneck graph. With message alignment, the Information Bottleneck nodes do not pass their output t_n directly into a quasi random graph. Instead, they determine novel, *aligned* integer messages z_n to be passed into a quasi random graph. The unknown recipient just receives an integer z which could originate from any node. The key characteristic is now that $p(x|z)$ is possibly independent of the particular local configuration of the transmitting node n, such that $p(x|z) \approx p(x|z,n)$. This avoids an undesired ambiguity in the meanings of a particular received integer message z. The presented message alignment algorithm can be used to design mappings $p(z_n|t_n)$ which achieve this aim.

7.4 Relations to the Sum-Product Algorithm

An interesting question remaining is how the integer based message passing in an Information Bottleneck graph is related to the message passing in the sum-product algorithm. This connection was discussed in [70, 76] in the scope of this thesis.

In order to explain it, the very simple message passing algorithm from Example 2.4.3.1 is revisited and related to integer based message passing in the corresponding Information Bottleneck graph here. The factor graph shown at top of Figure 7.4 is a simplified variant of the factor graph already studied in

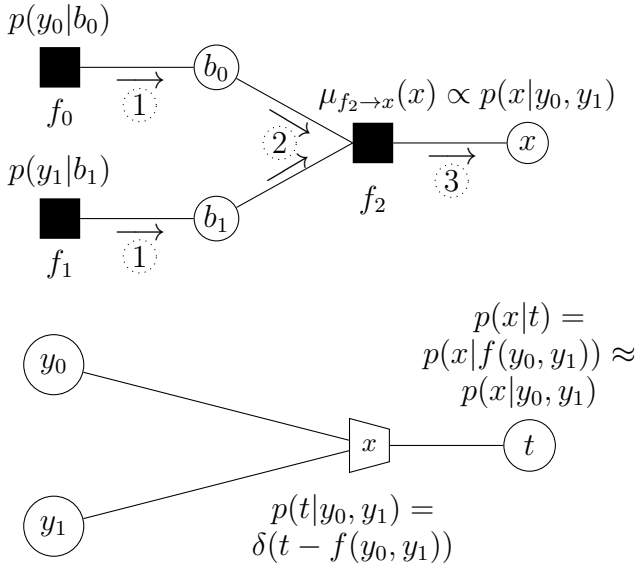


Figure 7.4: Comparison of message passing in a factor graph and an Information Bottleneck graph.

Example 2.4.3.1 which uses that $p(b_0) = p(b_1) = 1/2$. These constant factors can be dismissed in the message passing algorithm, as they cannot contribute any useful information. As a result, a simplified message passing algorithm to generate $p(x|y_0, y_1)$ is also visualized in the factor graph in Figure 7.4. For some given evidence y_0 and y_1, the algorithm starts at the leaf nodes f_0 and f_1 by sending $\mu_{f_0\to b_0}(b_0) \propto p(y_0|b_0)$ and $\mu_{f_1\to b_1}(b_1) \propto p(y_1|b_1)$. The degree two variable nodes b_0 and b_1 just pass these incoming messages directly towards node f_2. This node performs its update exactly as it has been explained in Example 2.4.3.1 and the key is that its outgoing message $\mu_{f_2\to x}(x)$ is proportional to the desired posterior distribution $p(x|y_0, y_1)$. As a result, one ends up with a belief on $x = b_0 \oplus b_1$ from evidence (y_0, y_1).

Underneath the factor graph in Figure 7.4, an elementary Information Bottleneck graph showing the operation performed at the check nodes in an LDPC decoder is shown. The shown Information Bottleneck node processes two inputs y_0, y_1 and delivers an integer t. However, the distribution $p(x|t)$ is known from the design process of the node. As a result, for a fixed configuration (y_0, y_1) of the inputs of the node, one also ends up with a belief on $x = b_0 \oplus b_1$, namely $p(x|\mathsf{T} = t)$. A key observation here is, that there are typically fewer possibilities for t than there are input combinations (y_0, y_1) of the node due

to the inherent compression of the node. Moreover, t is a function of (y_0, y_1), that is, $t = f(y_0, y_1)$. As a result, the belief $p(x|t) = p(x|f(y_0, y_1))$. Trivially, equality $p(x|f(y_0, y_1)) = p(x|y_0, y_1)$ can always be achieved if $|\mathcal{T}| = |\mathcal{Y}_0| \cdot |\mathcal{Y}_1|$, but in this case the node achieves no compression. The much more interesting case is $|\mathcal{T}| << |\mathcal{Y}_0| \cdot |\mathcal{Y}_1|$ which involves compression. In this case, $p(x|t) = p(x|f(y_0, y_1)) \approx p(x|y_0, y_1)$. This approximation is designed according to rules which preserve the relevant information $\mathrm{I}(\mathsf{X}; \mathsf{T})$ with the Information Bottleneck method.

Following this argumentation, one could certainly understand the message passing process in an Information Bottleneck graph as a sort of an approximated variant of the sum-product algorithm. It is, however, noteworthy that despite one can understand the connection between both message passing algorithms by considering that each integer message t has a belief $p(x|t)$ associated, processing this belief is only required to construct concatenated node structures and not during the actual message passing process. The reason for this is that in an Information Bottleneck graph, concatenated nodes are inherently matched to the output integers of their ancestors because one has to process the output distributions of the ancestor nodes to design the following nodes, as it has been done very often in this thesis.

The final consequence of this procedure is, that once an Information Bottleneck graph has been designed, the complete message passing algorithm collapses to the passing of integers and performing lookup operations at the factor nodes, without the need to represent any beliefs using real numbers.

7.5 Implementation Aspects

The information centric design of Information Bottleneck receivers aims to maximize the preserved relevant information in each Information Bottleneck node for a given output bit width of this node. This inherently yields a very compact, but highly informative signal representation in the form of unsigned integers which have to be passed between the concatenated Information Bottleneck nodes.

In a dedicated chip implementation of a receiver, this bit width determines the number of wires required to communicate between two interconnected

blocks over a bit parallel bus. In a bit serial architecture, it determines the required clock frequency of the serial bus used to transport the bits which represent the integers exchanged. These parameters influence the routing complexity, the chip area and the power consumption of a receiver implementation on a dedicated chip. In a DSP implementation of a receiver, the bit width influences the memory footprint of the receiver. As a result, minimizing the bit width used for the signal representation with a desired minimum loss of relevant information is a very rational design aim which can be achieved with the Information Bottleneck method in a very generic and natural way.

The Information Bottleneck design of a receiver yields a list of static lookup tables which completely characterize the implementation of the receiver. The DSP decoder implementations from Chapter 6 revealed that very simple addressing operations inside these lookup tables offer significant throughput gains in comparison to state-of-the-art signal processing techniques.

The design of the lookup tables to be used in a receiver with the Information Bottleneck method is admittedly very complex and requires sophisticated techniques to track and process output distributions of Information Bottleneck algorithms. This fact would normally question the entire concept of designing receiver components with the Information Bottleneck method because the involved probability distributions depend on channel parameters like the E_b/N_0 and the on-the-fly generation of the lookup tables is too complex. Interestingly, however, the resulting receiver components offer remarkable performance even with mismatched and static lookup tables. As a result, the lookup tables only have to be constructed once and for a fixed design-E_b/N_0. Hence, despite the design of the tables might be very complex, it can be carried out only once and offline. The actual resulting receiver implementation then has very low complexity, as it only uses lookups in static tables of unsigned integers.

7.6 Remaining Challenges

Despite the receiver design with the Information Bottleneck method is a promising alternative to the common signal processing approaches also some remaining challenges have been revealed so far.

The Information Bottleneck algorithms presented in Section 3.2 process the joint probability distribution $p(x, y)$ and their complexity is mainly determined by the cardinalities of X and Y. If these cardinalities are huge, the Information Bottleneck algorithms require a very long runtime to generate the lookup tables. Moreover, the resulting lookup table which characterizes $p(t|y)$ has $|\mathcal{Y}|$ entries. If this number is large, also the resulting lookup table is large. The construction and the memory complexity of the resulting lookup tables in fact limits the possibility to construct arbitrary receiver components with the available Information Bottleneck algorithms. One of the reasons is that some signal processing entities have to process a huge number of input variables. This huge number of input variables results in a huge number of possible input configurations which all have to be addressed in a lookup table implementation. Efficient implementation and generation of arbitrary relevant information preserving mappings $p(t|y)$ is an interesting field for future work on the topic. Moreover, constructing *globally* optimum mappings for some specific applications could be an interesting focus for further research.

In many investigated applications it was shown that it is possible to reduce the construction and also the implementation complexity of an Information Bottleneck node with many input variables by using the factor graph principle of opening the node and designing smaller component lookup tables which all process fewer inputs inside the opened node. While having enabled the design of many complex Information Bottleneck nodes, this process, however, also raised a lot of questions. First, more than one decomposition of an opened node in an Information Bottleneck graph exists in many cases. It is not trivial to answer the question what a favorable or even an optimum node decomposition of a complex Information Bottleneck node looks like.

Second, if a node is opened, one might have to decide for the local relevant random variables of many concatenated nodes. This was the case, for example, in Section 5.5, where a receiver structure has been designed which performs channel estimation, detection and decoding using only Information Bottleneck nodes. The choice of local relevant random variables has been made intuitively in Section 5.5, mainly motivated from the aim of enabling a logical *flow of relevant information* in the Information Bottleneck graph of the receiver. However, so far no rules to design such a flow are known. Moreover,

each and every single Information Bottleneck node inside an opened node can use a different output cardinality in general. This choice leaves a tremendous number of degrees of freedom if the number of Information Bottleneck nodes in the graph is large.

For all problems investigated in Chapter 5 it was possible to construct mismatched and static lookup tables which could compete with the performance of double precision signal processing components in the receiver. An open question still is, why the performance of the mismatched Information Bottleneck nodes is that good. Intuitively, one would expect the opposite because the mismatched components face other probability distributions during operation than for their design. Also knowing a general rule to identify design parameters like the design-E_b/N_0 which offer good performance, if the resulting lookup tables are actually used mismatched would be desirable.

Chapter 8

Conclusion

In this thesis, the Information Bottleneck method has been applied to solve various quantized signal processing problems that appear in communication receivers. The motivation for this was to vividly use the Information Bottleneck method to extract *relevant information* in the signal processing algorithms in order to build innovative receiver chains which could help to overcome hardware bottlenecks.

The simplest, yet very important use case of the Information Bottleneck method that has been investigated was the construction of scalar channel output quantizers which aim to preserve a maximum possible amount of relevant information for a given quantization bit width. A key difference to state-of-the-art signal processing was that the considered quantizers only delivered unsigned integer indices which can be stored using only a few bits in the hardware, instead of any real valued representation values. From an information theoretical perspective, the assignment of representation values from the domain of the input variable to the quantization regions is simply not necessary, as it cannot increase the preserved relevant information. The presented results on quantizer design with the Information Bottleneck method showed that it is often favorable in terms of preservation of relevant information in comparison to MMSE quantization with the Lloyd-Max algorithm.

After having constructed quantizers which just output plain integer indices instead of any real representation values, we were facing the problem that conventional algorithms for decoding of LDPC codes simply cannot be used without going back to the LLR domain. Since this would foil the main advantage of compact signal representation using unsigned integers, a novel LDPC decoding algorithm was developed with the Information Bottleneck method afterwards. The idea here was to use an Information Bottleneck algorithm to construct lookup tables which replace the classical node operations in a message passing decoder for LDPC codes. The design aim of these lookup tables was to preserve the maximum possible amount of relevant information on the codeword bit an outgoing message of a node represents. In order to determine the required input distributions of the Information Bottleneck algorithms applied to node design, a discretized density evolution was applied, which was similarly considered before by Kurkoski *et al.* in [28].

Starting from an extensive analysis of decoder construction for regular LDPC codes and binary modulation, the decoder construction framework was generalized to also work for higher order modulation schemes. Interestingly, unlike for conventional LLR based decoding algorithms, this required a major change in the system design with a novel technique that is called *message alignment*. The basic phenomenon that causes the requirement of message alignment for higher order modulation schemes is that identical integer indices which are passed into a random decoding graph could imply different beliefs on the respective codeword bits, depending on their position in the bit label of the modulation pattern. Since this ambiguity cannot be resolved from knowing only the incoming integer indices of a node, message alignment has been introduced to *align* the integer messages propagated into the decoding graph.

After the study of regular LDPC codes, a very similar alignment problem was observed in the design of Information Bottleneck decoders for irregular LDPC codes. Here, the outgoing integer messages of nodes with different degrees were not automatically aligned, due to the independent design of the node operations for the different node degrees. Therefore, the decoder construction framework for regular codes was amended by inclusion of a message alignment step to construct Information Bottleneck LDPC decoders for irregular codes.

Following the study of LDPC decoders, the Information Bottleneck design principle was used for quantized channel estimation and detection, such that Information Bottleneck LDPC decoders could also be applied to the quantized output of fading channels.

Receiver design with the Information Bottleneck method was revealed to end up in designing relevant information preserving lookup tables. As the most important result of this thesis, it was possible to build strongly quantized receiver components which could approach the performance of double precision reference algorithms up to small fractions of a decibel over E_b/N_0 in all aforementioned use cases of the Information Bottleneck method in receiver design. This result holds, although all required operations for signal processing were just lookup operations in static tables and the tables were designed offline and used mismatched to their actual design-E_b/N_0. Moreover, the memory demands required to store the lookup tables were practical.

Practical implementations of Information Bottleneck LDPC decoders and state-of-the-art decoders on a DSP platform were developed and compared in order to investigate potential benefits of the Information Bottleneck approach to signal processing exemplarily. The presented results have shown that not only the bit error rate performance of the constructed decoders was very close to that of the belief propagation decoding algorithm and the min-sum decoding algorithm was outperformed, but also the achievable net decoding throughputs of Information Bottleneck decoders were much higher than the ones of the conventional decoding algorithms on the same platform.

Information Bottleneck graphs were introduced and extensively used to visualize information relations between concatenated signal processing blocks designed with the Information Bottleneck method. Their ability to hierarchically model receiver components by opening and closing nodes allows to adapt the abstaction level of a receiver component to the required level at any point. It was shown that concatenating Information Bottleneck nodes in an Information Bottleneck graph essentially corresponds to designing a meaningful flow of relevant information through the signal processors. This process is the most fascinating aspect that has been revealed in the scope of this thesis, as it leads from a signal processing perspective to an information processing perspective on connected signal processing blocks.

Despite all blocks just process unsigned integer indices which require only a few bits in hardware, in the end the designed receivers manage to extract the information on the transmitted data without significant losses in comparison to the optimum conventional signal processing algorithms with high precision. Moreover, it is very easy to trade performance of the resulting signal processing blocks for complexity by simply changing the output bit widths of the applied Information Bottleneck algorithms used for the node design. This factor leaves enormous freedom to the designer of an Information Bottleneck receiver, as every single Information Bottleneck node can use a different output cardinality which can be adapted to the specific needs and hardware resources. As a result, the Information Bottleneck approach to receiver design has the potential to have a huge impact on the development of future communication receivers, especially if very low bit widths and high flexibility are required.

Appendix A

A.1 Proof of Proposition 2.2.1

Since $\mathsf{X} = \mathsf{Y}$, $p(x|y) = \delta(x-y)$, where $\delta(.)$ denotes the Kronecker delta function. Then the mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ can be expressed as

$$\begin{aligned}\mathrm{I}(\mathsf{X};\mathsf{Y}) &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \delta(x-y)p(y)\log\frac{\delta(x-y)}{p(x)} = \\ &\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \delta(x-y)p(y)\log\delta(x-y) - \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \delta(x-y)p(y)\log p(x) = \\ &\sum_{y\in\mathcal{Y}} p(y)\underbrace{\log\delta(y-y)}_{\log\delta(0)=\log 1=0} - \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \delta(x-y)p(y)\log p(x). \quad \text{(A.1)}\end{aligned}$$

The second term can be resolved as

$$\begin{aligned}&-\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \delta(x-y)p(y)\log p(x) = \\ &\qquad -\sum_{y\in\mathcal{Y}} p(y)\sum_{x\in\mathcal{X}}\delta(x-y)\log p(x) = -\sum_{y\in\mathcal{Y}} p(y)\log p(y) = \mathrm{H}(\mathsf{Y}) \quad \text{(A.2)}\end{aligned}$$

and also as

$$\begin{aligned}&-\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \delta(x-y)p(y)\log p(x) = \\ &\quad -\sum_{x\in\mathcal{X}}\log p(x)\sum_{y\in\mathcal{Y}}\delta(x-y)p(y) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = \mathrm{H}(\mathsf{X}). \quad \square \quad \text{(A.3)}\end{aligned}$$

A.2 Proof of Proposition 2.2.2

The proof follows directly from the definition of the mutual information, i.e.,

$$\begin{aligned}
\mathsf{I}(\mathsf{X};\mathsf{Y}_0,\mathsf{Y}_1) &= \sum_{x\in\mathcal{X}}\sum_{y_0\in\mathcal{Y}_0}\sum_{y_1\in\mathcal{Y}_1} p(x,y_0,y_1)\log\frac{p(x,y_0,y_1)}{p(x)p(y_0,y_1)} = \\
&\sum_{y_1\in\mathcal{Y}_1} p(y_1)\sum_{x\in\mathcal{X}}\sum_{y_0\in\mathcal{Y}_0} p(x,y_0|y_1)\log\frac{p(x,y_0|y_1)}{p(x)p(y_0|y_1)} = \\
&\sum_{y_1\in\mathcal{Y}_1} p(y_1)\sum_{x\in\mathcal{X}}\sum_{y_0\in\mathcal{Y}_0} p(x,y_0|y_1)\log\frac{p(x,y_0|y_1)p(x|y_1)}{p(x)p(y_0|y_1)p(x|y_1)} = \\
&\sum_{y_1\in\mathcal{Y}_1} p(y_1)\sum_{x\in\mathcal{X}}\sum_{y_0\in\mathcal{Y}_0} p(x,y_0|y_1)\log\frac{p(x,y_0|y_1)}{p(x|y_1)p(y_0|y_1)} + \\
&\sum_{y_1\in\mathcal{Y}_1} p(y_1)\sum_{x\in\mathcal{X}}\sum_{y_0\in\mathcal{Y}_0} p(x,y_0|y_1)\log\frac{p(x,y_0|y_1)}{p(x)p(y_1)} = \\
&\sum_{y_1\in\mathcal{Y}_1} p(y_1)\sum_{x\in\mathcal{X}}\sum_{y_0\in\mathcal{Y}_0} p(x,y_0|y_1)\log\frac{p(x,y_0|y_1)}{p(x|y_1)p(y_0|y_1)} + \\
&\sum_{y_1\in\mathcal{Y}_1} p(y_1)\sum_{x\in\mathcal{X}}\log\frac{p(x,y_1)}{p(x)p(y_1)}\underbrace{\sum_{y_0\in\mathcal{Y}_0} p(x,y_0|y_1)}_{p(x|y_1)} = \\
&\sum_{y_1\in\mathcal{Y}_1} p(y_1)\sum_{x\in\mathcal{X}}\sum_{y_0\in\mathcal{Y}_0} p(x,y_0|y_1)\log\frac{p(x,y_0|y_1)}{p(x|y_1)p(y_0|y_1)} + \\
&\qquad\sum_{x\in\mathcal{X}}\sum_{y_1\in\mathcal{Y}_1} p(x,y_1)\log\frac{p(x,y_1)}{p(x)p(y_1)}. \quad \square \qquad \text{(A.4)}
\end{aligned}$$

A.3 Derivation of Equation (2.39)

Recall that ρ_k is the fraction of edges connected to check nodes with degree k and that ρ'_k is the fraction of degree k check nodes. Let E denote the total number of edges in the Tanner graph of $\mathbf{H}$. Then $E\rho_k$ is the number of edges connected to degree k check nodes and $E\rho_k/k$ is the number of such degree k check nodes. The total number of check nodes is $N_\mathrm{c} = E\sum_{m=2}^{d_\mathrm{c}^{\max}}\rho_m/m$. Then ρ'_k is given by

$$\rho'_k = \frac{E\rho_k/k}{E\sum\limits_{m=2}^{d_\mathrm{c}^{\max}}\rho_m/m} = \frac{\rho_k/k}{\sum\limits_{m=2}^{d_\mathrm{c}^{\max}}\rho_m/m} \qquad \text{(A.5)}$$

Similarly, one obtains

$$\lambda'_l = \frac{E\lambda_l/l}{E\sum\limits_{n=1}^{d_{\mathrm{v}}^{\max}} \lambda_n/n} = \frac{\lambda_l/l}{\sum\limits_{n=1}^{d_{\mathrm{v}}^{\max}} \lambda_n/n}. \tag{A.6}$$

Finally, the code rate is

$$R = 1 - \frac{\sum\limits_{l=1}^{d_{\mathrm{v}}^{\max}} l\lambda'_l}{\sum\limits_{k=2}^{d_{\mathrm{c}}^{\max}} k\rho'_k} = 1 - \frac{\frac{\sum\limits_{l=1}^{d_{\mathrm{v}}^{\max}} \lambda_l}{\sum\limits_{n=1}^{d_{\mathrm{v}}^{\max}} \lambda_n/n}}{\frac{\sum\limits_{k=2}^{d_{\mathrm{c}}^{\max}} \rho_k}{\sum\limits_{m=2}^{d_{\mathrm{c}}^{\max}} \rho_m/m}} = 1 - \frac{\sum\limits_{m=2}^{d_{\mathrm{c}}^{\max}} \rho_m/m}{\sum\limits_{n=1}^{d_{\mathrm{v}}^{\max}} \lambda_n/n} = 1 - \frac{\sum\limits_{k=2}^{d_{\mathrm{c}}^{\max}} \rho_k/k}{\sum\limits_{l=1}^{d_{\mathrm{v}}^{\max}} \lambda_l/l}. \tag{A.7}$$

A.4 Derivation of Equation (2.59)

The box-plus sum $L(x_0) = L(b_0) \boxplus L(b_1)$ is a deterministic function of $L(b_0)$ and $L(b_1)$. When interpreting all LLRs as random variables, as a result

$$p\left(L(x_0)|L(b_0), L(b_1)\right) = \delta\left(L(x_0) - (L(b_0) \boxplus L(b_1))\right). \tag{A.8}$$

Since $L(b_0)$ and $L(b_1)$ are modelled to be independent, their joint distribution is given by

$$p\left(L(b_0), L(b_1)\right) = p\left(L(b_0)\right) p\left(L(b_1)\right). \tag{A.9}$$

Hence, the joint distribution $p\left(L(x_0), L(b_0), L(b_1)\right)$ is

$$\begin{aligned} p\left(L(x_0), L(b_0), L(b_1)\right) &= p\left(L(x_0)|L(b_0), L(b_1)\right) p\left(L(b_0)\right) p\left(L(b_1)\right) = \\ &\delta\left(L(x_0) - (L(b_0) \boxplus L(b_1))\right) p\left(L(b_0)\right) p\left(L(b_1)\right). \end{aligned} \tag{A.10}$$

Finally, in order to obtain $p\left(L(x_0)\right)$, one has to marginalize the joint distribution $p\left(L(x_0), L(b_0), L(b_1)\right)$ over $L(b_0)$ and $L(b_1)$. Again, it is assumed that the continuous LLRs are finely discretized, such that this marginalization is the sum

$$\begin{aligned} p\left(L(x_0)\right) &= \sum_{L(b_0)} \sum_{L(b_1)} p\left(L(x_0), L(b_0), L(b_1)\right) = \\ &\sum_{L(b_0)} \sum_{L(b_1)} \delta\left(L(x_0) - (L(b_0) \boxplus L(b_1))\right) p\left(L(b_0)\right) p\left(L(b_1)\right) = \\ &\sum_{\substack{L(b_0), L(b_1): \\ L(x_0) = L(b_0) \boxplus L(b_1)}} p\left(L(b_0)\right) p\left(L(b_1)\right). \end{aligned} \tag{A.11}$$

A.5 Derivation of the Information Bottleneck Equations

The Markov chain relation $\mathsf{X} \to \mathsf{Y} \to \mathsf{T}$ directly implies the connections

$$p(t) = \sum_{y \in \mathcal{Y}} p(t|y)p(y) \tag{A.12}$$

$$p(t|x) = \sum_{y \in \mathcal{Y}} p(t|y)p(y|x). \tag{A.13}$$

It is crucial to note that for any fixed pair $(t, y) \in \mathcal{T} \times \mathcal{Y}$, $p(t|y)$ in these equations is a real number from the interval $[0, 1]$. This means that for any fixed (t, y), $p(t|y)$ is just a single number. Hence, in a mathematical sense $p(t|y)$ for any fixed pair (t, y) can be interpreted as a variable which can be involved in equations. In total there exist $|\mathcal{T}||\mathcal{Y}|$ pairs (t, y) and, therefore, just as many variables $p(t|y)$. An exemplary equation which involves several variables $p(t|y)$ is the law of total probability in Equation (3.4). The interpretation of single $p(t|y)$ for fixed (t, y) as variables allows to apply simple *differential* calculus to functions involving any of these variables. For example, for a fixed pair $(t, y) \in \mathcal{T} \times \mathcal{Y}$ one might calculate $\frac{\partial}{\partial p(t|y)} p(t)$. Despite this notation suggests that the partial derivative is taken with respect to a function $p(t|y)$, for a fixed (t, y) this expression is nothing different than a simple partial derivative of a function which can be calculated using only simple differential calculus. Carrying out the partial derivative according to Equation (A.12) yields

$$\frac{\partial}{\partial p(t|y)} p(t) = \frac{\partial}{\partial p(t|y)} \sum_{y' \in \mathcal{Y}} p(t|y')p(y') = \sum_{y' \in \mathcal{Y}} p(y') \frac{\partial}{\partial p(t|y)} p(t|y'). \tag{A.14}$$

Please note that it is strictly necessary to relabel y into y' in Equation (A.14) to take the partial derivative with respect to $p(t|y)$ because otherwise it is impossible to distinguish $p(t|y)$ for the considered fixed pair (t, y) from the other appearing summands. Considering that $p(t|y')$ for $y' \neq y$ is independent of $p(t|y)$ and thus $\frac{\partial}{\partial p(t|y)} p(t|y') = 0$ for $y \neq y'$ and that $\frac{\partial}{\partial p(t|y)} p(t|y') = 1$ for $y = y'$, one finally obtains

$$\frac{\partial}{\partial p(t|y)} p(t) = \sum_{y' \in \mathcal{Y}} p(y') \frac{\partial}{\partial p(t|y)} p(t|y') = p(y). \tag{A.15}$$

Starting from Equation (A.13), a completely analogous argumentation yields

$$\frac{\partial}{\partial p(t|y)} p(t|x) = \sum_{y' \in \mathcal{Y}} p(y'|x) \frac{\partial}{\partial p(t|y)} p(t|y') = p(y|x). \tag{A.16}$$

In order to determine $p(t|y)$, one has to seek for the so called critical points of the Lagrangian (3.3). A critical point of the Lagrangian is a point where $\frac{\partial}{\partial p(t|y)} \mathcal{L}(p(t|y)) = 0 \, \forall (t, y) \in \mathcal{T} \times \mathcal{Y}$. Therefore, we need to calculate the partial derivative

$$\begin{aligned} &\frac{\partial}{\partial p(t|y)} \mathcal{L}(p(t|y)) = \\ &\quad \frac{\partial}{\partial p(t|y)} \mathrm{I}(\mathsf{Y}; \mathsf{T}) - \beta \frac{\partial}{\partial p(t|y)} \mathrm{I}(\mathsf{X}; \mathsf{T}) - \frac{\partial}{\partial p(t|y)} \sum_{y' \in \mathcal{Y}} \lambda(y') \sum_{t' \in \mathcal{T}} p(t'|y') \end{aligned} \tag{A.17}$$

with differential calculus, just as it has been applied in Equations (A.14) and (A.15). As it is indicated by Equation (A.17), due to the linearity of the partial differentiation operation, the differentiation can be carried out in three independent steps of differentiating $\mathrm{I}(\mathsf{Y}; \mathsf{T})$, $\mathrm{I}(\mathsf{X}; \mathsf{T})$ and the final sum term in Equation (A.17). In order to do so we first reformulate $\mathrm{I}(\mathsf{Y}; \mathsf{T})$ as

$$\begin{aligned} \mathrm{I}(\mathsf{Y}; \mathsf{T}) &= \sum_{y' \in \mathcal{Y}} \sum_{t' \in \mathcal{T}} p(y', t') \log \frac{p(y', t')}{p(y') p(t')} = \sum_{y' \in \mathcal{Y}} \sum_{t' \in \mathcal{T}} p(y') p(t'|y') \log \frac{p(t'|y')}{p(t')} = \\ &\left(\sum_{y' \in \mathcal{Y}} \sum_{t' \in \mathcal{T}} p(y') p(t'|y') \log p(t'|y') \right) - \left(\sum_{y' \in \mathcal{Y}} \sum_{t' \in \mathcal{T}} p(y') p(t'|y') \log p(t') \right) = \\ &\left(\sum_{y' \in \mathcal{Y}} \sum_{t' \in \mathcal{T}} p(y') p(t'|y') \log p(t'|y') \right) - \left(\sum_{t' \in \mathcal{T}} \log p(t') \sum_{y' \in \mathcal{Y}} p(y') p(t'|y') \right) = \\ &\left(\sum_{y' \in \mathcal{Y}} \sum_{t' \in \mathcal{T}} p(y') p(t'|y') \log p(t'|y') \right) - \left(\sum_{t' \in \mathcal{T}} p(t') \log p(t') \right). \end{aligned} \tag{A.18}$$

Therefore, differentiating $\mathrm{I}(\mathsf{Y}; \mathsf{T})$ with respect to $p(t|y)$ yields

$$\begin{aligned} &\frac{\partial}{\partial p(t|y)} \mathrm{I}(\mathsf{Y}; \mathsf{T}) = \\ &\left(\sum_{y' \in \mathcal{Y}} \sum_{t' \in \mathcal{T}} p(y') \frac{\partial}{\partial p(t|y)} p(t'|y') \log p(t'|y') \right) - \left(\sum_{t' \in \mathcal{T}} \frac{\partial}{\partial p(t|y)} p(t') \log p(t') \right) = \\ &\qquad p(y) \frac{\partial}{\partial p(t|y)} p(t|y) \log p(t|y) - \frac{\partial}{\partial p(t|y)} p(t) \log p(t), \end{aligned} \tag{A.19}$$

where it was again used that all summands independent of $p(t|y)$ vanish under the differentiation with respect to $p(t|y)$. Application of the product rule and the chain rule of differentiation finally yields

$$\begin{aligned}\frac{\partial}{\partial p(t|y)}\mathrm{I}(\mathsf{Y};\mathsf{T}) = \\ p(y)\left(\log p(t|y) + \frac{p(t|y)}{p(t|y)}\right) - \left(p(y)\log p(t) + \frac{1}{p(t)}p(y)p(t)\right) = \\ p(y)\left(\log p(t|y) + 1\right) - p(y)\left(\log p(t) + 1\right) = p(y)\log\frac{p(t|y)}{p(t)}. \quad \text{(A.20)}\end{aligned}$$

Equation (A.20) used that $\frac{\partial}{\partial p(t|y)}p(t) = p(y)$ from Equation (A.15).

To calculate $\frac{\partial}{\partial p(t|y)}\mathrm{I}(\mathsf{X};\mathsf{T})$ in a second step, we similarly first rewrite

$$\begin{aligned}\mathrm{I}(\mathsf{X};\mathsf{T}) = \sum_{x\in\mathcal{X}}\sum_{t'\in\mathcal{T}} p(x,t')\log\frac{p(x,t')}{p(x)p(t')} = \\ \left(\sum_{x\in\mathcal{X}}\sum_{t'\in\mathcal{T}} p(x)p(t'|x)\log p(t'|x)\right) - \left(\sum_{x\in\mathcal{X}}\sum_{t'\in\mathcal{T}} p(x)p(t'|x)\log p(t')\right) = \\ \left(\sum_{x\in\mathcal{X}}\sum_{t'\in\mathcal{T}} p(x)p(t'|x)\log p(t'|x)\right) - \left(\sum_{t'\in\mathcal{T}}\log p(t')\sum_{x\in\mathcal{X}} p(x)p(t'|x)\right) = \\ \left(\sum_{x\in\mathcal{X}}\sum_{t'\in\mathcal{T}} p(x)p(t'|x)\log p(t'|x)\right) - \left(\sum_{t'\in\mathcal{T}} p(t')\log p(t')\right). \quad \text{(A.21)}\end{aligned}$$

Then

$$\begin{aligned}\frac{\partial}{\partial p(t|y)}\mathrm{I}(\mathsf{X};\mathsf{T}) = \left(\sum_{x\in\mathcal{X}}\sum_{t'\in\mathcal{T}} p(x)\frac{\partial}{\partial p(t|y)}p(t'|x)\log p(t'|x)\right) - \\ \left(\sum_{t'\in\mathcal{T}}\frac{\partial}{\partial p(t|y)}p(t')\log p(t')\right). \quad \text{(A.22)}\end{aligned}$$

The second term in Equation (A.22) already appeared in Equation (A.19). Differentiation of the first term again requires the product rule and the chain rule of differentiation, such that finally

$$\begin{aligned}
&\frac{\partial}{\partial p(t|y)}\mathrm{I}(\mathsf{X};\mathsf{T}) = \\
&\left(\sum_{x\in\mathcal{X}} p(x)\left(p(y|x)\log p(t|x) + \frac{1}{p(t|x)}p(y|x)p(t|x)\right)\right) - p(y)\left(\log p(t)+1\right) = \\
&\qquad\left(\sum_{x\in\mathcal{X}} p(x)p(y|x)\left(\log p(t|x)+1\right)\right) - p(y)\left(\log p(t)+1\right) = \\
&\qquad\left(\sum_{x\in\mathcal{X}} p(y)p(x|y)\left(\log p(t|x)+1\right)\right) - p(y)\left(\log p(t)+1\right) = \\
&\left(\sum_{x\in\mathcal{X}} p(y)p(x|y)\left(\log p(t|x)+1\right)\right) - \left(\sum_{x\in\mathcal{X}} p(x|y)p(y)\left(\log p(t)+1\right)\right) = \\
&\qquad p(y)\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(t|x)}{p(t)} = p(y)\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|t)}{p(x)}. \quad \text{(A.23)}
\end{aligned}$$

Equation (A.23) used that $\frac{\partial}{\partial\, p(t|y)}p(t|x) = p(y|x)$ according to Equation (A.16).

To finally obtain the partial derivative $\frac{\partial}{\partial p(t|y)}\mathcal{L}(p(t|y))$, one is left with determining

$$\frac{\partial}{\partial p(t|y)}\sum_{y'\in\mathcal{Y}}\lambda(y')\sum_{t'\in\mathcal{T}}p(t'|y') = \sum_{y'\in\mathcal{Y}}\sum_{t'\in\mathcal{T}}\frac{\partial}{\partial p(t|y)}\lambda(y')p(t'|y') = \lambda(y). \quad \text{(A.24)}$$

Combining Equations (A.20), (A.23) and (A.24) yields

$$\begin{aligned}
\frac{\partial}{\partial p(t|y)}\mathcal{L}(p(t|y)) &= p(y)\log\frac{p(t|y)}{p(t)} - \beta p(y)\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|t)}{p(x)} - \lambda(y) = \\
&\qquad p(y)\left(\log\frac{p(t|y)}{p(t)} - \beta\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|t)}{p(x)} - \frac{\lambda(y)}{p(y)}\right). \quad \text{(A.25)}
\end{aligned}$$

From here, elementary operations yield

$$\begin{aligned}
&\frac{\partial}{\partial p(t|y)}\mathcal{L}(p(t|y)) = \\
&\qquad p(y)\left(\log\frac{p(t|y)}{p(t)} - \beta\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|t)p(x|y)}{p(x)p(x|y)} - \frac{\lambda(y)}{p(y)}\right) = \\
&\quad p(y)\left(\log\frac{p(t|y)}{p(t)} - \beta\sum_{x\in\mathcal{X}} p(x|y)\left(\log\frac{p(x|t)}{p(x|y)} + \log\frac{p(x|y)}{p(x)}\right) - \frac{\lambda(y)}{p(y)}\right) = \\
&\quad p(y)\left(\log\frac{p(t|y)}{p(t)} - \beta\sum_{x\in\mathcal{X}} p(x|y)\left(-\log\frac{p(x|y)}{p(x|t)} + \log\frac{p(x|y)}{p(x)}\right) - \frac{\lambda(y)}{p(y)}\right) = \\
&\quad p(y)\left(\log\frac{p(t|y)}{p(t)} - \beta\sum_{x\in\mathcal{X}} p(x|y)(-1)\left(\log\frac{p(x|y)}{p(x|t)} - \log\frac{p(x|y)}{p(x)}\right) - \frac{\lambda(y)}{p(y)}\right) = \\
&p(y)\left(\log\frac{p(t|y)}{p(t)} + \beta\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|y)}{p(x|t)} - \beta\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|y)}{p(x)} - \frac{\lambda(y)}{p(y)}\right) = \\
&\qquad p(y)\left(\log\frac{p(t|y)}{p(t)} + \beta\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|y)}{p(x|t)} - \right. \\
&\qquad\qquad \left.\left(\beta\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|y)}{p(x)} + \frac{\lambda(y)}{p(y)}\right)\right) \qquad \text{(A.26)}
\end{aligned}$$

Substituting

$$\tilde{\lambda}(y,\beta) = \beta\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|y)}{p(x)} + \frac{\lambda(y)}{p(y)} \qquad \text{(A.27)}$$

to absorb the terms which are independent of $p(t|y)$, finally reveals that the seeked partial derivative is given by

$$\begin{aligned}
&\frac{\partial}{\partial p(t|y)}\mathcal{L}(p(t|y)) = \\
&\qquad p(y)\left(\log\frac{p(t|y)}{p(t)} + \beta\sum_{x\in\mathcal{X}} p(x|y)\log\frac{p(x|y)}{p(x|t)} - \tilde{\lambda}(y,\beta)\right) = \\
&\qquad p(y)\left(\log\frac{p(t|y)}{p(t)} + \beta D_{\mathrm{KL}}\{p(x|y)\,|\,p(x|t)\} - \tilde{\lambda}(y,\beta)\right). \qquad \text{(A.28)}
\end{aligned}$$

Since $p(y) > 0$, to find critical points one has to solve

$$\log\frac{p(t|y)}{p(t)} + \beta D_{\mathrm{KL}}\{p(x|y)\,|\,p(x|t)\} - \tilde{\lambda}(y,\beta) = 0. \qquad \text{(A.29)}$$

This yields

$$\log \frac{p(t|y)}{p(t)} = -\beta D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\} + \tilde{\lambda}(y,\beta) \tag{A.30}$$

$$\frac{p(t|y)}{p(t)} = \exp\left(-\beta D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}\right) \exp\left(\tilde{\lambda}(y,\beta)\right) \tag{A.31}$$

$$p(t|y) = p(t) \exp\left(-\beta D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}\right) \exp\left(\tilde{\lambda}(y,\beta)\right). \tag{A.32}$$

Due to the fact that $p(t|y)$ has to be a valid probability distribution and recalling that $\tilde{\lambda}(y,\beta)$ is independent of $p(t|y)$ one is finally able to express $p(t|y)$ as

$$p(t|y) = \frac{p(t)}{Z(y,\beta)} \exp\left(-\beta D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}\right), \tag{A.33}$$

where $Z(y,\beta)$ is a normalization function which is given by

$$Z(y,\beta) = \sum_{t\in\mathcal{T}} p(t) \exp\left(-\beta D_{\mathrm{KL}}\{p(x|y) \,|\, p(x|t)\}\right). \tag{A.34}$$

The result of this derivation is an implicit formulation for $p(t|y)$ in Equation (A.33). The solution is implicit because due to Markov chain relation $\mathsf{X} \to \mathsf{Y} \to \mathsf{T}$, $p(x|t)$ which appears on the right hand side of Equation (A.33) inherently depends on $p(t|y)$, as it is expressed in Equations (3.6) and (3.7).

A.6 Derivation of Equation (5.39)

By definition, the conditional mutual information $\mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y})$ is given by

$$\begin{aligned}\mathrm{I}(\mathsf{B};\mathsf{L}|\mathsf{Y}) = \sum_y p(y) \sum_b \sum_l p(b,l|y) \log \frac{p(b,l|y)}{p(b|y)p(l|y)} = \\ \sum_y \sum_l p(y,l) \sum_b p(b|y,l) \log \frac{p(b|y,l)}{p(b|y)} = \\ \mathbb{E}_{\mathsf{Y},\mathsf{L}}\left\{D_{\mathrm{KL}}\left\{p(b|y,l) \,|\, p(b|y)\right\}\right\}. \end{aligned} \tag{A.35}$$

A.7 Memory Demands of Receiver Components

To obtain the size of the lookup tables in the proposed receiver structure, one has to count the numbers of bits required to store all lookup table entries. This is done in the following for the presented receivers with and without decision feedback.

A.7.1 Lookup Table Memory without Decision Feedback

The Information Bottleneck channel estimation scheme visualized in Figure 5.45 consists of $(P-1)$ two-input lookup tables. The inputs v_0^{re} and v_1^{re} of the first Information Bottleneck node $p\left(\hat{h}_0^{\mathrm{re}}|v_0^{\mathrm{re}}, v_1^{\mathrm{re}}\right)$ can take 2^q values each, resulting in 2^{2q} possible input configurations. The following Information Bottleneck nodes input an intermediate result from their ancestor node and another value from the quantizer, resulting in $2^{q^{\mathrm{ce}}} \cdot 2^q = 2^{q+q^{\mathrm{ce}}}$ possible input configurations. Each lookup table entry requires q^{ce} bits. Hence, the overall memory amount required to store the lookup tables for the Information Bottleneck channel estimation is

$$\mathrm{size}^{\mathrm{ce}} = q^{\mathrm{ce}} \cdot \frac{2^{2q} + 2^{q+q^{\mathrm{ce}}} \cdot (P-2)}{8 \cdot 1000} \text{ kilobytes}, \tag{A.36}$$

where q is the quantization bit width of the channel output quantizer, q^{ce} is the output bit width of the channel estimation lookup tables which is adjusted by the output cardinality of the applied Information Bottleneck algorithm to design the channel estimation and P is the number of pilot symbols.

The overall size of the lookup tables used for the detection lookup tables from Figure 5.46 is determined by the number of possible input configurations of the Information Bottleneck nodes $p\left(t_k^{\mathrm{re}}|\hat{h}^{\mathrm{re}}, r_k^{\mathrm{re}}\right)$ and $p\left(t_k|t_k^{\mathrm{re}}, t_k^{\mathrm{im}}\right)$. The node $p\left(t_k^{\mathrm{im}}|\hat{h}^{\mathrm{im}}, r_k^{\mathrm{im}}\right)$ is identical to $p\left(t_k^{\mathrm{re}}|\hat{h}^{\mathrm{re}}, r_k^{\mathrm{re}}\right)$ and does not need to be stored. Hence the size of the detector from Figure 5.46 is

$$\mathrm{size}^{\mathrm{det}} = q^{\mathrm{det}} \cdot \frac{2^{q+q^{\mathrm{ce}}} + 2^{2q^{\mathrm{det}}}}{8 \cdot 1000} \text{ kilobytes}, \tag{A.37}$$

where q is the quantization bit width of the channel output quantizer, q^{ce} is the output bit width of the channel estimation lookup tables and q^{det} is the output bit width of the detector.

The overall size of the lookup tables applied in the Information Bottleneck LDPC decoder is given by

$$\mathrm{size}^{\mathrm{dec}} = q^{\mathrm{det}} \cdot \frac{i_{\max} \cdot 2^{2q^{\mathrm{det}}}(d_{\mathrm{v}} + d_{\mathrm{c}} - 2)}{8 \cdot 1000} \text{ kilobytes}, \tag{A.38}$$

where $i_{\max}$ is the maximum number of decoder iterations, d_{v} and d_{c} are the variable node degrees of the applied regular LDPC code and q^{det} is the output bit width of the detector. Here, it is assumed that the bit width of the variable-to-check and the check-to-variable messages is identical and also q^{det}.

A.7.2 Lookup Table Memory with Decision Feedback

When decision feedback shall be used, the detector from Figure 5.46 has to be constructed once for the initial detection and one updated variant has to be constructed for the feedback iterations. Hence, the overall size of the detection lookup tables is

$$\text{size}^{\text{det}} = q^{\text{det}} \cdot \frac{2^{q+q^{\text{ce}}+1} + 2^{2q^{\text{det}}+1}}{8 \cdot 1000} \text{ kilobytes.} \tag{A.39}$$

The channel estimation scheme consists of a forward channel estimation scheme shown in Figure 5.45 and a feedback channel estimation scheme shown in Figure 5.49. The latter processes N_{FB} inputs. The overall size of the forward channel estimation lookup tables is given by Equation (A.36). The size of the feedback channel estimation scheme is

$$\text{size}^{\text{fb}} = q^{\text{ce}} \cdot \frac{2^{2q} + 2^{q+q^{\text{ce}}} \cdot (N_{\text{FB}} - 2) + 2^{2q^{\text{ce}}}}{8 \cdot 1000} \text{ kilobytes.} \tag{A.40}$$

The size of the lookup tables in the LDPC decoder is given by Equation (A.38). This decoder is held static for all decision feedback iterations and only needs to be stored once.

Bibliography

[1] C. Kestel, M. Herrmann, and N. Wehn, "When channel coding hits the implementation wall," in *Proceedings of 2018 IEEE 10th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Hong Kong, China, Dec. 2018, pp. 1–6.

[2] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[3] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," in *Proceedings of 37th Allerton Conference on Communication and Computation*, Monticello, USA, Sep. 1999, pp. 368–377.

[4] S. Gordon, H. Greenspan, and J. Goldberger, "Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations," in *Proceedings of Ninth IEEE International Conference on Computer Vision*, Nice, France, Oct. 2003, pp. 370–377.

[5] A. Bardera, J. Rigau, I. Boada, M. Feixas, and M. Sbert, "Image segmentation using information bottleneck method," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1601–1612, Jul. 2009.

[6] S. Buddha, K. So, J. Carmena, and M. Gastpar, "Function identification in neuron populations via information bottleneck," *Entropy 2013*, vol. 15, no. 5, pp. 1587–1608, May 2013.

[7] N. Slonim, N. Friedman, and N. Tishby, "Unsupervised document classification using sequential information maximization," in *Proceedings of 25th ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, Aug. 2002, pp. 129–136.

[8] N. Slonim, R. Somerville, N. Tishby, and O. Lahav, "Objective classification of galaxy spectra using the information bottleneck method," *Monthly Notices of the Royal Astronomical Society, Oxford University Press*, vol. 323, no. 2, pp. 270–284, May 2001.

[9] N. Slonim, "The information bottleneck: Theory and applications," Ph.D. dissertation, Hebrew University of Jerusalem, Jerusalem, Israel, 2002.

[10] S. H. Yella and H. Bourlard, "Information bottleneck based speaker diarization of meetings using non-speech as side information," in *Proceedings of 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 96–100.

[11] S. Still, "Information bottleneck approach to predictive inference," *Entropy 2014*, vol. 16, no. 2, pp. 968–989, Feb. 2014.

[12] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Proceedings of 2015 IEEE Information Theory Workshop (ITW)*, Jerusalem, Israel, Apr. 2015, pp. 1–5.

[13] G. Zeitler, "Low-precision analog-to-digital conversion and mutual information in channels with memory," in *Proceedings of 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Allerton, USA, Sep. 2010, pp. 745–752.

[14] A. Winkelbauer and G. Matz, "Rate-information-optimal gaussian channel output compression," in *Proceedings of 2014 48th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, USA, Mar. 2014, pp. 1–5.

[15] G. Zeitler, R. Koetter, G. Bauch, and J. Widmer, "Design of network coding functions in multihop relay networks," in *Proceedings of 2008 5th International Symposium on Turbo Codes and Related Topics*, Lausanne, Switzerland, Sep. 2008, pp. 249–254.

[16] G. Zeitler, R. Koetter, G. Bauch, and J. Widmer, "On quantizer design for soft values in the multiple-access relay channel," in *Proceedings of 2009 IEEE International Conference on Communications (ICC)*, Dresden, Germany, Jun. 2009, pp. 1–5.

[17] A. Winkelbauer and G. Matz, "Joint network-channel coding for the asymmetric multiple-access relay channel," in *Proceedings of 2012 IEEE International Conference on Communications (ICC)*, Ottawa, Canada, Jun. 2012, pp. 2485–2489.

[18] A. Winkelbauer and G. Matz, "Joint network-channel coding in the multiple-access relay channel: Beyond two sources," in *Proceedings of 2012 5th International Symposium on Communications, Control and Signal Processing*, Rome, Italy, May 2012, pp. 1–5.

[19] D. Kern and V. Kuehn, "Practical aspects of compress and forward with BICM in the 3-node relay channel," in *Proceedings of 2016 20th International ITG Workshop on Smart Antennas (WSA)*, Munich, Germany, Mar. 2016, pp. 1–7.

[20] D. Kern and V. Kuehn, "On compress and forward with multiple carriers in the 3-node relay channel exploiting information bottleneck graphs," in *Proceedings of 2017 11th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany, Feb. 2017, pp. 1–6.

[21] D. Kern and V. Kuehn, "On implicit and explicit channel estimation for compress and forward relaying OFDM schemes designed by information bottleneck graphs," in *2017 21th International ITG Workshop on Smart Antennas (WSA)*, Berlin, Germany, Mar. 2017, pp. 1–8.

[22] B. M. Kurkoski, "On the relationship between the KL means algorithm and the information bottleneck method," in *Proceedings of 2017 11th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany, Feb. 2017, pp. 1–6.

[23] S. Hassanpour, D. Wübben, A. Dekorsy, and B. Kurkoski, "On the relation between the asymptotic performance of different algorithms for information bottleneck framework," in *Proceedings of 2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017, pp. 1–6.

[24] S. Hassanpour, D. Wübben, and A. Dekorsy, "On the equivalence of double maxima and KL-means for information bottleneck-based source coding,"

in *Proceedings of 2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, Apr. 2018, pp. 1–6.

[25] S. Hassanpour, D. Wübben, and A. Dekorsy, "On the equivalence of two information bottleneck-based routines devised for joint source-channel coding," in *Proceedings of 2018 25th International Conference on Telecommunications (ICT)*, St. Malo, France, Jun. 2018, pp. 253–258.

[26] S. Hassanpour, D. Wübben, and A. Dekorsy, "A graph-based message passing approach for noisy source coding via information bottleneck principle," in *Proceedings of 2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6.

[27] S. Hassanpour, D. Wübben, and A. Dekorsy, "A graph-based message passing approach for joint source-channel coding via information bottleneck principle," in *Proceedings of 2018 IEEE 10th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Hong Kong, China, Dec. 2018, pp. 1–5.

[28] B. M. Kurkoski, K. Yamaguchi, and K. Kobayashi, "Noise thresholds for discrete LDPC decoding mappings," in *Proceedings of 2008 IEEE Global Communications Conference (GLOBECOM)*, New Orleans, USA, Nov. 2008, pp. 1–5.

[29] F. J. C. Romero and B. M. Kurkoski, "Decoding LDPC codes with mutual information-maximizing lookup tables," in *Proceedings of 2015 IEEE International Symposium on Information Theory (ISIT)*, Hong Kong, China, Jun. 2015, pp. 426–430.

[30] A. Balatsoukas-Stimming, M. Meidlinger, R. Ghanaatian, G. Matz, and A. Burg, "A fully-unrolled LDPC decoder based on quantized message passing," in *Proceedings of 2015 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2015, pp. 1–6.

[31] M. Meidlinger, A. Balatsoukas-Stimming, A. Burg, and G. Matz, "Quantized message passing for LDPC codes," in *Proceedings of 2015 49th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, USA, Nov. 2015, pp. 1606–1610.

[32] F. J. C. Romero and B. M. Kurkoski, "LDPC decoding mappings that maximize mutual information," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 9, pp. 2391–2401, Sep. 2016.

[33] M. Meidlinger and G. Matz, "On irregular LDPC codes with quantized message passing decoding," in *Proceedings of 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Sapporo, Japan, Jul. 2017, pp. 1–5.

[34] R. Ghanaatian, A. Balatsoukas-Stimming, T. C. Müller, M. Meidlinger, G. Matz, A. Teman, and A. Burg, "A 588-gb/s LDPC decoder based on finite-alphabet message passing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 329–340, Feb. 2018.

[35] M. Meidlinger, "Information-optimal decoding and demodulation on sparse graphs," Ph.D. dissertation, Technische Universität Wien, Wien, Österreich, 2018.

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

[37] H. Wymeersch, *Iterative Receiver Design*. New York, USA: Cambridge University Press, 2007.

[38] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[39] R. G. Gallager, "Low-density parity-check codes," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 1963.

[40] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon-limit error-correcting coding and decoding: Turbo codes," in *Proceedings of 1993 IEEE International Conference on Communications (ICC)*, Geneva, Switzerland, May 1993, pp. 1064–1070.

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

[42] "IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area network - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications." *IEEE Std. 802.11-2012*, 2012.

[43] "Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications; part 1: DVB-S2," *ETSI EN302 307 V1.4.1*, 2014.

[44] S. J. Johnson, *Iterative Error Correction.* New York, USA: Cambridge University Press, 2010.

[45] M. C. Davey and D. J. C. MacKay, "Low-density parity-check codes over GF(q)," *IEEE Communication Letters*, vol. 2, no. 6, pp. 165–167, Jun. 1998.

[46] Jia-ning Su, Zhou Jiang, Ke Liu, Xiao-yang Zeng, and Hao Min, "An efficient low complexity LDPC encoder based on LU factorization with pivoting," in *Proceedings of 2005 6th International Conference on ASIC*, Shanghai, China, Oct 2005, pp. 107–110.

[47] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, Feb 2001.

[48] D. Divsalar, H. Jin, and R.McEliece, "Coding theorems for "turbo-like" codes," in *Proceedings of 1998 36th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Allerton, USA, Oct 1998, pp. 201–210.

[49] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proceedings of 2000 IEEE 2nd International Symposium on Turbo Codes (ISTC)*, Brest, France, Sep. 2000, pp. 1–8.

[50] X. Wu, Y. Song, M. Jiang, and C. Zhao, "Adaptive-normalized/offset min-sum algorithm," *IEEE Communications Letters*, vol. 14, no. 7, pp. 667–669, July 2010.

[51] S.-Y. Chung, "On the construction of some capacity-approaching coding schemes," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2000.

[52] Sae-Young Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, Feb 2001.

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

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

[55] F. R. Kschischang, "Codes defined on graphs," *IEEE Communications Magazine*, vol. 41, no. 8, pp. 118–125, Aug. 2003.

[56] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[57] R. E. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, Jul. 1972.

[58] B. M. Kurkoski and H. Yagi, "Quantization of binary-input discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4544–4552, Aug. 2014.

[59] A. Winkelbauer, "Blind performance estimation and quantizer design with applications to relay networks," Ph.D. dissertation, Technische Universität Wien, Wien, Österreich, 2014.

[60] J. A. Zhang and B. M. Kurkoski, "Low-complexity quantization of discrete memoryless channels," in *Proceedings of 2016 International Symposium on Information Theory and Its Applications (ISITA)*, Monterey, USA, Oct. 2016, pp. 448–452.

[61] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, Jun. 2010.

[62] N. Slonim, N. Friedman, and N. Tishby, "Multivariate information bottleneck," *Neural computation*, vol. 18, no. 8, pp. 1739–1789, Jun. 2006.

[63] J. Lewandowsky, M. Stark, and G. Bauch, "Information bottleneck graphs for receiver design," in *Proceedings of 2016 IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 2888–2892.

[64] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[65] J. Lewandowsky and G. Bauch, "Trellis based node operations for LDPC decoders from the information bottleneck method," in *Proceedings of 2015 9th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Cairns, Australia, Dec. 2015, pp. 1–10.

[66] J. Lewandowsky and G. Bauch, "Information-optimum LDPC decoders based on the information bottleneck method," *IEEE Access*, vol. 6, pp. 4054–4071, Jan. 2018.

[67] D. Guo, S. Shamai, and S. Verdú, "Mutual information and minimum mean-square error in gaussian channels," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1261–1282, Apr. 2005.

[68] D. J. C. MacKay, "Encyclopedia of sparse graph codes," visited on Nov. 14th 2017. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[69] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss, "Information bottleneck for gaussian variables," *Journal of Machine Learning Research*, vol. 6, pp. 165–188, Jan. 2005.

[70] J. Lewandowsky, M. Stark, and G. Bauch, "Optimum message mapping LDPC decoders derived from the sum-product algorithm," in *Proceedings of 2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[71] J. Lewandowsky, M. Stark, and G. Bauch, "Message alignment for discrete LDPC decoders with quadrature amplitude modulation," in *Proceedings of 2017 IEEE International Symposium on Information Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 2925–2929.

[72] M. Stark, J. Lewandowsky, and G. Bauch, "Information-optimum LDPC decoders with message alignment for irregular codes," in *Proceedings of 2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6.

[73] M. Stark, J. Lewandowsky, and G. Bauch, "Information-bottleneck decoding of high-rate irregular LDPC codes for optical communication using message alignment," *Applied Sciences*, vol. 8, no. 10, Oct. 2018.

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

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

[76] J. Lewandowsky, M. Stark, R. Mendrzik, and G. Bauch, "Discrete channel estimation by integer passing in information bottleneck graphs," in *Proceedings of 2017 11th International ITG Conference on Systems, Communications and Coding (SCC)*, Feb. 2017, pp. 1–6.

[77] A. Genz, "Numerical computation of multivariate normal probabilities," *Journal of Computational and Graphical Statistics*, vol. 1, no. 2, pp. 141–149, Feb. 1992.

[78] C. Novak and G. Matz, "Low-complexity MIMO-BICM receivers with imperfect channel state information: Capacity-based performance comparison," in *Proceedings of 2010 IEEE Eleventh International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Marrakech, Morocco, Jun. 2010, pp. 1–5.

[79] Z. Zhang, L. Dolecek, M. Mainwright, V. Anantharam, and B. Nikolić, "Quantization effects in low-density parity-check decoders," in *Proceedings of the 2007 IEEE International Conference on Communications (ICC)*, Glasgow, UK, Jun. 2007, pp. 6231–6237.

[80] S. Park, B. Shim, and J. W. Choi, "Iterative channel estimation using virtual pilot signals for MIMO-OFDM systems," *IEEE Transactions on Signal Processing*, vol. 63, no. 12, pp. 3032–3045, Jun. 2015.

[81] M. C. Valenti and B. D. Woerner, "Iterative channel estimation and decoding of pilot symbol assisted turbo codes over flat-fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 9, pp. 1697–1705, Sep. 2001.

[82] J. Lewandowsky, M. Stark, and G. Bauch, "A discrete information bottleneck receiver with iterative decision feedback channel estimation," in *Proceedings of 2018 IEEE 10th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Hong Kong, China, Dec. 2018, pp. 1–5.

[83] J. Lewandowsky, G. Bauch, M. Tschauner, and P. Oppermann, "Design and evaluation of information bottleneck LDPC decoders for software defined radios," in *Proceedings of 2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Cairns, Australia, Dec. 2018, pp. 1–9.

[84] J. Lewandowsky, G. Bauch, M. Tschauner, and P. Oppermann, "Design and evaluation of information bottleneck LDPC decoders for digital signal processors," *IEICE Transactions on Communications*, vol. E102-B, no. 8, pp. 1363–1370, Jan. 2019.

[85] T. Ulversoy, "Software defined radio: Challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 4, pp. 531–555, May 2010.

[86] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26–38, May 1995.

[87] "TMS320C6474 data sheet," Texas Instruments Incorporated, Dallas, USA, Apr. 2011.

[88] "TMDSEVM6474L / TMDSEVM6474LE technical reference manual," eInfoChips Incorporated, California, USA, Mar. 2011.

[89] "TMS320C64x/C64x+ CPU and instruction set reference guide," Texas Instruments Incorporated, Dallas, USA, Jul. 2010.

This page is intentionally left blank.

Curriculum Vitae

Last name:	Lewandowsky
Given names:	Jan Christopher
Nationality:	German
Date of birth:	09.01.1987
Place of birth:	Wuppertal, Germany
08.1993-07.1997	Primary school in Wuppertal, Germany
08.1997-05.2006	Secondary school in Wuppertal, Germany
06.2006-07.2006	Self employed in Wuppertal, Germany
08.2006-09.2007	Officer candidate at the Bundeswehr officer school of the German Air Force in Fürstenfeldbruck, Germany
10.2007-02.2010	Studies of *electrical engineering* at the Universität der Bundeswehr Munich in Neubiberg, Germany Degree: Bachelor of Science (B.Sc.)
03.2010-09.2011	Studies of *electrical engineering* at the Universität der Bundeswehr Munich in Neubiberg, Germany Degree: Master of Science (M.Sc.)

10.2011-04.2013	Technical officer with the Bundeswehr Control and Reporting Center Sunrise in Schönewalde, Germany
05.2013-07.2018	Technical officer with the Federal Office of Bundeswehr Equipment, Information Technology and In-Service Support in Koblenz, Germany
08.2018-present	Scientific researcher with the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE) in Wachtberg, Germany
09.2014-present	External Ph.D. candidate with the Institute of Communications, Hamburg University of Technology in Hamburg, Germany