Reliability Information in Channel Decoding

Practical Aspects and Information Theoretical Bounds

Ingmar Land

Dissertation

Kiel 2005

Christian-Albrechts-University of Kiel
Faculty of Engineering
Information and Coding Theory Lab

Reliability Information in Channel Decoding

Practical Aspects and Information Theoretical Bounds

Dissertation

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

vorgelegt von

Ingmar Land

Kiel 2005

Tag der Einreichung: 7. September 2004
Tag der Disputation: 13. Dezember 2004

Berichterstatter: Prof. Dr.-Ing. Peter Adam Höher
Prof. Dr.-Ing. Johannes Huber
Prof. Dr.-Ing. Joachim Hagenauer

Preface

This thesis was written during my time as a research and teaching assistant at the Information and Coding Theory Lab, Faculty of Engineering, University of Kiel, Germany.

I wish to express my most sincere gratitude to my advisor Prof. Dr. Peter Adam Höher. I would like to thank him for all the inspiring discussions we had; for giving me scientific freedom which allowed me to develop my own ideas; and for introducing me to the scientific community. His scientific mind and his enthusiasm are reflected in this work.

I would like to thank Prof. Dr. Johannes Huber and Prof. Dr. Joachim Hagenauer for evaluating this work, for their comments on this thesis, and for their support.

I also would like to thank Prof. Dr. Ulrich Sorger, Prof. Dr. Johannes Huber, Dr. Simon Hüttinger, and Dr. Jossy Sayir for all the fruitful and stimulating discussions on information theory and coding, iterative decoding, and information combining.

Finally, I would like to thank all colleagues at the Information and Coding Theory Lab and at the Institute for Circuits and System Theory, University of Kiel, Germany, for the pleasant and inspiring working atmosphere, for all the discussions we had, and for the help with scientific, technical, and non-technical problems.

Ingmar Land

Kiel, May 2005

Abstract

This thesis addresses the use of reliability information in channel decoding and covers practical aspects as well as information-theoretical bounds. The considered transmission systems comprise linear binary channel encoders, symmetric memoryless communication channels, and non-iterative or iterative symbol-by-symbol soft-output channel decoders.

The notions of accurate and mismatched reliability values are introduced, and the measurement and improvement of the quality of reliability values are discussed. A criterion based on the Kullback-Leibler distance is proposed to assess the difference between accurate and mismatched reliability values. The concepts are applied to iterative decoders for parallel concatenated codes.

Accurate reliability values may be exploited to estimate transmission quality parameters, such as the bit error probability or the symbol-wise mutual information between encoder input and decoder output. The proposed method is unbiased and does not require knowledge of the transmitted data. A general framework for this kind of estimation is introduced, and the advantage of the proposed method over the conventional method is shown analytically by comparing the estimation variances. The proposed method may be used for a "blind" estimation at the receiver side or to speed up the estimation in simulations.

Symbol-by-symbol soft-output decoding may be interpreted as processing of mutual information. Assuming accurate reliability values at the input and at the output of a decoder, its decoding behavior may be characterized by information transfer functions, such as information processing characteristics (IPCs) or extrinsic information transfer (EXIT) functions. Bounds on information transfer functions are derived using the concept of bounding combined information, which is formed by combining single values of mutual information with respect to code constraints. These bounds are valid for all binary-input symmetric memoryless channels, and thus no Gaussian assumption is required, as in the original EXIT chart method. Single parity-check codes, repetition codes, and the accumulator are addressed. Based on such bounds, decoding thresholds for low-density parity-check codes are analytically determined.

Keywords: Linear binary codes, parallel concatenated codes (PCCs), serially concatenated codes (SCCs), low-density parity-check codes (LDPCCs), symbol-by-symbol soft-output decoding, iterative decoding, reliability information, parameter estimation, information processing characteristics (IPC), extrinsic information transfer (EXIT) functions, EXIT charts, information combining, bounds on information combining.

Contents

Chapter 1

Introduction

Modern communication systems aim at reliable transmission of digital data. This data may represent text, voice, images, as well as computer files or programs. Despite noisy communication channels, the data can be transmitted almost error-free by applying *channel coding*, as Shannon showed in his 1948 landmark article "A Mathematical Theory of Communication" [Sha48]. The transmitter adds redundancy to the data before transmission, and the receiver exploits this redundancy to recover the transmitted data from the received sequence; i.e., the transmitter performs channel encoding and the receiver performs channel decoding.

A block diagram for a typical system model is depicted in Fig. 1.1. (The inner structure of the channel decoder is explained below.) The source models the generation of the digital data, which may include source encoding and encryption, and the destination accepts the outputs of the channel decoder, and may perform decryption and source decoding.

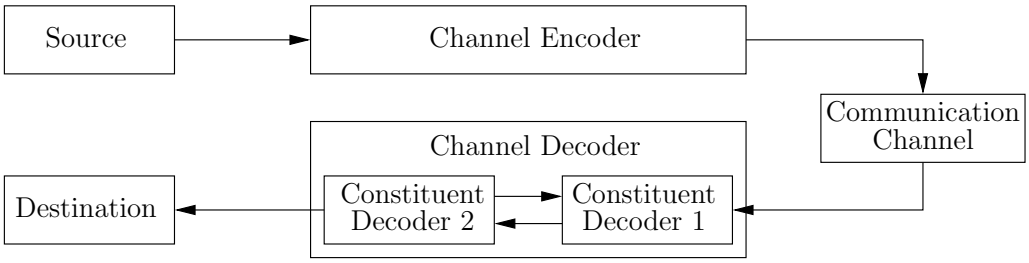


Figure 1.1: Model of a digital transmission system.

While the computational complexity of channel encoding is relatively low, the computational complexity of channel decoding is rather high, in particular, when powerful channel codes are employed. For some classes of channel codes, however, the decoding operation may be performed by two (or more) low-complexity constituent decoders that exchange decoding results in an iterative fashion, as depicted in Fig. 1.1. Examples of such codes are parallel concatenated codes, serially concatenated codes, and low-density parity-check codes.

Channel decoders may be distinguished with respect to the kind of outputs they generate. Hard-output decoders deliver estimates of the transmitted data, whereas soft-output

decoders additionally deliver information about the reliability of these estimates; this information is called *reliability information*. An important class of soft-output decoders are symbol-by-symbol soft-output decoders, which compute an estimate and reliability information for each transmitted data symbol. Such decoders are particularly suited to be used as constituent decoders of iterative decoders.

The reliability information delivered by a soft-output decoder may not be accurate, for example, when sub-optimal decoding algorithms are applied. Such mismatched reliability information usually decreases the performance of iterative decoders, since each constituent decoder assumes that the reliability information at its input is accurate when computing the soft-outputs. Thus, the question arises: how may the quality of reliability information be quantified and measured, and how may it be improved.

The performance of a transmission system is typically assessed by two parameters: (i) the probability that the estimated data symbols are erroneous and (ii) the mutual information between encoder input and decoder output. These transmission quality parameters may be estimated by the receiver without knowledge of the transmitted data when the soft-output decoder delivers accurate reliability information. In a similar way, the estimation of these parameters in simulations may be improved or speeded up by exploiting reliability information. The question is: how may the reliability information be exploited.

A soft-output decoder may be interpreted as a processor for mutual information, as both the input values and the output values carry information about the transmitted data. Following this interpretation and assuming accurate reliability information at the input and at the output of the decoder, the decoding behavior may be characterized by the mapping from mutual information associated with the decoder inputs to mutual information associated with the decoder outputs. These mappings depend on the stochastic structure of the communication channel, and therefore an interesting question is how these maps can be bounded when only the value of the input mutual information is given.

This thesis addresses the three questions raised above for transmission systems with binary channel codes and symbol-by-symbol soft-output decoders:

- How can the quality of reliability information be measured, and how can mismatched reliability information be improved or corrected?
- How can accurate reliability information be exploited for the estimation of transmission quality parameters, such as error probability or end-to-end mutual information?
- What are the bounds on the information processing behavior of soft-output decoders when assuming accurate reliability information at the inputs and at the outputs?

Both practical and theoretical aspects are discussed using methods from information theory and coding theory.

Organization of the Thesis

This thesis aims at the presentation of principles and concepts rather than an optimization of transmission systems. The main contributions are given in the form of definitions and

theorems to make them easily accessible and to provide for generality. The concepts and results are motivated and interpreted in the surrounding text, and they are further clarified and illustrated by examples.

The discussion of reliability information in channel decoding involves all parts of the transmission model. Therefore this thesis is not organized in the typical-order way, where the known fundamentals are presented in the first chapter and the new contributions in the following chapters, but in logical order. To make clear the separation between "known" and "new", each chapter starts with an overview of the new contributions. Known fundamentals on channel models and coding schemes are included in Chapter 2 and Chapter 3; the questions on reliability information, which have been raised above, are discussed in Chapter 4, Chapter 5, and Chapter 6. The notation, some properties of log-likelihood ratios, and some basic notions from information theory are provided in the appendices.

The contents of the individual chapters are outlined in the following. Further details are provided in the introduction and the summary of each chapter.

Chapter 2 addresses models for symmetric memoryless channels with discrete input alphabets and discrete or continuous output alphabets; the focus is on channels with binary input alphabets. Properties following from the symmetry of the channels are derived and discussed. The concepts presented in this chapter play a fundamental role throughout this thesis.

Chapter 3 deals with coding schemes. Linear binary channel codes and a general model for symbol-by-symbol soft-output decoding are recapitulated. Furthermore, parallel and serially concatenated codes and low-density parity-check codes are presented in a unified way. This chapter mainly summarizes known results on iteratively decodable codes.

The quality of reliability information is discussed in Chapter 4. For measuring and improving the reliability information, a criterion based on the Kullback-Leibler distance is introduced. This concept is applied to improve the iterative decoding of a parallel concatenated code.

A framework for estimating transmission quality parameters based on reliability information is introduced in Chapter 5. The presented new method requires no knowledge of the transmitted data. Furthermore, it is unbiased and has a smaller estimation variance than the conventional method. The general framework is specified to the estimation of the bit error probability and of the symbol-wise mutual information.

Chapter 6 addresses information theoretical bounds on the use of reliability information by channel decoders. Bounds on information combining are presented for single parity-check codes and for repetition codes. These results are applied to bound information transfer functions (information processing characteristics and extrinsic information transfer characteristics) for these codes and for the accumulator. Furthermore, they are used to analyze decoding thresholds of low-density parity-check codes.

The main results are summarized and conclusions are drawn in Chapter 7. Here, possible extensions and applications of the presented work are also discussed.

Parts of this thesis have been published in [LHS00, LH00a, HLS00, LH01, LC02, LH03, LHHH03, LHG04, LHH04b, LSH04, TL04, LHH04a, LHHH05a, LHHH05b].

Chapter 2

Channel Models

The class of discrete-input symmetric memoryless channels (DISMCs) comprises a large number of channel models which are often employed to analyze the performance of digital communication systems. Among these channel models are the binary symmetric channel, the binary erasure channel, the binary erasure channel, and the additive white Gaussian noise (AWGN) channel with binary antipodal signaling, called binary-input AWGN channel for short. For many coding schemes, the superchannel between encoder input and decoder output can also be modeled as a binary-input symmetric channel. In some cases, this superchannel can even be regarded as memoryless; e.g., when interleaving is applied before encoding and de-interleaving after decoding.

This chapter addresses properties of memoryless channels that have discrete input alphabets and discrete or continuous output alphabets. The focus is on binary-input channels. In addition to this, all channels are assumed to be discrete in time and to have no feedback from the output to the input. The classes of channels considered and the acronyms employed are listed in Table 2.1.

The new contributions are as follows:

(a) Subchannel indicators are introduced, which decompose channels into subchannels.

(b) Binary-input symmetric memoryless channels (BISMCs) are shown to be decomposable into binary symmetric subchannels (BSCs).

(c) Error probability profiles and mutual information profiles are introduced to characterize BISMCs with respect to their statistical properties.

Using subchannel indicators, the notion of symmetry, well-known for channels with discrete output alphabets, is extended to channels that have continuous output alphabets. The decomposition into binary symmetric channels plays a central role throughout this thesis, as it simplifies many derivations and provides insightful interpretations. A binary symmetric channel is completely defined by only one statistical parameter, such as the crossover probability, the magnitude of the log-likelihood ratio (called reliability value), or the mutual information. Accordingly, a BISMC is completely characterized by the distribution of this parameter with respect to the subchannels.

The definitions and theorems given in this chapter are formulated in a rather general way, so that a large number of cases is covered. As generality and legibility are sometimes inversely proportional, many examples are provided to clarify the underlying concepts.

Channel class	Acronym
Discrete-input memoryless channel	DIMC
Discrete memoryless channel	DMC
Discrete-input symmetric memoryless channel	DISMC
Binary-input symmetric memoryless channel	BISMC

Table 2.1: Classes of channels and their acronyms. (While a DMC is defined to have both a discrete input alphabet and discrete output alphabet, the other channels may have discrete or continuous output alphabets.)

2.1 Symmetric Channels

A memoryless channel is a probabilistic mapping from an input alphabet $\mathbb{X}$ to an output alphabet $\mathbb{Y}$. The mapping is described by a conditional probability distribution $p_{Y|X}(y|x)$ of the channel output $y \in \mathbb{Y}$ given the channel input $x \in \mathbb{X}$. This conditional probability distribution is a function of y with parameter x; it denotes a probability mass function if the output alphabet is discrete, and a probability density function if the output alphabet is continuous. As a memoryless channel describes the transition from a random variable $X \in \mathbb{X}$ to a random variable $Y \in \mathbb{Y}$, we denote it symbolically by $X \to Y$ and refer to $p_{Y|X}(y|x)$ as its transition distribution. The notation $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$ is adopted to define a memoryless channel. (Channels with memory are not addressed in this thesis; information about such channels may be found in [Gal68, CT91].)

In this thesis, we assume that the channel inputs are real values, $\mathbb{X} \subseteq \mathbb{R}$, and that the channel outputs are either real values, $\mathbb{Y} \subseteq \mathbb{R}$, or vectors of real values, $\mathbb{Y} \subseteq \mathbb{R}^L$ with $L \in \mathbb{N}^+$. Furthermore, we restrict ourselves to channels with *discrete input alphabets* $\mathbb{X}$; the output alphabets $\mathbb{Y}$ may be discrete or continuous. The binary alphabet $\mathbb{B} := \{-1, +1\}$, corresponding to the signaling commonly used for binary phase shift keying (BPSK), is of special interest. If both the input and the output alphabet are discrete, the channel is called a *discrete memoryless channel* (DMC). For a DMC, the transition distribution may be written in form of a probability matrix $[p_{Y|X}(y|x)]_{|\mathbb{X}|\times|\mathbb{Y}|}$ (each row corresponding to one input value and each column corresponding to one output value), denoted as the transition matrix.

An important concept in information theory is the notion of *symmetric channels* [Gal68, CT91, Joh92]. As indicated by the name, the transition distribution of such a channel fulfills a certain symmetry condition, and the mutual information is maximized for a "symmetric" input distribution. In precise terms, a symmetric channel achieves channel capacity (cf. Appendix D) for independent and uniformly distributed (i.u.d.) input values [CT91]. For a DMC, the symmetry condition is usually defined via properties of its transition matrix. The following definition is consistent with [Gal68, CT91, Joh92].

Definition 2.1 (Symmetric Discrete Memoryless Channel)
A *discrete memoryless channel* is called *symmetric* if the output alphabet can be partitioned into subsets in such a way that for each subset, the matrix of transition probabilities (each row corresponding to one input value and each column corresponding to one output value of the subset) has the property that each row is a permutation of each other row and each column is a permutation of each other column. A discrete memoryless channel is called *strongly symmetric* if this permutation property holds for the transition matrix (without partitioning into subsets). —

The notation introduced and the definition are illustrated by a few examples.

Example 2.1
A *binary symmetric channel* (BSC) $X \to Y$, $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$, with crossover probability ϵ is defined by a binary input alphabet $\mathbb{X} = \{x_1, x_2\}$, a binary output alphabet $\mathbb{Y} = \{y_1, y_2\}$, and the transition matrix

$$\begin{bmatrix} p_{Y|X}(y_1|x_1) & p_{Y|X}(y_2|x_1) \\ p_{Y|X}(y_1|x_2) & p_{Y|X}(y_2|x_2) \end{bmatrix} = \begin{bmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{bmatrix}.$$

The BSC with $\mathbb{X} = \mathbb{Y} = \mathbb{B}$ is illustrated in Fig. 2.1.

Since in the transition matrix, the first row is a permutation of the second row and the first column is a permutation of the second column, the BSC is strongly symmetric according to Definition 2.1. ◇

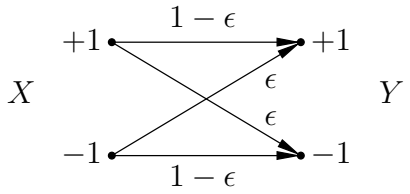


Figure 2.1: Binary symmetric channel (BSC).

Example 2.2
A *binary erasure channel* (BEC) $X \to Y$, $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$, with erasure probability δ is defined by a binary input alphabet $\mathbb{X} = \{x_1, x_2\}$, a ternary output alphabet $\mathbb{Y} = \{y_1, y_2, y_3\}$, and the transition matrix

$$\begin{bmatrix} p_{Y|X}(y_1|x_1) & p_{Y|X}(y_2|x_1) & p_{Y|X}(y_3|x_1) \\ p_{Y|X}(y_1|x_2) & p_{Y|X}(y_2|x_2) & p_{Y|X}(y_3|x_2) \end{bmatrix} = \begin{bmatrix} 1-\delta & \delta & 0 \\ 0 & \delta & 1-\delta \end{bmatrix}.$$

The output value y_2 is called erasure (in the literature often denoted by Δ). The BEC with $\mathbb{X} = \mathbb{B}$ and $\mathbb{Y} = \{-1, 0, +1\}$ is shown in Fig. 2.2.

When partitioning the output alphabet into the subsets $\mathbb{Y}_0 = \{y_2\}$ and $\mathbb{Y}_1 = \{y_1, y_3\}$, we obtain the corresponding submatrices

$$\begin{bmatrix} p_{Y|X}(y_2|x_1) \\ p_{Y|X}(y_2|x_2) \end{bmatrix} = \begin{bmatrix} \delta \\ \delta \end{bmatrix}, \qquad \begin{bmatrix} p_{Y|X}(y_1|x_1) & p_{Y|X}(y_3|x_1) \\ p_{Y|X}(y_1|x_2) & p_{Y|X}(y_3|x_2) \end{bmatrix} = \begin{bmatrix} 1-\delta & 0 \\ 0 & 1-\delta \end{bmatrix}.$$

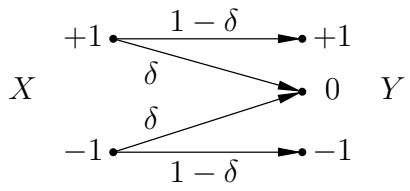


Figure 2.2: Binary erasure channel (BEC).

As these matrices fulfill the required conditions, the BEC is symmetric according to Definition 2.1. ◇

Example 2.3
A symmetric DMC with vector-valued outputs may be constructed as follows. Consider two independent BSCs $X_1 \to Y_1$ and $X_2 \to Y_2$ of which the inputs are coupled such that $X_1 = X_2$. Let $X := X_1 = X_2$ denote their common input and let $\boldsymbol{Y} := [Y_1, Y_2]$ denote the vector of channel outputs. Then, the channel $X \to \boldsymbol{Y}$ is obviously a DMC with binary input alphabet and vector-valued output alphabet.

This channel can easily be shown to be a symmetric channel according to Definition 2.1. Due to the similarity to parallel concatenated codes [BM96b], we say that the two BSCs are parallel concatenated, and we call $X \to \boldsymbol{Y}$ a *parallel concatenated channel*. ◇

The concept of symmetric DMCs is now extended to channels with continuous (possibly vector-valued) output alphabets. We call these channels discrete-input memoryless channels (DIMCs). First we introduce the notion of subchannels, and then we define symmetric channels via strongly symmetric subchannels.

Definition 2.2 (Subchannel Indicator)
A random variable is called a *subchannel indicator* of a discrete-input memoryless channel if it is a function of the channel output and statistically independent of the channel input.

—

A subchannel indicator allows to split the probabilistic mapping from channel input to channel output into two subsequent steps. For illustration, consider a DIMC $X \to Y$, $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$. Assume that $A \in \mathbb{A}$ is a subchannel indicator for this channel, and let $p_A(a)$ denote the distribution of A, i.e., the probability mass function for discrete $\mathbb{A}$ or the probability density function for continuous $\mathbb{A}$. Let further $f : \mathbb{Y} \to \mathbb{A}$ denote the function mapping Y to A. Notice that A induces a partition of the output alphabet $\mathbb{Y}$ into the subsets

$$\mathbb{Y}(a) = \{y \in \mathbb{Y} : f(y) = a\}, \tag{2.1}$$

$a \in \mathbb{A}$.

In the first step, the value of the subchannel indicator, say $A = a$, is drawn, independently from the channel input (cf. Definition 2.2). Since a subchannel indicator is a function of the channel output, the value a determines the set of possible output values,

$\mathbb{Y}(a)$. (In the case of a DMC, this corresponds to a set of column vectors of the transition matrix.) In the second step, the channel output is drawn from this subset.

The mathematical justification of this interpretation is given by the following equation chain:

$$\begin{aligned} p_{Y|X}(y|x) &= p_{Y,A|X}(y,a|x) \\ &= p_{A|X}(a|x) \cdot p_{Y|X,A}(y|x,a) \\ &= p_A(a) \cdot p_{Y|X,A}(y|x,a). \end{aligned}$$

In the first line, we have used the fact that A is a function of Y, and in the last line, we have used the fact that A is independent from X. Accordingly, the transition distribution of the channel $X \to Y$ is factorized into the distribution of A, $p_A(a)$, and the conditional transition distribution $p_{Y|X,A}(y|x,a)$ depending on the value a. This motivates to interpret the channel defined by $p_{Y|X,A}(y|x,a)$ as a subchannel, symbolically denoted by $X \to Y|A=a$.

Definition 2.3 (Decomposition of DIMCs)
Let $X \to Y$, $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$, denote a discrete-input memoryless channel. Let further $A \in \mathbb{A}$ denote a subchannel indicator of this channel, and let $\mathbb{Y}(a)$, $a \in \mathbb{A}$, denote the partition of $\mathbb{Y}$ induced by A (cf. Equ. (2.1)).

(a) For each $a \in \mathbb{A}$, the channel $X \to Y|A=a$, $(\mathbb{X}, \mathbb{Y}(a), p_{Y|X,A}(y|x,a))$, is called a *subchannel* of $X \to Y$.

(b) The set of subchannels $X \to Y|A=a$, $a \in \mathbb{A}$, is called a *decomposition* of $X \to Y$ induced by A.

(c) A channel is called a *minimal channel* if it cannot be further decomposed. A decomposition is called a *maximal decomposition* if all subchannels are minimal.

—

For convenience, we use the abbreviation "subchannel $A=a$" to refer to the subchannel $X \to Y|A=a$. Based on the above definition, it is straight forward to give a general definition of a symmetric channel.

Definition 2.4 (Symmetric Discrete-Input Memoryless Channel)
A *discrete-input memoryless channel* (with discrete or continuous output alphabet) is called *symmetric* if it can be decomposed into strongly symmetric discrete memoryless channels. —

As for symmetric DMCs, the capacity of symmetric DIMCs is achieved for i.u.d. inputs.

As a discrete-input symmetric memoryless channel (DISMC) (discrete or continuous output alphabet) is a generalization of a DMC (discrete input and discrete output alphabet), the definition of symmetry for DIMCs (Definition 2.4) must imply the definition of symmetry for DMCs (Definition 2.1). The following example illustrates the fact that the two definitions are consistent. Furthermore, the definitions introduced are recapitulated.

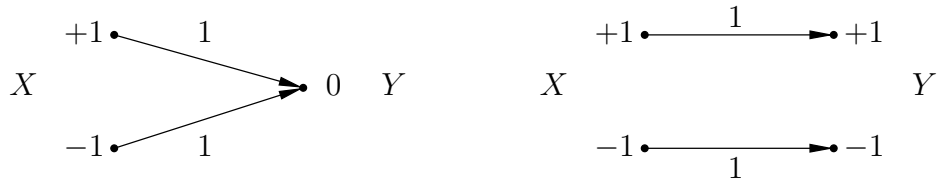
Example 2.4
Consider the BEC $X \to Y$, $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$, with erasure probability δ. Let $\mathbb{X} = \mathbb{B}$ and $\mathbb{Y} = \{-1, 0, +1\}$, as depicted in Fig. 2.2. In Example 2.2, it was shown that this channel is symmetric according to Definition 2.1. Here, we show that it is also symmetric according to Definition 2.4 (as it should be). First, we guess a function on the output alphabet, which leads to a subchannel indicator for the BEC. Then, we determine the induced decomposition into subchannels and check if the subchannels are strongly symmetric.

Let the function $f : \mathbb{Y} \to \mathbb{A} = \{0, 1\}$ be defined as $f(y) := \text{abs}(y)$. The random variable $A = f(Y)$, $A \in \mathbb{A}$, can easily be seen to be a subchannel indicator of the BEC (cf. Definition 2.2). On the one hand, A is a function of Y by definition; on the other hand, A is independent of X, because $p_{A|X}(a|x) = p_A(a)$, as can easily be seen. The corresponding partition of the output alphabet $\mathbb{Y}$ is given by the two subsets $\mathbb{Y}(0) = \{0\}$ and $\mathbb{Y}(1) = \{-1, +1\}$ (cf. Equ. (2.1)).

Now, consider the decomposition of the BEC induced by A (cf. Definition 2.3). For $A = 0$ and $A = 1$, we have the subchannels $X \to Y|A = 0$, $(\mathbb{X}, \mathbb{Y}(0), p_{Y|X,A}(y|x, 0))$, and $X \to Y|A = 1$, $(\mathbb{X}, \mathbb{Y}(1), p_{Y|X,A}(y|x, 1))$, with the transition matrices

$$\begin{bmatrix} p_{Y|X,A}(0|+1,0) \\ p_{Y|X,A}(0|-1,0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} p_{Y|X,A}(-1|+1,1) & p_{Y|X,A}(+1|+1,1) \\ p_{Y|X,A}(-1|-1,1) & p_{Y|X,A}(+1|-1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

respectively. Since the two subchannels are strongly symmetric, the BEC is symmetric according to Definition 2.4. The two subchannels are depicted in Fig. 2.3. ◇



(a) $A = 0$ (b) $A = 1$

Figure 2.3: Subchannels of the binary erasure channel (BEC), which is shown in Fig. 2.2.

When comparing Example 2.4 to Example 2.2, we see that the decomposition of a DISMC into strongly symmetric subchannels, required by Definition 2.4, corresponds exactly to the grouping of output values, required by Definition 2.1. Thus, it can easily be seen that Definition 2.4 generalizes Definition 2.1.

With this generalization, we intended to obtain a definition of symmetry that is also applicable for channels with continuous output alphabets. In order to show that the given definition is reasonable, we consider the AWGN channel with binary input alphabet.

Example 2.5
The AWGN channel with BPSK mapping $X \to Y$, $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$, is defined by $\mathbb{X} = \mathbb{B}$, $\mathbb{Y} = \mathbb{R}$, and

$$Y = X + N$$

with

$$p_N(n) = \frac{1}{\sqrt{2\pi\sigma_N^2}} \exp\Big(-\frac{n^2}{2\sigma_N^2}\Big), \tag{2.2}$$

$n \in \mathbb{R}$, denoting the probability density function of the Gaussian noise[1] N. Usually, we define $\sigma_N^2 = E_s/(2N_0)$, where E_s denotes the signal energy per channel input symbol and N_0 denotes the single-sided noise power density. We refer to this channel as *binary-input AWGN channel* (BI-AWGNC).

The BI-AWGNC is shown in Fig. 2.4 in two different ways: Fig. 2.4(a) is the usual representation, whereas Fig. 2.4(b), which is more of an illustration, is more appropriate for our purposes and follows the representation of the BSC and the BEC. Since the output alphabet is continuous, the values of Y are depicted as the axis of real values, and since transitions to all values of Y are possible, the transitions are symbolically depicted as curved arrows. The transition distribution is shown on the right-hand side.

To show that this channel is symmetric, we may apply the subchannel indicator $A \in \mathbb{A}$ defined by $A = \mathrm{abs}(Y)$. The corresponding partition of the output alphabet is given by the subsets $\mathbb{Y}(0) = \{0\}$ and $\mathbb{Y}(a) = \{-a, +a\}$, $a \in \mathbb{R}^+$. Notice that the number of subsets is infinite. As an example, the subset $\mathbb{Y}(5) = \{-5, +5\}$ is marked in the figure. The subchannels resulting from decomposition induced by A are similar to the subchannels in the previous example. The subchannel $A = 0$ has only one possible output element; thus, it is trivially strongly symmetric. The subchannels $A = a$ for $a \in \mathbb{R}^+$ can easily be seen to be BSCs (see subchannel $A = 5$ in Fig. 2.4(b)); thus, these subchannels are also strongly symmetric. (Notice that $p_A(a)$ is the probability density function of the subchannel indicators and thus of the subchannels.) As we have a decomposition into strongly symmetric subchannels, the binary-input AWGN channel is symmetric according to Definition 2.4. ◇

From Example 2.5, one is tempted to suggest that the condition of symmetry in Definition 2.4 may be equivalent to the much simpler condition:

$$p_{Y|X}(y|x) = p_{Y|X}(-y| - x) \quad \text{for } x \in \mathbb{X} \text{ and } y \in \mathbb{Y}. \tag{2.3}$$

Since $A := \mathrm{sgn}(Y)$ is a subchannel indicator in this case, the condition (2.3) is sufficient for symmetric according to Definition 2.4. However, it is not necessary, as shown by the following simple example. Consider two DIMCs $X \to Y$ and $X \to Y'$, related by $Y' = Y + 1$; let $X \to Y$ be symmetric. Then, $X \to Y'$ is symmetric by Definition 2.4 (as it should be), but it is not symmetric according to (2.3). However, we conjecture that each DISMC can be transformed such that (2.3) is fulfilled by applying a bijective mapping to the channel inputs and to the channel outputs. For DISMCs with binary-inputs, this

[1]The letter N is used to denote the code word length and to denote the random variable for the channel noise. The meaning becomes clear from the context.

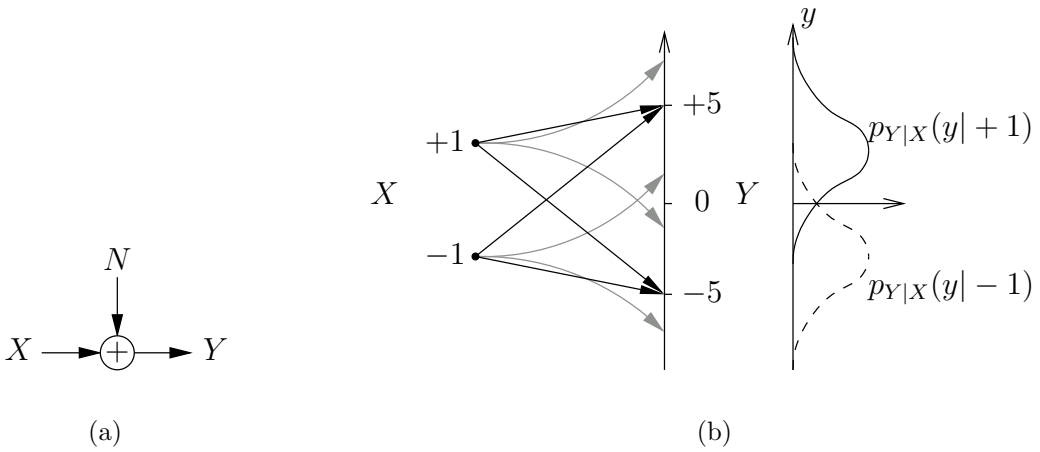


Figure 2.4: Binary-input AWGN channel (BI-AWGNC).

transformation may be performed by mapping the channel inputs to $\mathbb{B}$ and the channel outputs to their corresponding conditional log-likelihood ratios (LLRs). This is addressed in Lemma 2.2. (The lemma is provided in the following section, as the decomposition into BSCs, introduced therein, allows for a simple proof.)

To conclude the discussion of discrete-input symmetric memoryless channels (DISMCs), we remark that any subchannel of a DISMC is again a DISMC. In the remaining part of this chapter, we focus on DISMCs with binary input alphabets.

2.2 Decomposition into BSCs

The maximal decomposition of a binary-input symmetric memoryless channel (BISMC) is of special interest, because each resulting subchannel is a binary symmetric channel (BSC) or can be interpreted as an equivalent BSC. In this section, we discuss such a decomposition.

A BEC with erasure probability 1 and a BSC with crossover probability 1/2 cannot be used for information transmission, because their capacities are zero. As a BISMC with only one output value corresponds to a BEC with erasure probability 1, we may introduce the following definition without loss of generality.

Definition 2.5
A BEC with erasure probability 1 is called equivalent to a BSC with crossover probability 1/2. —

In Example 2.4 and Example 2.5, we considered the decomposition of a BEC and of a binary-input AWGN channel into strongly symmetric subchannels. In both examples, the subchannels turned out to be either BSCs or channels having only one possible output value ($Y = 0$). Therefore, *all* subchannels are BSCs. In the sequel, we generalize this observation.

Lemma 2.1
A minimal subchannel of a binary-input symmetric memoryless channel has an output alphabet with either one or two elements.

Proof. Let $X \to Y$ denote a BISMC with output alphabet $\mathbb{Y}$. We show by contradiction that there are no minimal subchannels $X \to Y|A = a$ with output alphabet $\mathbb{Y}(a)$ such that $\mathbb{Y}(a)$ contains three or more elements.

Assume that $\mathbb{Y}(a)$ contains more than two elements, say n, and let $[p_{Y|X,A}(y|x,a)] \in [0,1]^{2\times n}$ denote the transition matrix of subchannel $A = a$. Since the subchannel is strongly symmetric, each column of the transition matrix is a permutation of each other column. Without loss of generality, assume that we have n_1-times column $[p_1, p_2]^\mathsf{T}$ and n_2-times column $[p_2, p_1]^\mathsf{T}$, where $n_1 + n_2 = n$. As the entries of each row add up to one,

$$\sum_{y\in\mathbb{Y}(a)} p_{Y|X,A}(y|x_1,a) = 1$$

for $x \in \mathbb{X}$, and we have for the first and the second row

$$\begin{aligned} n_1p_1 + n_2p_2 &= 1, \\ n_1p_2 + n_2p_1 &= 1, \end{aligned}$$

respectively. Combining these two equations, we obtain

$$\begin{aligned} & n_1p_1 + n_2p_2 &&= n_1p_2 + n_2p_1 \\ \Leftrightarrow \quad & n_1(p_1 - p_2) &&= n_2(p_1 - p_2) \\ \Leftrightarrow \quad & (n_1 - n_2)(p_1 - p_2) &&= 0. \end{aligned}$$

Thus, either $n_1 = n_2$ or $p_1 = p_2$.

For $n_1 = n_2$, the columns of the transition matrix can be grouped into pairs

$$\begin{bmatrix} p_1 & p_2 \\ p_2 & p_1 \end{bmatrix},$$

and the subchannel $X \to Y|A = a$ can be further decomposed such that each new subchannel corresponds to one of these pairs. For $p_1 = p_2$, all columns are identical, and the subchannel $X \to Y|A = a$ can be further decomposed such that each new subchannel corresponds to one column. In either case, the subchannel $X \to Y|A = a$ can be further decomposed, and thus, it is *not* minimal, in contradiction to the assumption. QED

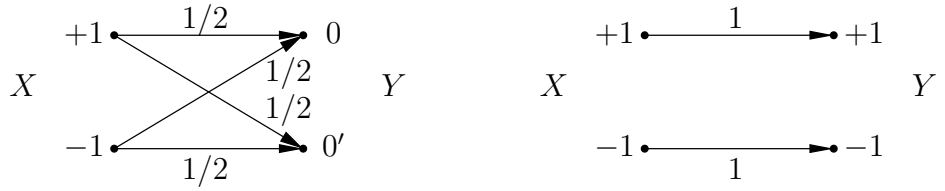
Theorem 2.1 (Decomposition of a BISMC into BSCs)
Every binary-input symmetric memoryless channel can be decomposed into subchannels that are binary symmetric channels.

Proof. Let $X \to Y$ denote a BISMC with output alphabet $\mathbb{Y}$, and consider a maximal decomposition into strongly symmetric subchannels $X \to Y|A = a$ with output alphabets $\mathbb{Y}(a)$, $a \in \mathbb{A}$. Due to the maximal decomposition, all subchannels are minimal. Thus, all sets $\mathbb{Y}(a)$ contain either one or two elements, according to Lemma 2.1. If $\mathbb{Y}(a)$ contains one element, the corresponding subchannel is equivalent to a BSC with crossover probability 1/2. If $\mathbb{Y}(a)$ contains two elements, the corresponding subchannel is a strongly symmetric DMC with binary inputs and binary outputs, and thus a BSC. QED

In imprecise terms, the theorem states that for each channel output value, we have one of the following two cases: (a) the two channel input values are equally probable (corresponding to a subchannel with one output value); (b) there is another channel output value for which the probabilities for the two channel inputs are reversed (as is the case for a BSC).

The following examples illustrate the theorem.

Example 2.6
Consider the BEC $X \to Y$ with input alphabet $\mathbb{X} = \mathbb{B}$, output alphabet $\mathbb{Y} = \{-1, 0, +1\}$, and erasure probability δ, as discussed in Example 2.2 and depicted in Fig. 2.2. Using the same subchannel indicator $A \in \mathbb{A} = \{0, 1\}$ as in Example 2.2, we have the subchannel $X \to Y|A = 0$ with one output element and the subchannel $X \to Y|A = 1$ with two output elements. The subchannel corresponding to $A = 0$ is equivalent to a BSC with crossover probability 1/2; we may define its output alphabet as $\mathbb{Y}'(0) = \{0, 0'\}$. The subchannel corresponding to $A = 1$ is already a BSC. This decomposition into BSCs is shown in Fig. 2.5. ◇



(a) $A = 0$ (b) $A = 1$

Figure 2.5: Decomposition of the binary erasure channel (BEC), shown in Fig. 2.2, into binary symmetric subchannels.

Example 2.7
Consider the binary-input AWGN channel $X \to Y$ with $\mathbb{X} = \mathbb{B}$, $\mathbb{Y} = \mathbb{R}$, and $p_{Y|X}(y|x) = p_N(y - x)$ with $p_N(n)$ denoting the Gaussian distribution of the noise, given in (2.2). As shown in Example 2.5, $A := \mathrm{abs}(Y)$ is a subchannel indicator. For $A > 0$, the subchannels are BSCs. For $A = 0$, the subchannel has only one output element and is equivalent to a BSC with crossover probability 1/2. Thus, we have a decomposition into BSCs. ◇

Theorem 2.1 is applied in the following lemma, which addresses a question on the symmetry condition (2.3) raised in the previous section. (Notice the definition of LLRs in Appendix C.)

Lemma 2.2
Consider a binary-input symmetric memoryless channel $X \to Y$ with input alphabet $\mathbb{B}$ and output alphabet $\mathbb{Y}$. Consider further the mapping from the channel outputs Y to the log-likelihood ratios $Z \in \mathbb{R}$ defined by $z := L(X|Y = y)$ for i.u.d. channel inputs X.

Then, the channel $X \to Z$ is a binary-input symmetric memoryless channel that fulfills the symmetry condition

$$p_{Z|X}(z|x) = p_{Z|X}(-z|-x)$$

for all $x \in \mathbb{B}$ and $z \in \mathbb{R}$.

Proof. The channel $X \to Y$ is assumed to be symmetric, and thus, there is a subchannel indicator that decomposes it into BSCs. Let this subchannel indicator be denoted by A. Consider a subchannel $X \to Y|A = a$, and let the LLRs corresponding to the two channel outputs be denoted by z_1 and z_2. Since the subchannel is a BSC, we have $z_1 = -z_2$ and thus

$$p_{Z|X,A}(z|x,a) = p_{Z|X,A}(-z|-x,a)$$

for all $x \in \mathbb{B}$ and $z \in \{z_1, z_2\}$. This consideration holds for every subchannel, and therefore

$$p_{Z|X}(z|x) = p_{Z|X}(-z|-x)$$

for all $x \in \mathbb{B}$ and $z \in \mathbb{R}$. This condition is sufficient for the symmetry of the channel $X \to Z$, and so we have proved the lemma. QED

The concept of decomposing a BISMC into BSCs using an appropriate subchannel indicator is employed in many derivations in this thesis. In the following section, it provides a means for an abstract statistical characterization of a BISMC.

2.3 Characterizing Parameters

Two important statistical parameters of a BISMC are the error probability and the mutual information for independent and uniformly distributed (i.u.d.) input values. A more detailed statistical characterization is provided by the error probability profile and the mutual information profile, which are introduced in this section.

For the definition of the error probability, we have to consider the estimation of the channel inputs on the basis of the channel outputs. Consider a BISMC $X \to Y$ defined by $(\mathbb{B}, \mathbb{Y}, p_{Y|X}(y|x))$. (The channel outputs may be vector-valued.) A function[2] $\mathrm{dec} : \mathbb{Y} \to \mathbb{X}$ is called an estimator for the channel input, and $\hat{x} := \mathrm{dec}(y)$ is the estimate of the channel input. Two optimal estimators are the maximum-likelihood estimator and the maximum a-posteriori estimator. The *maximum-likelihood (ML) estimator* is defined by the rule:

$$\mathrm{dec}^{\mathsf{ML}}(y) := \begin{cases} +1 & \text{for } p_{Y|X}(y|+1) > p_{Y|X}(y|-1), \\ -1 & \text{for } p_{Y|X}(y|+1) < p_{Y|X}(y|-1), \\ \text{randomly chosen from } \mathbb{B} & \text{for } p_{Y|X}(y|+1) = p_{Y|X}(y|-1). \end{cases}$$

Similarly, the *maximum a-posteriori (MAP) estimator* is defined by the rule:

$$\mathrm{dec}^{\mathsf{MAP}}(y) := \begin{cases} +1 & \text{for } p_{X|Y}(+1|y) > p_{X|Y}(-1|y), \\ -1 & \text{for } p_{X|Y}(+1|y) < p_{X|Y}(-1|y), \\ \text{randomly chosen from } \mathbb{B} & \text{for } p_{X|Y}(+1|y) = p_{X|Y}(-1|y). \end{cases}$$

[2]The expression dec stands for detector or decoder.

The two estimators are equivalent for i.u.d. inputs: Applying Bayes' rule,

$$p_{X|Y}(x|y) = \frac{p_X(x)}{p_Y(y)} \cdot p_{Y|X}(y|x),$$

the two criteria can be seen to be identical, since the fraction $p_X(x)/p_Y(y)$ is constant with respect to x.

The *error probability* of a BISMC $X \to Y$ is defined as the probability that, for i.u.d. channel inputs, the channel input X differs from its ML or MAP estimate:

$$P_e := \Pr(X \neq \mathrm{dec}^{\mathsf{ML}}(Y)) = \Pr(X \neq \mathrm{dec}^{\mathsf{MAP}}(Y)). \tag{2.4}$$

The definition of the error probability of BISMCs via i.u.d. inputs is motivated by two facts. First, the probability $\Pr(X \neq \mathrm{dec}^{\mathsf{ML}}(Y))$ does not depend on the input distribution. Second, the probability $\Pr(X \neq \mathrm{dec}^{\mathsf{MAP}}(Y))$ is maximal for i.u.d. inputs. Thus, the error probability of a BISMC may alternatively, but less obviously, be defined as the largest error probability for MAP estimation, maximized over all input distributions. (Notice the similarity to the alternative definition of the mutual information of a BISMC, stated below.)

The error probability does not reflect the fact that there may be channel outputs that lead to the same input estimate, but with different probabilities, or in imprecise terms, that there may be channel outputs with different reliabilities. A statistical parameter which takes this into account is the mutual information. The *mutual information* of a BISMC $X \to Y$ is defined as the mutual information between channel input X and channel output Y for i.u.d. inputs:

$$I := I(X;Y)\Big|_{p_X(x)=1/2}. \tag{2.5}$$

As in the case of the error probability, the definition via i.u.d. inputs may be motivated as follows. Since for a symmetric channel, the capacity is achieved for i.u.d. inputs, the value I, defined above, is equal to the capacity of the BISMC. Thus, the mutual information of a BISMC may alternatively be defined as the largest mutual information between channel input and channel output, maximized over all input distributions. (Notice the similarity to the alternative definition of the error probability of a BISMC, stated above.)

The error probability, the mutual information, and their relation are illustrated in two examples. Let

$$h(\rho) := -\rho \operatorname{ld} \rho - (1-\rho) \operatorname{ld}(1-\rho),$$

$\rho \in [0,1]$, denote the binary entropy function, and let $h^{-1}(\iota)$, $\iota \in [0,1]$, denote the inverse of $h(\rho)$ for $\rho \in [0,1/2]$ (cf. Appendix D).

Example 2.8
For the BSC with crossover probability ϵ, see Example 2.1, the error probability is (trivially) given by $P_e = \epsilon$, and the mutual information is given by

$$I = 1 - h(\epsilon). \tag{2.6}$$

Thus, we have the relations $I = 1 - h(P_e)$ and $P_e = h^{-1}(1-I)$. ◇

Example 2.9
For the BEC with erasure probability δ, see Example 2.2, the error probability is given by $P_e = \delta/2$, and the mutual information is given by

$$I = 1 - \delta. \tag{2.7}$$

Thus, we have the relations $I = 1 - P_e/2$ and $P_e = (1 - I)/2$. $\diamond$

The error probability and the mutual information of a BISMC are closely related. Given one of the two parameters, an upper and a lower bound for the other parameter can be given [HH03]. This is addressed at the end of this section.

A BISMC can be decomposed into subchannels that are BSCs, according to Theorem 2.1, and the statistical properties of a BSC are completely defined by only one statistical parameter. Combining these two facts, we may characterize a BISMC by the distribution of a statistical parameter with respect to the subchannels. In the sequel, we investigate this concept and start with statistical parameters of a BSC. If not otherwise stated, we assume i.u.d. channel inputs.

A BSC $X \to Y$, $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$, is usually specified by the crossover probability

$$\epsilon := \Pr(X \neq \mathrm{dec}^{\mathsf{ML}}(Y));$$

for formal reasons we define it via the error probability. Equivalently, we may use the mutual information resulting from i.u.d. inputs,

$$j := I(X;Y).$$

As a third statistical parameter, we consider the *reliability value*[3] defined as the magnitude of the log-likelihood ratio (see Appendix C) for i.u.d. channel inputs:

$$\lambda := |L(X|Y = y)|. \tag{2.8}$$

(Notice that for i.u.d. channel inputs, $|L(Y = y|X)| = |L(X|Y = y)|$.) Reliability values are discussed in detail in Chapter 4. The channel parameters ϵ, j, and λ are, of course, functions of the transition probabilities $p_{Y|X}(y|x)$, and there are one-to-one mappings between the parameters:

$$\begin{aligned} \epsilon &= h^{-1}(1-j) = \frac{1}{1+\exp(\lambda)}, \\ j &= 1 - h(\epsilon) = 1 - h\Big(\frac{1}{1+\exp(\lambda)}\Big), \\ \lambda &= \ln\frac{1-\epsilon}{\epsilon} = \ln\frac{1-h^{-1}(1-j)}{h^{-1}(1-j)}. \end{aligned} \tag{2.9}$$

Remark 2.1
For a BSC, the reliability value λ is proportional to the negative derivative of the mutual information j with respect to the crossover probability ϵ; to be precise, $dj/d\epsilon = -\lambda/\ln 2$.

[3] In [HOP96], the value λ/y is called the reliability value of the channel, assuming that the transition distribution fulfills the symmetry condition $p_{Y|X}(y|x) = p_{Y|X}(-y| - x)$.

Consider now a BISMC $X \to Y$, $(\mathbb{X}, \mathbb{Y}, p_{Y|X}(y|x))$, with a subchannel indicator $A \in \mathbb{A}$ inducing a maximal decomposition, i.e., a decomposition into BSCs. Since each subchannel $A = a$ is a BSC, we may assign a crossover probability, a mutual information value, and an reliability value to each subchannel. For this purpose, we define the following functions (assuming i.u.d. inputs):

$$\begin{aligned} f_{\mathcal{E}} : \mathbb{A} &\to \mathbb{E} \\ a &\mapsto \Pr(X \neq \mathrm{dec}^{\mathsf{ML}}(Y)|A = a), \\ f_J : \mathbb{A} &\to \mathbb{J} \\ a &\mapsto I(X;Y|A = a), \\ f_\Lambda : \mathbb{A} &\to \mathbb{L} \\ a &\mapsto |L(X|Y = y, A = a)|. \end{aligned} \tag{2.10}$$

As the subchannel indicator A is a random variable, we may also regard the function values as random variables.

Definition 2.6 (Indicators and Profiles for BISMCs)
Let $X \to Y$ denote a BISMC, and let $A \in \mathbb{A}$ denote a subchannel indicator that induces a maximal decomposition (i.e., a decomposition into BSCs). Using the functions defined in (2.10), the random variables

$$\begin{aligned} \mathcal{E} &:= f_{\mathcal{E}}(A) \in \mathbb{E}, \\ J &:= f_J(A) \in \mathbb{J}, \\ \Lambda &:= f_\Lambda(A) \in \mathbb{L} \end{aligned}$$

are called *error probability indicator*, *mutual information indicator*, and *reliability value indicator*, respectively. Their distributions $p_{\mathcal{E}}(\epsilon)$, $p_J(j)$, and $p_\Lambda(\lambda)$, denoting the probability mass function for discrete $\mathbb{A}$ and the probability density function for continuous $\mathbb{A}$, are called *error probability profile*, *mutual information profile*, and *reliability value profile*, respectively. —

These profiles represent an abstract statistical characterization of a BISMC: neither the input alphabet nor the output alphabet is relevant, and subchannels (BSCs) that have the same error probability (mutual information, reliability value) are not distinguished. For decoding, only the probability or probability density $p_{Y|X}(y|x)$ associated with a channel output y is important, but not the value of the channel output itself. Thus, the profiles defined above completely represent the statistical properties of a BISMC with respect to the capability of information transmission. Due to the one-to-one mappings between the three parameters given in (2.9), the three profiles provide equivalent statistical characterizations of a BISMC.

Example 2.10
Consider a binary symmetric erasure channel (BSEC) $X \to Y$ with input alphabet $\mathbb{B}$, output alphabet $\mathbb{Y} = \{-1, 0, +1\}$, crossover probability ρ, and erasure probability δ. This channel is defined by the transition probabilities

$$p_{Y|X}(y|x) = \begin{cases} 1 - \rho - \delta & \text{for } y = x, \\ \delta & \text{for } y = 0, \\ \rho & \text{for } y \neq x, \end{cases}$$

$x \in \mathbb{X}$. This BSEC is depicted in Fig. 2.6.

A maximal decomposition of the BSEC is induced by the subchannel indicator $A := \mathrm{abs}(Y)$, $A \in \mathbb{A} = \{0, 1\}$. The subchannel $X \to Y|A = 0$ has the output alphabet $\mathbb{Y}(0) = \{0\}$ and may be converted into an equivalent BSC that has crossover probability $f_{\mathcal{E}}(0) = 1/2$, see Definition 2.5; the probability for this subchannel is $p_A(0) = \delta$. The subchannel $X \to Y|A = 1$ has the output alphabet $\mathbb{Y}(1) = \{-1, +1\}$ and is a BSC with crossover probability $f_{\mathcal{E}}(1) = \rho/(1-\delta)$; the probability for this subchannel is $p_A(1) = 1 - \delta$.

The (discrete) profiles of the BSEC are given as follows.

- Error probability profile:

$$p_{\mathcal{E}}(\epsilon) = \begin{cases} 1-\delta & \text{for } \epsilon = \dfrac{\rho}{1-\delta}, \\ \delta & \text{for } \epsilon = \dfrac{1}{2}. \end{cases}$$

- Mutual information profile:

$$p_J(j) = \begin{cases} 1-\delta & \text{for } j = 1 - h\Big(\dfrac{\rho}{1-\delta}\Big), \\ \delta & \text{for } j = 0. \end{cases}$$

- Reliability value profile:

$$p_\Lambda(\lambda) = \begin{cases} 1-\delta & \text{for } \lambda = \ln \dfrac{1-\rho-\delta}{\rho}, \\ \delta & \text{for } \lambda = 0. \end{cases}$$

$\diamond$

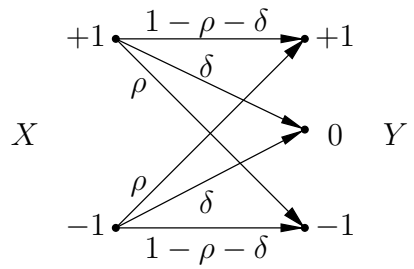


Figure 2.6: Binary symmetric erasure channel (BSEC).

Example 2.11
Consider the binary-input AWGN channel $X \to Y$ from Example 2.5: $X \in \mathbb{B}$, $Y = X + N \in \mathbb{R}$; the noise N is Gaussian distributed according to (2.2) and has variance $\sigma_N^2 = E_s/(2N_0)$. As shown in Example 2.7, the subchannel indicator $A := \mathrm{abs}(Y)$ induces a decomposition into BSCs.

Since $L(X|Y = y) = 2/\sigma_N^2 \cdot y$ [HOP96], the reliability value indicator Λ and the subchannel indicator A are related as

$$\Lambda = \frac{2}{\sigma_N^2} \cdot A.$$

Based on this relation, the error probability indicator $\mathcal{E}$ and the mutual information indicator J can be computed using (2.10). $\diamond$

The following theorem relates the error probabilities and the mutual information values of the subchannels to error probability and the mutual information of the BISMC, respectively. The proof of the theorem makes use of the concept of subchannel indicators.

Theorem 2.2 (Expectation of Indicators)
For a BISMC, the error probability P_e is the expected value of the error probability indicator $\mathcal{E}$, and the mutual information I is the expected value of the mutual information indicator J:

$$P_e = \mathrm{E}\{\mathcal{E}\} \quad \textit{and} \quad I = \mathrm{E}\{J\}.$$

Proof. Let A denote a subchannel indicator inducing a decomposition of the BISMC into BSCs. We assume that A is continuous, so that $p_A(a)$ denotes the probability density function. (For discrete A, the integrals have to be replaced by sums.)

For proving the first part, we write the following chain of equalities:

$$\begin{aligned}
P_e &\overset{(a)}{=} \Pr(X \neq \mathrm{dec}^{\mathsf{ML}}(Y)) \\
&= \int_{a\in\mathbb{A}} \Pr(X \neq \mathrm{dec}^{\mathsf{ML}}(Y), A = a)\, da \\
&= \int_{a\in\mathbb{A}} p_A(a) \underbrace{\Pr(X \neq \mathrm{dec}^{\mathsf{ML}}(Y)|A = a)}_{f_{\mathcal{E}}(a)}\, da \\
&\overset{(b)}{=} \mathrm{E}\big\{f_{\mathcal{E}}(A)\big\} \\
&\overset{(c)}{=} \mathrm{E}\big\{\mathcal{E}\big\}.
\end{aligned}$$

The applied relations are (a) the definition of the error probability according to (2.4), (b) the definition of function $f_{\mathcal{E}}$ according to (2.10), and (c) the the error probability indicator $\mathcal{E}$ according to Definition 2.6.

For proving the second part, we apply the chain rule for mutual information:

$$I \overset{(a)}{=} I(X;Y) \overset{(b)}{=} I(X;Y,A) \overset{(c)}{=} I(X;Y|A) + I(X;A).$$

We have used (a) the definition of I according to (2.5), and (b) the fact that A is a function of Y. As A is defined to be independent from X, we have $I(X;A) = 0$. Using the definition of the mutual information indicator J, see Definition 2.6 and (2.10), we may write

$$I(X;Y|A) = \int_{a\in\mathbb{A}} p_A(a) \underbrace{I(X;Y|A = a)}_{f_J(a)}\, da = \mathrm{E}\{f_J(A)\} = \mathrm{E}\{J\},$$

which completes the proof. QED

Whereas the expectations of the error probability indicator and of the mutual information indicator have well-defined meanings, namely the error probability and the mutual information of the BISMC, the expectation of the reliability value indicator has no obvious meaning.

As mentioned above, the error probability and the mutual information of a BISMC are closely related. To be precise, given one of the two values, bounds for the other one can be given, as stated and proven in [HR70, HH03]. Here, we use another method of proof using the concept of error probability indicators and mutual information indicators.

Theorem 2.3 (Relation between Error Probability and Mutual Information) *For a BISMC with error probability P_e and mutual information I, the following two inequalities hold:*

$$\begin{aligned} h^{-1}(1-I) \quad &\leq \quad P_e \quad \leq \quad \tfrac{1}{2}(1-I), \\ 1-h(P_e) \quad &\leq \quad I \quad \leq \quad 1-2P_e. \end{aligned}$$

Proof. First part: Using Theorem 2.2 and the conversion from mutual information to error probability according to (2.9), we obtain

$$P_e = \mathrm{E}\{\mathcal{E}\} = \mathrm{E}\{h^{-1}(1-J)\}.$$

Now, we exploit two properties of the inverse of the binary entropy function, h^{-1}. First, the function h^{-1} is convex-$\cup$; thus we can apply Jensen's inequality [CT91] and obtain

$$\mathrm{E}\{h^{-1}(1-J)\} \geq h^{-1}\big(\mathrm{E}\{1-J\}\big) = h^{-1}(1-I).$$

Second, the function h^{-1} is lower-bounded as $h^{-1}(\iota) \leq \frac{1}{2}\iota$; using this bound, we obtain

$$\mathrm{E}\{h^{-1}(1-J)\} \leq \mathrm{E}\{\tfrac{1}{2}(1-J)\} = \tfrac{1}{2}(1-I).$$

Second part: Inverting the left-hand relation of the first part, $h^{-1}(1-I) \leq P_e$, gives $1-h(P_e) \leq I$. (Notice that the binary entropy function is monotonically increasing for arguments in $[0, \frac{1}{2}]$.) Inverting the right-hand relation of the first part, $P_e \leq \frac{1}{2}(1-I)$, gives $I \leq 1-2P_e$. QED

Alternatively, we may start the proof with

$$I = \mathrm{E}\{J\} = \mathrm{E}\{1-h(\mathcal{E})\}$$

and exploit the convexity of $1-h(.)$ in a similar way as above. The bounds hold with equality when the channel is a BSC or a BEC (cf. Example 2.8 and Example 2.9).

Mutual information profiles are applied in Chapter 6 to proof bounds on combining of mutual information values. In particular, this concept provides an elegant method to explain the bounds on information combining for single parity check codes and for repetition codes (cf. Section 6.2.4).

2.4 Summary

In this chapter, we have discussed the symmetry of memoryless channels with discrete input alphabet and discrete or continuous output alphabet. The notion of a subchannel indicator has been introduced, and a definition of a subchannel has been given. As has been shown, every binary-input symmetric memoryless channel (BISMC) can be decomposed into subchannels that are binary symmetric channels (BSCs). Based on this property, we have defined the error probability indicator $\mathcal{E}$, the reliability value indicator Λ, and the mutual information indicator J. The distributions of these random variables, denoted as error probability profile, reliability value profile, and mutual information profile, respectively, completely characterize the statistical properties of a BISMC with respect to its capability of information transmission. A first application of these concepts has been the proof for the relation between the error probability and the mutual information of a BISMC. In Chapter 4, Chapter 5, and Chapter 6, these concepts are used to prove and interpret theorems.

Chapter 3

Channel Coding Schemes

The ingredients of a powerful coding scheme are an encoder generating a code with good distance properties and a decoder with reasonable complexity. Especially for very noisy communication channels, where maximum-likelihood (ML) decoding or near ML decoding is necessary for good performance, these two aspects are often contradictory. For example, algebraic codes have very good distance properties, but ML decoding is too complex; on the other hand, convolutional codes having moderate memory lengths can efficiently be ML decoded, but their distance properties are poor[1].

Special code constructions allow for iterative decoding schemes that achieve near ML decoding. These kinds of coding schemes are addressed in this chapter. Typically, such a code includes a small number of low-weight codewords, and the complexity of the decoder grows linearly with the code length. This trade-off between distance properties of the code and complexity of the decoder leads to powerful coding schemes for communication channels with low to medium transmission quality. In particular, these coding schemes may be constructed in such a way that the capacity of the communication channel is achieved as the code length tends to infinity.

The codes are built up by combining relatively simple constituent codes. This may be explicit in the special structure of the encoder, as in parallel and serially concatenated codes, or implicit by a special structure of the parity-check matrix of the code, as in low-density parity check codes. The decoders consist of several constituent decoders. Each constituent decoder takes into account only a subset of all code constraints. By exchanging information about info symbols[2] or code symbols between the constituent decoders in an iterative fashion, the overall decoder tries to find a solution fulfilling all constraints – this solution approximates the ML info word or the ML code word.

In this chapter, first some properties of binary linear encoders are given, and the notion of a systematically extended code is explained. Then a general model for symbol-by-symbol soft-output decoding is introduced; this model may be used for coding schemes with noniterative or iterative decoding structures. Based on this decoding model, two optimal decoding principles are reviewed: LogAPP decoding and MaxLogAPP decoding[3].

[1] Notice that convolutional codes are good with respect to the decoding delay [JZ99].

[2] Throughout this thesis, the term "info symbol" is prefered to the term "information symbol" to avoid any confusion with the term "mutual information" or the general term "information".

[3] Both decoding principles can equivalently be formulated with probabilities. The first one is then

Using these encoder and decoder descriptions, three types of iteratively decodable codes are addressed: parallel concatenated codes, serially concatenated codes, and low-density parity-check codes. For each code, the encoder and the iterative decoder are described, the generator matrix and parity-check matrices are given, and the relation between the parity-check matrix and the iterative decoding scheme is discussed.

This chapter mainly deals with fundamentals of symbol-by-symbol soft-output decoding, concatenated codes, and iterative decoding, presented in a unified way. Besides that, there are also new contributions:

(a) Systematic extensions of codes are utilized to incorporate information about info symbols (often called a-priori information) in decoding algorithms.

(b) Requirements for pre-decoding soft-values, necessary for optimal operation of decoding algorithms, are made explicit.

(c) Generator matrices and parity-check matrices for parallel and serially concatenated codes are given, using only generator matrices and parity-check matrices of the constituent codes.

(d) Iterative decoders operate on a code embedding the actual code, but not on the code itself. This prerequisite for iterative decoding is discussed in detail using the structures of the parity-check matrices.

(e) The structures of iteratively decodable codes may be utilized to find iterative decoders for other codes. This possible application is discussed.

In the following section, we start with some properties of linear codes.

3.1 Linear Binary Encoders

In this thesis, we consider only binary codes. The binary symbols are represented in either binary field $\mathbb{F}_2 := \{0, 1\}$ or $\mathbb{B} := \{-1, +1\}$. The representation over $\mathbb{B}$ is more suited to describe the transmission, whereas the representation over $\mathbb{F}_2$ is more suited to describe encoding and code properties. For $\mathbb{F}_2$, addition and multiplication are those operations in the modulo-two arithmetic; for $\mathbb{B}$, addition and multiplication are defined via the equivalent operations in $\mathbb{F}_2$, using the one-to-one mapping[4]

$$\begin{aligned} \mathrm{bpsk} : \mathbb{F}_2 &\to \mathbb{B} \\ 0 &\mapsto +1 \\ 1 &\mapsto -1. \end{aligned} \tag{3.1}$$

The inverse mapping is denoted by bpsk^{-1}. By convention, we write symbols and words over $\mathbb{B}$ as a and $\boldsymbol{a}$, and the corresponding symbols and words over $\mathbb{F}_2$ as $\breve{a}$ and $\breve{\boldsymbol{a}}$. Symbols $a \in \mathbb{B}$ and $\breve{a} \in \mathbb{F}_2$ are related by

$$\breve{a} = \mathrm{bpsk}^{-1}(a),$$

commonly called APP decoding.

[4]This mapping is commonly used for binary phase shift keying (BPSK).

and words $\boldsymbol{a} = [a_0, \cdots, a_{L-1}] \in \mathbb{B}^L$ and $\breve{\boldsymbol{a}} = [\breve{a}_0, \ldots, \breve{a}_{L-1}] \in \mathbb{F}_2^L$ are related by

$$\breve{a}_i = \mathrm{bpsk}^{-1}(a_i),$$

$i = 0, 1, \ldots, L-1$.

A *linear binary encoder* for a linear binary (N, K) code is a linear mapping

$$\begin{aligned} \mathrm{enc} : \mathbb{B}^K &\rightarrow \mathbb{B}^N \\ \boldsymbol{u} &\mapsto \boldsymbol{x} \end{aligned}$$

from *info words*[5] $\boldsymbol{u} \in \mathbb{B}^K$ of length K to *code words* $\boldsymbol{x} \in \mathbb{B}^N$ of length N. As index sets for the components of the info word and the code word, we use

$$\mathcal{K} := \{0, 1, \ldots, K-1\}, \qquad \mathcal{N} := \{0, 1, \ldots, N-1\};$$

accordingly, the info word and the code word are given as

$$\boldsymbol{u} = [u_0, \ldots, u_{K-1}], \qquad \boldsymbol{x} = [x_0, \ldots, x_{N-1}].$$

The set of code words $\boldsymbol{x}$ defined by enc is called the *linear binary code*

$$\mathcal{C} := \{\boldsymbol{x} \in \mathbb{B}^N : \boldsymbol{x} = \mathrm{enc}(\boldsymbol{u}), \boldsymbol{u} \in \mathbb{B}^K\}.$$

The code rate (and also the encoder rate) is given by $R = K/N$.

A code $\mathcal{C}'$ is called an *equivalent code* of code $\mathcal{C}$ if the order of the symbols in code words $\boldsymbol{x}' \in \mathcal{C}'$ are simply rearrangements of the order in the code words $\boldsymbol{x} \in \mathcal{C}$. As there is a one-to-one correspondence between $\boldsymbol{x} \in \mathbb{B}^N$ and $\breve{\boldsymbol{x}} \in \mathbb{F}_2^N$ (cf. above), we do not distinguish between the set of code words over $\mathbb{F}_2$ and the set of code words over $\mathbb{B}$, whenever this is possible without causing ambiguity, and denote both as code $\mathcal{C}$. Thus, we have by convention

$$\boldsymbol{x} \in \mathcal{C} \quad \Leftrightarrow \quad \breve{\boldsymbol{x}} \in \mathcal{C}$$

for $\boldsymbol{x} \in \mathbb{B}^N$ and the corresponding $\breve{\boldsymbol{x}} \in \mathbb{F}_2^N$.

The encoding may be defined by means of a matrix $\boldsymbol{G} \in \mathbb{F}_2^{K \times N}$ of rank K, called *generator matrix*:

$$\breve{\boldsymbol{x}} = \breve{\boldsymbol{u}}\boldsymbol{G}. \tag{3.2}$$

Thus, we have

$$\mathcal{C} = \{\breve{\boldsymbol{x}} \in \mathbb{F}_2^N : \breve{\boldsymbol{x}} = \breve{\boldsymbol{u}}\boldsymbol{G}, \breve{\boldsymbol{u}} \in \mathbb{F}_2^K\}. \tag{3.3}$$

A matrix $\boldsymbol{H} \in \mathbb{F}_2^{M \times N}$ of rank $(N - K)$ is called a *parity-check matrix* of code $\mathcal{C}$ if

$$\breve{\boldsymbol{x}}\boldsymbol{H}^{\mathsf{T}} = \boldsymbol{0} \quad \text{for all } \breve{\boldsymbol{x}} \in \mathcal{C}, \tag{3.4}$$

where $\boldsymbol{H}^{\mathsf{T}}$ denotes the transposed matrix of $\boldsymbol{H}$. Notice that $M \geq N - K$. The conditions given by (3.4) are called *code constraints* of $\mathcal{C}$. From (3.4), we immediately have the relation

$$\boldsymbol{G}\boldsymbol{H}^{\mathsf{T}} = \boldsymbol{0}$$

[5] In this thesis, the term "info word" is used instead of "information word" to clearly distinguish it from "mutual information", which is often abbreviated by "information".

between the generator matrix and a parity-check matrix of a code. Equivalently to (3.3), we have

$$\mathcal{C} = \{\breve{\boldsymbol{x}} \in \mathbb{F}_2^N : \breve{\boldsymbol{x}}\boldsymbol{H}^{\mathsf{T}} = \boldsymbol{0}\}. \tag{3.5}$$

For decoding, not only knowledge of the code, but also knowledge of the encoding is required. An implicit description of both is given by the generator matrix. For an explicit description, we may extend each code word by the corresponding info word.

Definition 3.1 (Systematically Extended Code)
Let $\mathcal{C}$ denote a binary linear (N, K) code defined by a generator matrix $\boldsymbol{G}$. The systematically extended code

$$\mathcal{C}_{\mathsf{syxt}} := \{\breve{\boldsymbol{x}}_{\mathsf{syxt}} \in \mathbb{F}_2^{K+N} : \breve{\boldsymbol{x}}_{\mathsf{syxt}} = [\breve{\boldsymbol{u}}\ \breve{\boldsymbol{x}}], \breve{\boldsymbol{x}} = \breve{\boldsymbol{u}}\boldsymbol{G}, \breve{\boldsymbol{u}} \in \mathbb{F}_2^K\} \tag{3.6}$$

of code $\mathcal{C}$ is the $(N + K, K)$ code obtained by extending each code word of $\mathcal{C}$ by the corresponding info word. —

As $\mathcal{C}_{\mathsf{syxt}}$ implies the encoding of $\mathcal{C}$ according to $\boldsymbol{G}$, it enables an elegant description of the decoding operation of $\mathcal{C}$. Of special interest are the following subsets of $\mathcal{C}_{\mathsf{syxt}}$: For $b \in \mathbb{F}_2$, we define

$$\begin{aligned} \mathcal{C}_{\mathsf{syxt}}(\breve{u}_k = b) &:= \{[\breve{\boldsymbol{u}}\ \breve{\boldsymbol{x}}] \in \mathcal{C}_{\mathsf{syxt}} : \breve{u}_k = b\}, \\ \mathcal{C}_{\mathsf{syxt}}(\breve{x}_n = b) &:= \{[\breve{\boldsymbol{u}}\ \breve{\boldsymbol{x}}] \in \mathcal{C}_{\mathsf{syxt}} : \breve{x}_n = b\}, \end{aligned} \tag{3.7}$$

$k \in \mathcal{K}$, $n \in \mathcal{N}$. Each subset comprises all code words that have the same symbol in a certain position. These subsets are employed for defining LogAPP decoding and MaxLogAPP decoding (cf. Section 3.4).

In the sequel, some relations between the generator and the check matrix for the (original) code $\mathcal{C}$ and its systematically extended code $\mathcal{C}_{\mathsf{syxt}}$ are discussed. The generator matrix of $\mathcal{C}_{\mathsf{syxt}}$ is given by

$$\boldsymbol{G}_{\mathsf{syxt}} := \begin{bmatrix} \boldsymbol{I} & \boldsymbol{G} \end{bmatrix} \in \mathbb{F}_2^{K\times(K+N)}, \tag{3.8}$$

where $\boldsymbol{I}$ denotes the identity matrix, and a parity-check matrix is given by[6]

$$\boldsymbol{H}_{\mathsf{syxt}} := \begin{bmatrix} \boldsymbol{G}^{\mathsf{T}} & \boldsymbol{I} \end{bmatrix} \in \mathbb{F}_2^{N\times(K+N)}. \tag{3.9}$$

Notice that the encoding of $\mathcal{C}$ according to $\boldsymbol{G}$ is included in the parity-check matrix of the systematically extended code, but not in that of the original code.

A parity-check matrix of code $\mathcal{C}_{\mathsf{syxt}}$, which explicitly contains $\boldsymbol{H}$, can be derived as follows. Let $\boldsymbol{A} \in \mathbb{F}_2^{K\times N}$ denote a matrix fulfilling

$$\boldsymbol{G}\boldsymbol{A}^{\mathsf{T}} = \boldsymbol{I}, \tag{3.10}$$

such that matrix $\boldsymbol{A}^{\mathsf{T}}$ is an inverse[7] of the generator matrix $\boldsymbol{G}$. Multiplying (3.2) by $\boldsymbol{A}^{\mathsf{T}}$ from the right-hand side, we obtain

$$\breve{\boldsymbol{x}}\boldsymbol{A} = \breve{\boldsymbol{u}}\boldsymbol{G}\boldsymbol{A}^{\mathsf{T}} = \breve{\boldsymbol{u}}. \tag{3.11}$$

[6]The general rule is: $\boldsymbol{G} = [\boldsymbol{I}\ \boldsymbol{P}] \Rightarrow \boldsymbol{H} = [-\boldsymbol{P}^{\mathsf{T}}\ \boldsymbol{I}]$. See [MS88].

[7]The inverse of a generator matrix always exists, but it is not unique.

When writing the constraints given by (3.4) and (3.11) in matrix notation as

$$[\breve{\boldsymbol{u}}\ \breve{\boldsymbol{x}}] \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{H} \end{bmatrix}^{\mathsf{T}} = \boldsymbol{0} \quad \text{for all } \breve{\boldsymbol{x}}_{\mathsf{syxt}} = [\breve{\boldsymbol{u}}\ \breve{\boldsymbol{x}}] \in \mathcal{C}_{\mathsf{syxt}},$$

we can easily identify

$$\boldsymbol{H}_{\mathsf{syxt},A} := \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{H} \end{bmatrix} \in \mathbb{F}_2^{N \times (K+N)} \tag{3.12}$$

to be a parity-check matrix for the systematically extended code $\mathcal{C}_{\mathsf{syxt}}$, that has the desired property. As this matrix has rank N, no rows are redundant.

Example 3.1
Consider the repetition code of length $N = 3$ ($K = 1$), denoted by $\mathcal{R}_3$, with generator matrix $\boldsymbol{G}$ and parity-check matrix $\boldsymbol{H}$:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \qquad \boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

For the systematically extended code, the generator matrix $\boldsymbol{G}_{\mathsf{syxt}}$ and the parity-check matrix $\boldsymbol{H}_{\mathsf{syxt}}$, according to (3.8) and (3.9), are given by

$$\boldsymbol{G}_{\mathsf{syxt}} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{G} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}, \qquad \boldsymbol{H}_{\mathsf{syxt}} = \begin{bmatrix} \boldsymbol{G}^{\mathsf{T}} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

The systematically extended code over $\mathbb{B}$ is given by

$$\mathcal{R}_{3,\mathsf{syxt}} = \{[+1, +1, +1, +1], [-1, -1, -1, -1]\}.$$

In each code word, the first symbol is the info symbol, and the remaining part is the code word of the original repetition code $\mathcal{R}_3$. ◇

Example 3.2
Consider the single parity check code of length $N = 3$ ($K = 2$), denoted by $\mathcal{S}_3$, with generator matrix $\boldsymbol{G}$ and parity-check matrix $\boldsymbol{H}$:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \qquad \boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}.$$

For the systematically extended code $\mathcal{S}_{3,\mathsf{syxt}}$, the generator matrix $\boldsymbol{G}_{\mathsf{syxt}}$ and the parity-check matrix $\boldsymbol{H}_{\mathsf{syxt}}$, according to (3.8) and (3.9), are given by

$$\boldsymbol{G}_{\mathsf{syxt}} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{G} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \qquad \boldsymbol{H}_{\mathsf{syxt}} = \begin{bmatrix} \boldsymbol{G}^{\mathsf{T}} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

An inverse generator matrix, according to (3.10), is given by

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

According to (3.12), a parity-check matrix for the systematically extended code $\mathcal{S}_{3,\mathsf{syxt}}$, that explicitly contains $\boldsymbol{H}$, is obtained as

$$\boldsymbol{H}_{\mathsf{syxt},A} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{H} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

The systematically extended code over $\mathbb{B}$ is given by

$$\begin{aligned} \mathcal{C}_{\mathsf{syxt}} = \quad & \{[+1,+1,+1,+1,+1],[-1,+1,-1,-1,+1], \\ & [+1,-1,+1,-1,-1],[-1,-1,-1,+1,-1]\}. \end{aligned}$$

In each code word, the first two symbols represent the info word, and the remaining part is the code word of the original single parity check code $\mathcal{S}_3$. $\diamond$

A linear binary code can compactly be represented in a *trellis* [Wol78, McE96, Var98, LKFF98, JZ99]. To include also the encoding rule, i.e., the mapping between info words and code words, one may simply use the trellis of the corresponding systematically extended code. The complexity of the resulting trellis can usually be decreased when the systematic symbols are not placed before the original code words (as in Definition 3.1), but "spread" over the whole code word. (Notice that this code is an equivalent code of the systematically extended code.) In case of a convolutional code, the encoder provides already the structure for a "natural" construction of the code trellis [JZ99]. Further information can be found in the literature.

3.2 Decoding Model

Decoding relies on the knowledge of the underlying transmission system or, more generally, on the assumption of a particular transmission model. The latter is especially the case for constituent decoders of iterative decoding schemes. In the sequel, a framework for symbol-by-symbol soft-output decoding is introduced. This framework, depicted in Fig. 3.1, includes the assumed transmission model, comprising a linear channel encoder, binary-input symmetric memoryless channels (BISMCs) (cf. Chapter 2), and a general symbol-by-symbol soft-output decoder. Two special and very important soft-output decoding principles, LogAPP decoding[8] and MaxLogAPP decoding, are addressed in Section 3.4.

In the sequel, "symbol-by-symbol soft-output decoding" may be abbreviated with "soft-output decoding" when possible without causing ambiguity.

3.2.1 Transmission Model

The soft-output decoder assumes the following transmission model: A *binary symmetric source* (BSS) generates independent and uniformly distributed (i.u.d.) info symbols from

[8] LogAPP decoding, which uses LLRs as inputs and outputs, is equivalent to APP decoding, which uses probabilities as inputs and outputs. This thesis focuses on LogAPP decoding.

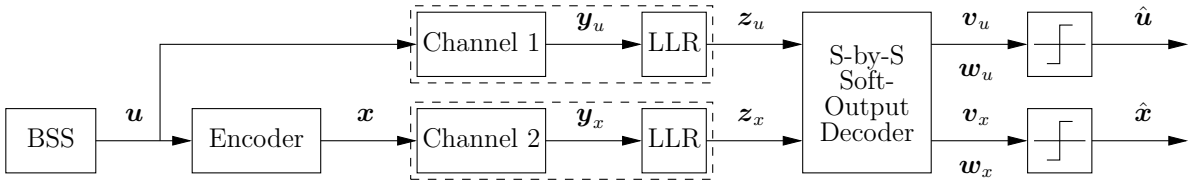


Figure 3.1: General decoding model, including a symbol-by-symbol soft-output decoder.

the alphabet $\mathbb{B}$. The *binary linear encoder* maps each info word $\boldsymbol{u}$ onto a code word $\boldsymbol{x}$ from code $\mathcal{C}$, where

$$\boldsymbol{u} = [u_0, \ldots, u_{K-1}], \qquad \boldsymbol{x} = [x_0, \ldots, x_{N-1}].$$

Notice that $[\boldsymbol{u}\,\boldsymbol{x}]$ is a code word of the systematically extended code $\mathcal{C}_{\mathsf{syxt}}$, i.e.,

$$[\boldsymbol{u}\,\boldsymbol{x}] \in \mathcal{C}_{\mathsf{syxt}}. \tag{3.13}$$

The info symbols and the code symbols are transmitted over independent *symmetric memoryless channels*, called info-symbol channel (Channel 1) and code-symbol channel (Channel 2). The channel outputs, $\boldsymbol{y}_u$ and $\boldsymbol{y}_x$, are converted to channel log-likelihood ratios[9] (LLRs), $\boldsymbol{z}_u$ and $\boldsymbol{z}_x$. (Channel LLRs are defined below.)

Based on these assumptions, the *symbol-by-symbol soft-output decoder* takes the channel LLRs for info and code symbols as pre-decoding soft-values, and it computes post-decoding soft-values for info symbols or code symbols. These may be complete post-decoding values, denoted by $\boldsymbol{v}_u$ and $\boldsymbol{v}_x$, or extrinsic post-decoding values, denoted by $\boldsymbol{w}_u$ and $\boldsymbol{w}_x$. (The difference between "complete" and "extrinsic" is addressed below.) For computing the post-decoding values, the soft-output decoder only takes into account that $[\boldsymbol{u}\,\boldsymbol{x}] \in \mathcal{C}_{\mathsf{syxt}}$. (Symbol-by-symbol soft-output decoding is discussed in detail in Section 3.2.2.) Finally, the post-decoding soft-values may be hard-decided to the estimated info word $\hat{\boldsymbol{u}}$ or the estimated code word $\hat{\boldsymbol{x}}$.

Notice that we employ the following notation: Channel outputs are denoted by y, channel LLRs are denoted by z, complete post-decoding values are denoted by v, and extrinsic post-decoding values are denoted by w. The indices indicate whether the soft-values are for info symbols, index "u", or code symbols, index "x". The index sets for the words of symbols and soft-values are

$$\mathcal{K} := \{0, 1, \ldots, K-1\}, \qquad \mathcal{N} := \{0, 1, \ldots, N-1\}.$$

The index set $\mathcal{K}$ is used for the info word and all corresponding words of soft-values, i.e., for $\boldsymbol{u}$, $\boldsymbol{y}_u$, $\boldsymbol{z}_u$, $\boldsymbol{v}_u$, $\boldsymbol{w}_u$, $\hat{\boldsymbol{u}}$. The index set $\mathcal{N}$ is used for the code word and all corresponding words of soft-values, i.e., for $\boldsymbol{x}$, $\boldsymbol{y}_x$, $\boldsymbol{z}_x$, $\boldsymbol{v}_x$, $\boldsymbol{w}_x$, $\hat{\boldsymbol{x}}$.

The given model is very general. It may be applied to *coding schemes with noniterative decoders* as well as to *coding schemes with iterative decoders*. For coding schemes with

[9] Details about log-likelihood ratios are given in Appendix C.

noniterative decoders, the code-symbol channel is a communication channel, and the info-symbol channel is only a dummy channel, over which no information can be transmitted (corresponding to infinite noise); the info-symbol channel may also be used to model the availability of a-priori information about info symbols. For coding schemes with iterative decoders, the soft-output decoder represents a constituent decoder, and accordingly, the encoder represents a (possibly virtual[10]) encoder for a constituent code. The info-symbol and the code-symbol channel may be communication channels or virtual channels within an iterative decoder. Such virtual channels are often referred to as extrinsic channels or a-priori channels in the literature on iterative decoding; in this thesis, we call them "a-priori channels", because the decoder interprets the channel outputs as a-priori values.

The info symbol channel $U \to Y_u$ (Channel 1 in Fig. 3.1), and the code symbol channel $X \to Y_x$ (Channel 2 in Fig. 3.1) are assumed to be BISMCs. The channel output corresponding to info symbol u_k is denoted by $y_{u,k}$, and the channel output corresponding to code symbol x_n is denoted by $y_{x,n}$. The words of channel outputs corresponding to the info word $\boldsymbol{u}$ and the code word $\boldsymbol{x}$ are written as

$$\boldsymbol{y}_u = [y_{u,0}, y_{u,1}, \ldots, y_{u,K-1}], \qquad \boldsymbol{y}_x = [y_{x,0}, y_{x,1}, \ldots, y_{x,N-1}],$$

respectively.

As the channels are memoryless, each channel output corresponds to exactly one info or code symbol. The LLR for a symbol given only the direct observation of this symbol is called *channel LLR*. Accordingly, the channel LLR for info symbols u_k and the channel LLR for code symbols x_n are defined as

$$\begin{aligned} z_{u,k} &:= L(U_k|Y_{u,k} = y_{u,k}) = \ln \frac{\Pr(U_k = +1|Y_{u,k} = y_{u,k})}{\Pr(U_k = -1|Y_{u,k} = y_{u,k})}, \\ z_{x,n} &:= L(X_n|Y_{x,n} = y_{x,n}) = \ln \frac{\Pr(X_n = +1|Y_{x,n} = y_{x,n})}{\Pr(X_n = -1|Y_{x,n} = y_{x,n})}, \end{aligned}$$

respectively, $k = 0, 1, \ldots, K-1$ and $n = 0, 1, \ldots, N-1$. The words of channel LLRs corresponding to the info word $\boldsymbol{u}$ and the code word $\boldsymbol{x}$ are written as

$$\boldsymbol{z}_u = [z_{u,0}, z_{u,1}, \ldots, z_{u,K-1}], \qquad \boldsymbol{z}_x = [z_{x,0}, z_{x,1}, \ldots, z_{x,N-1}],$$

respectively.

The computation of channel LLRs (LLR in Fig. 3.1) can be interpreted as a conversion of channel outputs into LLRs. Notice that the backward and the forward LLR are equal, i.e.,

$$\begin{aligned} L(U_k|Y_{u,k} = y_{u,k}) &= L(Y_{u,k} = y_{u,k}|U_k), \\ L(X_n|Y_{x,n} = y_{x,n}) &= L(Y_{x,n} = y_{x,n}|X_n), \end{aligned}$$

because (i) the info symbols are equiprobably distributed by definition, and (ii) the code symbols are equiprobably distributed due to the linear encoding of equiprobably distributed info symbols. When possible without causing ambiguity, we use the shorthand notation $L(U|y)$ for $L(U|Y = y)$.

[10]This is the case for LDPC codes.

Example 3.3
The channel LLR can easily be computed for the following three channels:

- BI-AWGNC $(\mathbb{B}, \mathbb{R}, p_{Y|X}(y|x))$ with noise variance σ_N^2:

$$L(X|y_x) = \frac{2}{\sigma_N^2} y_x.$$

- BSC $(\mathbb{B}, \mathbb{B}, p_{Y|X}(y|x))$ with crossover probability ϵ:

$$L(X|y_x) = \ln \frac{1-\epsilon}{\epsilon} y_x.$$

- BEC $(\mathbb{B}, \{-1, 0, +1\}, p_{Y|X}(y|x))$ with erasure probability δ:

$$L(X|y_x) = \begin{cases} +\infty & \text{for } y_x = +1, \\ 0 & \text{for } y_x = 0, \\ -\infty & \text{for } y_x = -1. \end{cases}$$

(The indices are omitted for convenience.) $\diamond$

The superchannel between the info symbols and their channel LLRs, $U \to Z_u$, and the superchannel between the code symbols and their channel LLRs, $X \to Z_x$, are also BISMCs, as can easily be seen.

3.2.2 Symbol-by-Symbol Soft-Output Decoding

Decoding in the conventional sense is the inversion of the mapping from the info word or the code word to the received word. The result of this kind of decoding is an estimate of the transmitted info or code word; therefore it is called *hard-output decoding*. As opposed to this, *soft-output decoding* provides not only estimates for the most probable info or code word but also some kind of reliability information. (Notice that both hard-output decoding and soft-output decoding may be based on hard or soft inputs.)

The kind of reliability information depends on the applied decoding scheme. Important and frequently used schemes are the following:

- A *bounded minimum distance decoder* [Fri94, Bos98, LC83] outputs either (i) a code word estimate or (ii) a decoding failure. Correspondingly, we have the following reliability information: (i) the estimated code word is equal to the transmitted code word; (ii) all code words are equally probable. In addition to this, the number of corrected symbol errors may be used.

- A *list decoder* [Has87, SS94, NS95] generates a list of code words, ordered with respect to their probabilities. Correspondingly, the higher the position of a code word in the list, the higher is the reliability when deciding for this code word.

- A *symbol-by-symbol soft-output decoder* provides reliability information about each info symbol or each code symbol. The optimality of the soft-outputs depends on what they are used for by the following processing stage. For example, they may be used to estimate the info or code symbols such that the symbol or the word error rate is minimized; they may also be used as soft-inputs by a subsequent decoding stage, as in iterative decoding. (Notice that these three examples lead to three different optimality criteria.) Algorithms for this kind of decoding are the BCJR algorithm, the LogAPP algorithm, the MaxLogAPP algorithm, and the soft-output Viterbi algorithm; details are provided in Section 3.4.

In the sequel, we restrict ourselves to symbol-by-symbol soft-output decoding. For convenience, we may refer to this kind of decoding by simply "soft-output decoding", when possible without causing ambiguity.

A general symbol-by-symbol soft-output decoder takes one soft-value for each info and each code symbol, and it computes one (new) soft-value for each info symbol or for each code symbol (cf. Fig. 3.1). These computations are based on the code structure. The input values of the decoder are called pre-decoding soft-values, and the output values are called post-decoding soft-values. Both kinds of soft-values are assumed to be real-valued.

In literature on iterative decoding, *pre-decoding values* are commonly called *channel values* if they come from a communication channel, and they are called *a-priori values* if they come from other constituent decoders. Constant a-priori values may also be used to model a-priori distributions of info symbols. The pre-decoding values for info symbol u_k and code symbol x_n are denoted by $z_{u,k}$ and $z_{x,n}$, respectively. The words of pre-decoding values (pre-decoding words) corresponding to the info word $\boldsymbol{u}$ and the code word $\boldsymbol{x}$ are denoted by

$$\boldsymbol{z}_u = [z_{u,0}, z_{u,1}, \ldots, z_{u,K-1}], \qquad \boldsymbol{z}_x = [z_{x,0}, z_{x,1}, \ldots, z_{x,N-1}].$$

The soft-output decoder may compute general post-decoding values or extrinsic post-decoding values; the latter are of special importance in the context of iterative decoding. The *post-decoding values* for info symbol u_k and code symbol x_n are denoted by $v_{u,k}$ and $v_{x,n}$, respectively. The words of post-decoding values (post-decoding words) corresponding to the info word $\boldsymbol{u}$ and the code word $\boldsymbol{x}$ are denoted by

$$\boldsymbol{v}_u = [v_{u,0}, v_{u,1}, \ldots, v_{u,K-1}], \qquad \boldsymbol{v}_x = [v_{x,0}, v_{x,1}, \ldots, v_{x,N-1}].$$

In a similar way, *extrinsic post-decoding values* for info symbol u_k and code symbol x_n are denoted by $w_{u,k}$ and $w_{x,n}$, respectively. The words of extrinsic post-decoding values (extrinsic post-decoding words) corresponding to the info word $\boldsymbol{u}$ and the code word $\boldsymbol{x}$ are denoted by

$$\boldsymbol{w}_u = [w_{u,0}, w_{u,1}, \ldots, w_{u,K-1}], \qquad \boldsymbol{w}_x = [w_{x,0}, w_{x,1}, \ldots, w_{x,N-1}].$$

While a general post-decoding value may depend on all pre-decoding values, an extrinsic post-decoding value is required to fulfill the following condition:

Definition 3.2 (Extrinsic Value)
A post-decoding value (soft or hard) for an info or a code symbol is called extrinsic if it is independent from the corresponding pre-decoding value for this symbol. —

Thus, $w_{u,k}$ is independent from $z_{u,k}$, and $w_{x,n}$ is independent from $z_{x,n}$. For convenience, we call an extrinsic post-decoding value also simply an extrinsic value. When we want to emphasize that a post-decoding value is not extrinsic, we call it a complete post-decoding value or a nonextrinsic post-decoding value.

Soft-output decoding is a mapping from pre-decoding values for info and code symbols to post-decoding values for info or code symbols, called *soft-output decoding function*. An implementation of a soft-output decoding function, e.g., by means of a certain algorithm, is called a *soft-output decoder*. We define the following two decoding functions.

Definition 3.3 (Symbol-by-Symbol Soft-Output Decoding)
An *info-symbol soft-output decoding function* is a mapping

$$\begin{aligned} \mathbf{dec}_{\mathsf{info}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^K \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{v}_u \end{aligned}$$

from pre-decoding values to post-decoding values for info symbols. Correspondingly, a *code-symbol soft-output decoding function* is a mapping

$$\begin{aligned} \mathbf{dec}_{\mathsf{code}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^N \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{v}_x \end{aligned}$$

from pre-decoding values to post-decoding values for code symbols. —

For both decoding functions, the first argument is the word of pre-decoding values for info symbols, $\boldsymbol{z}_u$, and the second argument is the word of pre-decoding values for code symbols, $\boldsymbol{z}_x$.

In the context of iterative decoding, computation of extrinsic post-decoding values, called *extrinsic soft-output decoding*, is of special interest. Similarly to the general case, we define the following two decoding functions.

Definition 3.4 (Extrinsic Symbol-by-Symbol Soft-Output Decoding)
A soft-output decoding function is called an extrinsic soft-output decoding function if the function values are extrinsic post-decoding values. An *info-symbol extrinsic soft-output decoding function* is a mapping

$$\begin{aligned} \mathbf{dec}^{\mathsf{ext}}_{\mathsf{info}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^K \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{w}_u \end{aligned}$$

from pre-decoding values to extrinsic post-decoding values for info symbols. Correspondingly, a *code-symbol extrinsic soft-output decoding function* is a mapping

$$\begin{aligned} \mathbf{dec}^{\mathsf{ext}}_{\mathsf{code}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^N \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{w}_x \end{aligned}$$

from pre-decoding values to extrinsic post-decoding values for code symbols. —

Complete and extrinsic soft-output decoding are based on both the code $\mathcal{C}$ and the encoding, i.e., the mapping from info words to code words. Equivalently, we may say that soft-output decoding takes into account the code constraints given by the systematically extended code $\mathcal{C}_{\mathsf{syxt}}$. For emphasizing the underlying code constraints, we introduce the following *symbolic notation* for the above decoding functions:

$$\begin{aligned}
\boldsymbol{v}_u &= \mathbf{dec}_{\mathsf{info}}(\boldsymbol{z}_u, \boldsymbol{z}_x \,\|\, \mathcal{C}_{\mathsf{syxt}}),\\
\boldsymbol{v}_x &= \mathbf{dec}_{\mathsf{code}}(\boldsymbol{z}_u, \boldsymbol{z}_x \,\|\, \mathcal{C}_{\mathsf{syxt}}),\\
\boldsymbol{w}_u &= \mathbf{dec}^{\mathsf{ext}}_{\mathsf{info}}(\boldsymbol{z}_u, \boldsymbol{z}_x \,\|\, \mathcal{C}_{\mathsf{syxt}}),\\
\boldsymbol{w}_x &= \mathbf{dec}^{\mathsf{ext}}_{\mathsf{code}}(\boldsymbol{z}_u, \boldsymbol{z}_x \,\|\, \mathcal{C}_{\mathsf{syxt}})
\end{aligned}$$

The term "$\|\, \mathcal{C}_{\mathsf{syxt}}$" may be read "based on code $\mathcal{C}_{\mathsf{syxt}}$". If no pre-decoding values for info symbols are available, the original code is sufficient to determine post-decoding values for code symbols. Therefore, if $\boldsymbol{z}_u = \mathbf{0}$, we may write the code-symbol soft-output decoding functions shortly as

$$\begin{aligned}
\mathbf{dec}_{\mathsf{code}}(\boldsymbol{z}_x \,\|\, \mathcal{C}) &:= \mathbf{dec}_{\mathsf{code}}(\mathbf{0}, \boldsymbol{z}_x \,\|\, \mathcal{C}_{\mathsf{syxt}}),\\
\mathbf{dec}^{\mathsf{ext}}_{\mathsf{code}}(\boldsymbol{z}_x \,\|\, \mathcal{C}) &:= \mathbf{dec}^{\mathsf{ext}}_{\mathsf{code}}(\mathbf{0}, \boldsymbol{z}_x \,\|\, \mathcal{C}_{\mathsf{syxt}}).
\end{aligned}$$

As our focus is on soft outputs, we will omit the term "soft-output" when talking about decoding, decoding functions, and decoders, if possible without causing ambiguity.

In this thesis, we assume that the decoders have a symbol-wise symmetry.

Definition 3.5 (Symbol-wise Symmetric Soft-Output Decoder)
Assume a transmission system comprising a binary symmetric source, a binary linear channel encoder, binary-input symmetric memoryless channels, and a symbol-by-symbol soft-output decoder. Then, an info-symbol decoder is called symbol-wise symmetric if

$$p_{V_{u,k}|U_k}(v_u|u) = p_{V_{u,k}|U_k}(-v_u| - u)$$

for all $k \in \mathcal{K}$, $u \in \mathbb{B}$, $v_u \in \mathbb{R}$, and a code-symbol decoder is called symbol-wise symmetric if

$$p_{V_{x,n}|X_n}(v_x|x) = p_{V_{x,n}|X_n}(-v_x| - x)$$

for all $n \in \mathcal{N}$, $x \in \mathbb{B}$, $v_x \in \mathbb{R}$. Similarly, an info-symbol extrinsic decoder is called symbol-wise symmetric if

$$p_{W_{u,k}|U_k}(w_u|u) = p_{W_{u,k}|U_k}(-w_u| - u)$$

for all $k \in \mathcal{K}$, $u \in \mathbb{B}$, $w_u \in \mathbb{R}$, and an code-symbol extrinsic decoder is called symbol-wise symmetric if

$$p_{W_{x,n}|X_n}(w_x|x) = p_{W_{x,n}|X_n}(-w_x| - x)$$

for all $n \in \mathcal{N}$, $x \in \mathbb{B}$, $w_x \in \mathbb{R}$. —

When symbol-wise symmetric decoders are employed, the superchannel between an info symbol or a code symbol and the corresponding post-decoding value, $U_k \to V_{u,k}$ or

$X_n \to V_{x,n}$, is a binary-input symmetric memoryless channel. Similarly, when symbol-wise extrinsic decoders are employed, the superchannel between an info symbol or a code symbol and the corresponding extrinsic post-decoding value, $U_k \to W_{u,k}$ or $X_n \to W_{x,n}$, is a binary-input symmetric memoryless channel. Notice that each symbol position corresponds to one (separate) channel.

In some cases, it is useful to abstract from the symbol positions and consider simply the superchannel $U \to V_u$ (similarly for code symbols and extrinsic decoders). Even though this superchannel is no longer memoryless, it may be *interpreted* as memoryless if only symbol-wise statistical properties are of interest, like the average symbol error probability or the average symbol-wise mutual information. This is exploited in Chapter 4 and Chapter 5.

Though commonly used decoders can assumed to be symbol-wise symmetric, general conditions for this property are of interest. The following lemma deals with such conditions.

First, we introduce some notation. Consider an info-symbol soft-output decoder for a systematically extended code $\mathcal{C}_{\mathsf{syxt}}$ computing the post-decoding soft-value

$$v_{u,k} = \mathrm{dec}_{\mathsf{info},k}(\boldsymbol{z} \,\|\, \mathcal{C}_{\mathsf{syxt}})$$

for the info symbol U_k, where we use the abbreviation $\boldsymbol{z} := [\boldsymbol{z}_u, \boldsymbol{z}_x]$. The set of words $\boldsymbol{z}$ leading to $v_{u,k} = v$ is called the *soft-value decoding region* for $v_{u,k} = v$, and it is denoted by

$$\mathbb{D}(v_{u,k} = v) := \{\boldsymbol{z} \in \mathbb{R}^{K+N} : v_{u,k} = v\},$$

$k \in \mathcal{K}$, $v \in \mathbb{R}$. For code-symbol soft-output decoders, the definition is similar. Notice that the notation for soft-value decoding regions is similar to that for code cosets, introduced in (3.7).

Now consider the two cosets $\mathcal{C}_{\mathsf{syxt}}(u_k = +1)$ and $\mathcal{C}_{\mathsf{syxt}}(u_k = -1)$ of code $\mathcal{C}_{\mathsf{syxt}}$, defined in (3.7). Writing the element-wise multiplication of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ as

$$\boldsymbol{a} \otimes \boldsymbol{b} := [a_1 b_1, a_2 b_2, \ldots, a_n, b_n],$$

we have the following relation between the two cosets:

$$\mathcal{C}_{\mathsf{syxt}}(u_k = +1) = \mathcal{C}_{\mathsf{syxt}}(u_k = -1) \otimes \boldsymbol{x}'_{\mathsf{syxt}} \tag{3.14}$$

for all $\boldsymbol{x}'_{\mathsf{syxt}} \in \mathcal{C}_{\mathsf{syxt}}(u_k = -1)$ and all $k \in \mathcal{K}$. This holds in a similar way for cosets with respect to code symbols.

Using this notation, we can now state the condition for symbol-wise symmetry.

Lemma 3.1 (Condition for Symbol-Wise Symmetric Decoders)
An info-symbol soft-output decoder for a systematically extended code $\mathcal{C}_{\mathsf{syxt}}$ is symbol-wise symmetric if and only if the soft-value decoding regions fulfill the condition

$$\mathbb{D}(v_{u,k} = v) = \mathbb{D}(v_{u,k} = -v) \otimes \boldsymbol{x}'_{\mathsf{syxt}}$$

for all $\boldsymbol{x}'_{\mathsf{syxt}} \in \mathcal{C}_{\mathsf{syxt}}(u_k = -1)$, $v \in \mathbb{R}$, and $k \in \mathcal{K}$. This holds in a similar way for code-symbol soft-output decoders.

Proof. According to the definition of symbol-wise symmetry, we have to show that

$$p_{V_{u,k}|U_k}(v|+1) = p_{V_{u,k}|U_k}(-v|-1) \tag{3.15}$$

for all $v \in \mathbb{R}$ and $k \in \mathcal{K}$ if and only if the above condition on the soft-value decoding regions is fulfilled. To simplify the notation, we assume that the symbol alphabets of $z_{u,k}$ and z_{x_n} are discrete. For continuous symbol alphabets, the sums with respect to $z_{u,k}$ and z_{x_n} may simply be replaced by integrals in the following derivations. For convenience, we may use the abbreviation $\boldsymbol{z} := [\boldsymbol{z}_u, \boldsymbol{z}_x]$. We start with the symmetry of the channels, and then we exploit the linearity of the code.

By definition, the channels $U \to Y_u$ and $X \to Y_x$ are BISMCs and their outputs are LLRs. Therefore, we have

$$p_{Z_u|U}(z|u) = p_{Z_u|U}(-z|-u), \qquad p_{Z_x|X}(z|x) = p_{Z_x|X}(-z|-x)$$

for $z \in \mathbb{R}$ and $u, x \in \mathbb{B}$ by Lemma 2.2. Due to this symmetry, the conditional probability of $\boldsymbol{z}$ has the property

$$p_{\boldsymbol{Z}|\boldsymbol{X}_{\mathsf{syxt}}}(\boldsymbol{z}|\boldsymbol{x}_{\mathsf{syxt}}) = p_{\boldsymbol{Z}|\boldsymbol{X}_{\mathsf{syxt}}}(\boldsymbol{z} \otimes \boldsymbol{x}_{\mathsf{syxt}}|\mathbf{1}), \tag{3.16}$$

where $\mathbf{1}$ denotes the all-one vector.

The left-hand side of (3.15) can be expressed as

$$\begin{aligned} p_{V_{u,k}|U_k}(v|+1) &= 2^{-K+1} \cdot \sum_{\boldsymbol{z} \in \mathbb{D}(v_{u,k}=v)} \sum_{\boldsymbol{x}_{\mathsf{syxt}} \in \mathcal{C}_{\mathsf{syxt}}(u_k=+1)} p_{\boldsymbol{Z}|\boldsymbol{X}_{\mathsf{syxt}}}(\boldsymbol{z}|\boldsymbol{x}_{\mathsf{syxt}}) \\ &= 2^{-K+1} \cdot \sum_{\boldsymbol{z} \in \mathbb{D}(v_{u,k}=v)} \sum_{\boldsymbol{x}_{\mathsf{syxt}} \in \mathcal{C}_{\mathsf{syxt}}(u_k=+1)} p_{\boldsymbol{Z}|\boldsymbol{X}_{\mathsf{syxt}}}(\boldsymbol{z} \otimes \boldsymbol{x}_{\mathsf{syxt}}|\mathbf{1}), \end{aligned} \tag{3.17}$$

where we have applied (3.16) in the second line. Similarly, the right-hand side of (3.15) can be expressed as

$$p_{V_{u,k}|U_k}(-v|-1) = 2^{-K+1} \cdot \sum_{\boldsymbol{z} \in \mathbb{D}(v_{u,k}=-v)} \sum_{\boldsymbol{x}_{\mathsf{syxt}} \in \mathcal{C}_{\mathsf{syxt}}(u_k=-1)} p_{\boldsymbol{Z}|\boldsymbol{X}_{\mathsf{syxt}}}(\boldsymbol{z} \otimes \boldsymbol{x}_{\mathsf{syxt}}|\mathbf{1}). \tag{3.18}$$

Now, we apply variable substitutions in (3.18), using an arbitrary but fixed $\boldsymbol{x}'_{\mathsf{syxt}} \in \mathcal{C}_{\mathsf{syxt}}(u_k = -1)$. We substitute $\boldsymbol{z}$ by $\boldsymbol{z} \otimes \boldsymbol{x}'_{\mathsf{syxt}}$ and $\boldsymbol{x}_{\mathsf{syxt}}$ by $\boldsymbol{x}_{\mathsf{syxt}} \otimes \boldsymbol{x}'_{\mathsf{syxt}}$; this does not change the argument of the probability in (3.18), because

$$(\boldsymbol{z} \otimes \boldsymbol{x}'_{\mathsf{syxt}}) \otimes (\boldsymbol{x}_{\mathsf{syxt}} \otimes \boldsymbol{x}'_{\mathsf{syxt}}) = \boldsymbol{z} \otimes \boldsymbol{x}_{\mathsf{syxt}}.$$

In the sums, we replace $\mathbb{D}(v_{u,k} = -v)$ by $\mathbb{D}(v_{u,k} = v) \otimes \boldsymbol{x}'_{\mathsf{syxt}}$ and $\mathcal{C}_{\mathsf{syxt}}(u_k = -1)$ by $\mathcal{C}_{\mathsf{syxt}}(u_k = -1) \otimes \boldsymbol{x}'_{\mathsf{syxt}}$. Notice that

$$\mathcal{C}_{\mathsf{syxt}}(u_k = -1) \otimes \boldsymbol{x}'_{\mathsf{syxt}} = \mathcal{C}_{\mathsf{syxt}}(u_k = +1), \tag{3.19}$$

as shown in (3.14). Doing so, (3.18) can be written as

$$p_{V_{u,k}|U_k}(-v|-1) \;=\; 2^{-K+1} \cdot \sum_{\boldsymbol{z}\in\mathbb{D}(v_{u,k}=-v)\otimes\boldsymbol{x}'_{\mathsf{syxt}}} \sum_{\boldsymbol{x}_{\mathsf{syxt}}\in\mathcal{C}_{\mathsf{syxt}}(u_k=+1)} p_{\boldsymbol{Z}|\boldsymbol{X}_{\mathsf{syxt}}}(\boldsymbol{z}\otimes\boldsymbol{x}_{\mathsf{syxt}}|\mathbf{1}). \tag{3.20}$$

When comparing (3.17) and (3.20), it can easily be seen that (3.15) is fulfilled for all channels $U \to Y_u$ and $X \to Y_x$ if and only if

$$\mathbb{D}(v_{u,k}=v) = \mathbb{D}(v_{u,k}=-v)\otimes\boldsymbol{x}'_{\mathsf{syxt}}.$$

QED

The LogAPP decoder and the MaxLogAPP decoder, which are addressed in Section 3.4, can easily be seen to fulfill these conditions. Therefore, they are symbol-wise symmetric, as also observed in [RU01a,AK02]. Since a Viterbi decoder gives the same decoding results as a MaxLogAPP decoder with subsequent hard-decisions (cf. Section 3.4.2), it is also symbol-wise symmetric. We conjecture that also iterative decoders with symbol-wise constituent decoders are symbol-wise symmetric[11].

Useful and reasonable decoding requires that pre-decoding values can be correctly interpreted by the decoder. For example, the real-valued output of an AWGN channel and the binary output of a BSC have completely different meanings. Furthermore, possible statistical dependencies between pre-decoding values have to be taken into account. These observations motivate to use "standardized" pre-decoding values, so that a decoder can interpret them correctly without the need of having knowledge about the statistical properties of their "source". For that purpose, we introduce *standardized pre-decoding values* that fulfill the following two conditions:

(a) The pre-decoding values are LLRs:

$$z_{u,k} = L(U_k|z_{u,k}), \qquad z_{x,n} = L(X_n|z_{x,n}), \tag{3.21}$$

for $k = 0, 1, \ldots, K-1$ and $n = 0, 1, \ldots, N-1$.

(b) The pre-decoding values are conditionally independent:

$$p(\boldsymbol{z}_u, \boldsymbol{z}_x|\boldsymbol{u}, \boldsymbol{x}) = \prod_{k=0}^{K-1} p(z_{u,k}|u_k) \cdot \prod_{n=0}^{N-1} p(z_{x,n}|x_n) \tag{3.22}$$

for all $[\boldsymbol{u}\ \boldsymbol{x}] \in \mathcal{C}_{\mathsf{syxt}}$.

In the following, we refer to such values as *pre-decoding values that are conditionally independent LLRs*.

Due to this "standardization", a pre-decoding value that is equal to 0 means that no information about the corresponding symbol is available. Accordingly, if info symbols are assumed to be i.u.d. and only code symbols are transmitted, this can be taken into account by setting $\boldsymbol{z}_u = \mathbf{0}$. Condition 3.21 is illustrated by the following example:

[11] Reduced-state equalizers and decision-feedback equalizers are conjectured to be also symbol-wise symmetric.

Example 3.4
Consider a symbol X and a noisy observation y of this symbol. Let p denote the probability for $X = +1$ given y, and let z denote the conditional LLR for X given y:

$$\begin{aligned} p &:= \Pr(X = +1|Y = y), \\ z &:= L(X|Y = y) = \ln \frac{\Pr(X = +1|Y = y)}{\Pr(X = -1|Y = y)} = \ln \frac{p}{1-p}. \end{aligned}$$

The probabilities for X given p are obviously

$$\begin{aligned} \Pr(X = +1|P = p) &= p, \\ \Pr(X = -1|P = p) &= 1 - p. \end{aligned}$$

Therefore, the conditional LLR is given by

$$L(X|P = p) = \ln \frac{\Pr(X = +1|P = p)}{\Pr(X = -1|P = p)} = \ln \frac{p}{1-p} = z.$$

Finally, as there is a one-to-one relation between z and p, we have

$$L(X|Z = z) = L(X|P = p) = z.$$

In general, this holds if and only if z is an a-posteriori LLR. ◇

Assuming that the pre-decoding values are conditionally independent LLRs does not significantly restrict generality for two reasons. First, if a decoder can interpret soft-values correctly, then equivalently, the soft-values can be converted to LLRs before being passed to the decoder. Second, if a decoder can exploit statistical dependencies between soft-values, then equivalently, the soft-values can be converted to independent soft-values before being passed to the decoder[12]. The concept of these "standardized" pre-decoding values rather enables a clear structure for a decoder that is built up by several processing units, such as an iterative decoder built up by constituent decoders. Only the unit delivering soft-values has to know the statistical properties associated with these soft-values, but not the unit accepting these soft-values. Thus, each unit is "responsible" only for "its own" statistics.

Remark 3.1
Besides LLRs, other measures may equivalently be used as "standardized" pre-decoding values, e.g., the probability that a symbol is equal to +1. In this case, the superchannels between info or code symbols and the corresponding post-decoding values are still symmetric channels in the sense of Definition 2.4. However, the representation of the symmetry is less straight-forward.

Estimates for the info and the code symbols are obtained by applying *hard decisions* to the post-decoding soft-values[13]: If a post-decoding value is equal to zero, the symbol estimate is randomly chosen from $\mathbb{B}$; otherwise, the estimate is determined as

$$\begin{aligned} \hat{u}_k &:= \operatorname{sgn}(v_{u,k}), \\ \hat{x}_n &:= \operatorname{sgn}(v_{x,n}), \end{aligned}$$

[12]This is similar to employing a sample-whitened matched filter for the equalization of an intersymbol-interference channel, as proposed in [And73].

[13]Estimating symbols based on extrinsic values is possible, but in many cases not reasonable.

$k = 0, 1, \ldots, K-1$ and $n = 0, 1, \ldots, N-1$. The estimated info word and the estimated code word are denoted by

$$\begin{aligned} \hat{\boldsymbol{u}} &= [\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{K-1}], \\ \hat{\boldsymbol{x}} &= [\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_{N-1}], \end{aligned}$$

respectively.

The *probability of a word error* is

$$P_w := \Pr(\hat{\boldsymbol{U}} \neq \boldsymbol{U}) = \Pr(\hat{\boldsymbol{X}} \neq \boldsymbol{X}).$$

The error probability of info words and that of code words are identical as there is a one-to-one relation between info words and code words. The *probability of an info symbol error* is given by

$$\Pr(\hat{U} \neq U) = \frac{1}{K} \sum_{k \in \mathcal{K}} \Pr(\hat{U}_k \neq U_k).$$

The *probability of a code symbol error* is given by

$$\Pr(\hat{X} \neq X) = \frac{1}{N} \sum_{n \in \mathcal{N}} \Pr(\hat{X}_n \neq X_n).$$

In general, the error probability of info symbols and that of code symbols are different.

3.3 Information Transfer Functions

A soft-input soft-output decoder may be interpreted as a non-linear filter for soft-values [HH89a, HH89b, LYHH93, LHS00, LH01]. For characterizing the input-output behavior, the general decoding model from the previous section (Fig. 3.1) may be employed. The inputs may be the pre-decoding values for info or code symbols, and the outputs may be the complete or extrinsic post-decoding values for info or code symbols.

A complete description of the input-output behavior is given by the joint conditional probability distributions of the inputs and the outputs, given the corresponding info or code symbol. For simplification, both the input and the output distribution may be described by a single statistical parameter, sometimes denoted as noise parameter. Typically used parameters are the mean value, the variance, the ratio of squared mean and variance (corresponding to a signal-to-noise ratio) [HH89b, EGH01, CRU01, TtBH02], entropy or mutual information [Hoe95, tB99a, HHJF01, TtBH02]. (Notice that entropy and mutual information are equivalent, as the info symbols are assumed to be i.u.d. in the decoding model.)

Characterizing decoders via the average symbol-wise mutual information, averaged with respect to the symbol positions, was originally introduced for constituent decoders of iterative decoders [tBSY98, tB99b, tB99a]. The extrinsic information transfer (EXIT) characteristic of a constituent decoder describes the processing of mutual information associated with the inputs, called a-priori information, to mutual information associated with the extrinsic outputs, called extrinsic information. The chart depicting the EXIT

characteristics of all constituent decoders is called EXIT chart. In a similar way, the overall coding scheme can be characterized [HHJF01, HHFJ02, Hue04]. The information processing characteristic (IPC) of a coding scheme describes the processing of mutual information of the communication channel, called channel information, to mutual information between encoder input and (possibly soft) decoder output, called overall mutual information.

In general, a decoder may be characterized by the mapping of pre-decoding mutual information to post-decoding mutual information, called *information transfer function*. Such functions, agreeing with EXIT characteristics and IPCs, are defined in the sequel. We start with the characterizations of the input and the output distributions via the values of mutual information (cf. Fig. 3.1). For convenience, we may abbreviate "mutual information" by "information".

Definition 3.6
The values of info-symbol and the code-symbol pre-decoding information, also called a-priori information, are defined as

$$I_{\mathsf{apri},U} := \frac{1}{K}\sum_{k\in\mathcal{K}} I(U_k; Z_{u,k}), \qquad I_{\mathsf{apri},X} := \frac{1}{N}\sum_{n\in\mathcal{N}} I(X_n; Z_{x,n}).$$

The values of info-symbol and the code-symbol complete post-decoding information, for short complete information, are defined as

$$I_{\mathsf{cmp},U} := \frac{1}{K}\sum_{k\in\mathcal{K}} I(U_k; V_{u,k}), \qquad I_{\mathsf{cmp},X} := \frac{1}{N}\sum_{n\in\mathcal{N}} I(X_n; V_{x,n}).$$

The values of info-symbol and the code-symbol extrinsic post-decoding information, for short extrinsic information values, are defined as

$$I_{\mathsf{ext},U} := \frac{1}{K}\sum_{k\in\mathcal{K}} I(U_k; W_{u,k}), \qquad I_{\mathsf{ext},X} := \frac{1}{N}\sum_{n\in\mathcal{N}} I(X_n; W_{x,n}).$$

The word-wise complete information per info symbol, for short word-wise complete information, is defined as

$$I_{\mathsf{wcmp},U} := \frac{1}{K} I(\boldsymbol{U}; \boldsymbol{Z}_u, \boldsymbol{Z}_x)$$

—

Notice that $K = |\mathcal{K}|$ and $N = |\mathcal{N}|$ are the lengths of the info and the code word, respectively. The terms "a-priori information" and "extrinsic information" are commonly used in the context of EXIT functions, whereas "pre-decoding information" and "post-decoding information" emphasize the association to pre-decoding and post-decoding soft-values.

The symbol-wise mutual information values are values of average symbol-wise mutual information, and they are labeled with the corresponding info or code symbol. On the other hand, the word-wise complete information is the overall mutual information between

the info word and all observations, and it is labeled with "w" meaning "word-wise". Notice that this is the maximal value of mutual information between encoder inputs and channel outputs.

Using these values of mutual information, extrinsic soft-output decoders may be characterized by the following functions:

Definition 3.7 (EXIT Functions)
Assume the decoding model from Section 3.2 with arbitrary but fixed models for the info-symbol and the code-symbol channel. The function

$$\begin{aligned} \mathrm{itf}_{\mathsf{ext},U} : [0,1] &\rightarrow [0,1] \\ I_{\mathsf{apri},U} &\mapsto I_{\mathsf{ext},U} \end{aligned}$$

with parameter $I_{\mathsf{apri},X}$ is called info-symbol extrinsic information transfer (EXIT) function. The function

$$\begin{aligned} \mathrm{itf}_{\mathsf{ext},X} : [0,1] &\rightarrow [0,1] \\ I_{\mathsf{apri},X} &\mapsto I_{\mathsf{ext},X} \end{aligned}$$

with parameter $I_{\mathsf{apri},U}$ is called code-symbol extrinsic information transfer (EXIT) function. —

The term "EXIT function" was introduced in [AKtB02, AKtB03] and is a synonym for "EXIT characteristic". However, it makes clear that a function is addressed. EXIT functions describe a property of the underlying extrinsic soft-output decoding functions, and they depend on the applied models of the info-symbol and the code-symbol channel.

Example 3.5
Consider a systematic single parity check code with info word length $K = 3$ and BECs as channel models. The systematic symbols are regarded as info symbols and only the parity symbol is regarded as code symbol; thus we have the code word length $N = 1$. According to the decoding model from Section 3.2, the info symbols u_0, u_1, u_2 are transmitted over the info-symbol channel $U \rightarrow Z_u$, yielding the channel LLRs $z_{u,0}, z_{u,1}, z_{u,2}$; the code symbol x_0 (identical to the parity symbol) is transmitted over the code-symbol channel $X \rightarrow Z_x$, yielding the channel LLR $z_{x,0}$.

Extrinsic soft-values for the info symbols are computed using the info-symbol extrinsic soft-output decoding functions (cf. Definition 3.4)

$$\begin{aligned} w_{u,0} &:= L(U_0|z_{u,1}, z_{u,2}, z_{x,0}), \\ w_{u,1} &:= L(U_1|z_{u,0}, z_{u,2}, z_{x,0}), \\ w_{u,2} &:= L(U_2|z_{u,0}, z_{u,1}, z_{x,0}). \end{aligned}$$

(This corresponds to LogAPP decoding, which is explained in Section 3.4.1.)

The info-symbol EXIT function can easily be computed by inspecting the erasure probabilities (cf. Section 6.3), and it can analytically be expressed as

$$I_{\mathsf{ext},U} = \mathrm{itf}_{\mathsf{ext},U}(I_{\mathsf{apri},U}; I_{\mathsf{apri},X}) = (I_{\mathsf{apri},U})^{K-1} \cdot I_{\mathsf{apri},X},$$

where $K - 1 = N - 2 = 2$. This function is depicted in Fig. 3.2. ◇

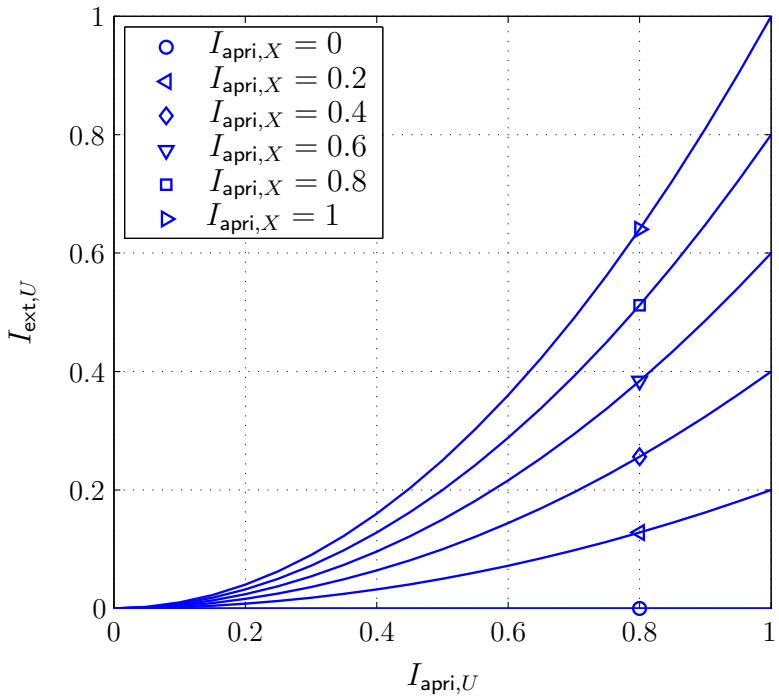


Figure 3.2: Info-symbol EXIT functions for the single parity check code from Example 3.5; the info word length is $K = 3$. The info-symbol channel and the code-symbol channel are BECs. (Mutual information is given in bit/use.)

Similarly to extrinsic soft-output decoders, general soft-output decoders may be characterized. We assume that only code symbols are transmitted and that the info-symbol channel does not exist[14].

Definition 3.8 (CIT Functions)
Assume the decoding model from Section 3.2 with arbitrary but fixed model for the code-symbol channel and no a-priori information on info symbols, i.e., $I_{\mathsf{apri},U} = 0$. The function

$$\begin{aligned} \mathrm{itf}_{\mathsf{cmp}} : [0,1] &\rightarrow [0,1] \\ I_{\mathsf{apri},X} &\mapsto I_{\mathsf{cmp},U} \end{aligned}$$

is called the complete information transfer (CIT) function. The function

$$\begin{aligned} \mathrm{itf}_{\mathsf{wcmp}} : [0,1] &\rightarrow [0,1] \\ I_{\mathsf{apri},X} &\mapsto I_{\mathsf{wcmp},U} \end{aligned}$$

is called the word-wise complete information transfer (CIT) function. —

The term "CIT function" is a generalization of "EXIT function", and it is a synonym for "information processing characteristic". However, it makes clear that actually a function is addressed. Theoretical results and applications can be found in [Hue04]. Similar

[14] A generalization is possible and straight-forward.

to EXIT functions, CIT functions describe a property of the underlying coding scheme, and they depend on the applied models for the code-symbol channel.

An upper bound for the CIT function is derived in [HHFJ02, Hue04]: For a coding scheme of rate R and a communication channel with mutual information I_{ch}, the complete information per info symbol is upper-bounded as

$$I_{W,\mathsf{cmp}} \leq \min\Big\{\frac{1}{R} I_{\mathsf{ch}}, 1\Big\}.$$

A coding scheme is called an *ideal coding scheme* if it attains this bound and leads to the minimal error probability. The ideal coding scheme may be used as a reference to assess the performance of real coding schemes (cf. Section 6.3).

Example 3.6
Consider a systematically encoded single parity check code of length N and BECs as channel models. The info symbols are identical to the systematic symbols, and thus we have info word length $K = N - 1$. According to the decoding model from Section 3.2, the info symbols $u_0, u_1, \ldots, u_{K-1}$ are transmitted over the info-symbol channel $U \to Z_u$, yielding the channel LLRs $z_{u,0}, z_{u,1}, \ldots, z_{u,K-1}$; the code symbols $x_0, x_1, \ldots, x_{N-1}$ are transmitted over the code-symbol channel $X \to Z_x$, yielding the channel LLRs $z_{x,0}, z_{x,1}, \ldots, z_{x,N-1}$.

For CIT functions according to Definition 3.8, no info-symbol channel is assumed and therefore we set $z_{u,k} = 0$ for all $k = 0, 1, \ldots, K-1$.

Complete post-decoding soft-values for the info symbols are computed using the info-symbol soft-output decoding functions (cf. Definition 3.3)

$$v_{u,k} := L(U_k | z_{x,0}, z_{x,1}, \ldots, z_{x,N-1}),$$

$k = 0, 1, \ldots, K-1$. (This corresponds to LogAPP decoding, which is explained in Section 3.4.1.) The info-symbol channel LLRs are omitted, because they are equal to zero and thus do not effect the decoding result.

The CIT function can easily be computed by inspecting the erasure probabilities (cf. Section 6.3), and it can analytically be expressed as

$$I_{\mathsf{cmp},U} = \mathrm{itf}_{\mathsf{cmp}}(I_{\mathsf{apri},X}) = 1 - (1 - I_{\mathsf{apri},X}) \cdot (1 - (I_{\mathsf{apri},X})^{N-1}).$$

This function is depicted in Fig. 3.3. ◇

Information transfer functions generally *depend* on the applied models for the info-symbol and the code-symbol channel, though they abstract from the channel models as they plot only values of mutual information. Bounds on information transfer functions that are valid for *all* BISMCs and that depend *only* on the mutual information of the info-symbol and the code-symbol channel are derived in Chapter 6.

3.4 Soft-Output Decoding Principles

Two basic principles for symbol-by-symbol soft-output decoding of linear codes are LogAPP decoding and MaxLogAPP decoding. Both decoding principles are optimum in

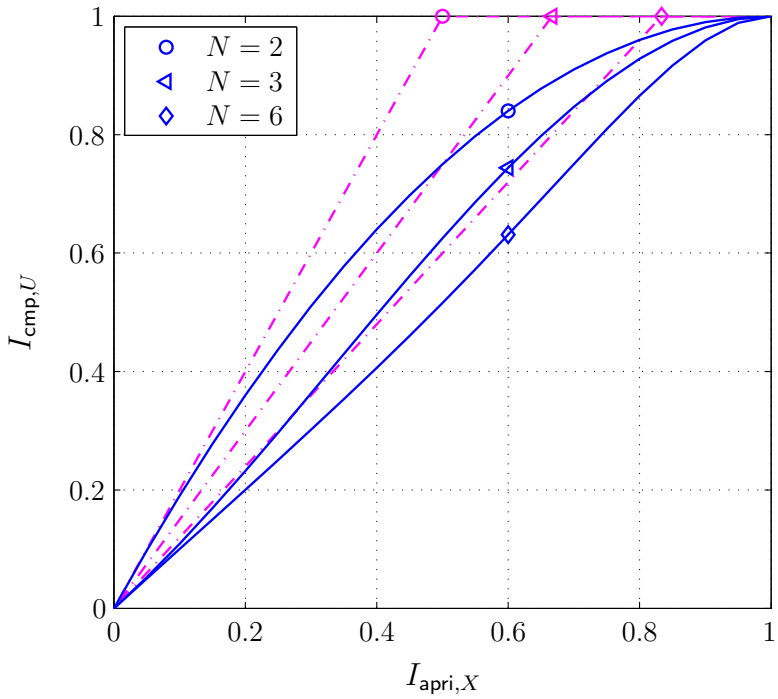


Figure 3.3: Complete information transfer (CIT) functions for the single parity check codes from Example 3.6 (solid lines) and CIT functions for ideal coding schemes (dash-dotted lines); several code word lengths N. The code-symbol channel is a BEC. (Mutual information is given in bit/use.)

some sense: LogAPP decoding leads to minimum bit error probability, and MaxLogAPP decoding leads to minimum word error probability. On the other hand, MaxLogAPP decoding may also be interpreted as an approximation of LogAPP decoding. Before defining the two decoding principles, some practical aspects are outlined.

For implementing LogAPP and MaxLogAPP decoding, various algorithms are proposed in literature. The computations are commonly performed using either probabilities or logarithmic probabilities. Calculations in the probability domain and calculations in the logarithmic probability domain (also called logarithmic domain or simply log-domain) are equivalent, only the applied operations differ [Hub02]; for practical implementations, the logarithmic domain shows numerical advantages [RHV97]. Typically, each algorithm shows a trade-off between required memory size and computational complexity. The following three algorithms represent the "main" types. Each may be used to implement LogAPP or MaxLogAPP decoding by choosing the appropriate operators. Two algorithms are trellis-based, and one algorithm is graph-based.

1. The first trellis-based algorithm is the *two-way algorithm*, also known as forward-backward algorithm [BCJR74, RHV97, JZ99, PM03]. The post-decoding values are computed by means of a forward recursion, a backward recursion, and a final combination. Typically, the memory requirement is relatively high. In the case of high-rate codes, decoding on the trellis of the dual code may be advanta-

geous [BD76, HR76, BDG79, HOP96].

2. The second trellis-based algorithm is the *one-way algorithm* [Bat87, HH89b, LVS95, JZ99], also called soft-output Viterbi algorithm (SOVA) [HH89b]. The computations are similar to that of a Viterbi algorithm [Vit67, For73], where after each step, update operations for preliminary post-decoding values are performed. Typically, the memory requirement is relatively low.

3. The third algorithm is the *message passing algorithm*, which operates on a cycle-free graph of the code (or of the systematically extended code). Variations are the belief propagation algorithm, the sum-product algorithm and the min-sum algorithm [WLK95, KFL01, Loe04]. Typically, the memory requirement is in between that of the two-way and the one-way algorithm.

The equivalence of a two-way and a one-way algorithm for MaxLogAPP decoding is explicitly proved in [FBLH98].

In the sequel, we focus on the underlying decoding principles; for implementational aspects, we refer the reader to the literature. Furthermore, we restrict ourselves to binary codes.

To describe LogAPP and MaxLogAPP decoding, we employ the following notation. Let $\boldsymbol{a}$ denote a vector with index set $\mathcal{I} := \{0, 1, \ldots, L-1\}$. Then $\boldsymbol{a}_{\setminus i}$ denotes this vector without element a_i, and $\boldsymbol{a}_{0 \to i}$ denotes[15] this vector with element a_i replaced by 0:

$$\begin{aligned} \boldsymbol{a} &= [a_0, \ldots, a_{L-1}], \\ \boldsymbol{a}_{\setminus i} &= [a_0, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{L-1}], \\ \boldsymbol{a}_{0 \to i} &= [a_0, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_{L-1}] \end{aligned}$$

(cf. Appendix B). Furthermore, let

$$\max_{i \in \mathcal{I}}{}^{\star}\, a_i := \ln \sum_{i \in \mathcal{I}} \exp(a_i) \tag{3.23}$$

for $a_i \in \mathbb{R}$, $i \in \mathcal{I}$ (cf. Appendix C).

3.4.1 LogAPP Decoding

A LogAPP decoder[16] computes a-posteriori LLRs for info and/or code symbols, provided that the pre-decoding values for info and code symbols are conditionally independent LLRs (cf. (3.21) and (3.22)). As both the inputs and the outputs are LLRs, a LogAPP decoder may be regarded as a filter for LLRs [LH00b, LHS00]

[15] The notation $\boldsymbol{a}_{0 \to i}$ indicates symbolically that the element at position i is replaced by a zero, i.e., "$0 \to a_i$".

[16] The abbreviation "LogAPP" stands for "logarithmic a-posteriori probability". Some authors use "LogMAP" and "LogAPP" equivalently. As the "M" in LogMAP stands for "maximum", this nomenclature makes sense only if the hard decision is part of the decoder, which often is not the case. As we are interested in soft-output decoding, we use the term LogAPP. (cf. Footnote 18.)

A-posteriori LLRs for info symbols and a-posteriori LLRs for code symbols are defined as

$$L(U_k|\boldsymbol{z}_u,\boldsymbol{z}_x) := \ln\frac{\Pr(U_k=+1|\boldsymbol{Z}_u=\boldsymbol{z}_u,\boldsymbol{Z}_x=\boldsymbol{z}_x)}{\Pr(U_k=-1|\boldsymbol{Z}_u=\boldsymbol{z}_u,\boldsymbol{Z}_x=\boldsymbol{z}_x)}, \tag{3.24}$$

$$L(X_n|\boldsymbol{z}_u,\boldsymbol{z}_x) := \ln\frac{\Pr(X_n=+1|\boldsymbol{Z}_u=\boldsymbol{z}_u,\boldsymbol{Z}_x=\boldsymbol{z}_x)}{\Pr(X_n=-1|\boldsymbol{Z}_u=\boldsymbol{z}_u,\boldsymbol{Z}_x=\boldsymbol{z}_x)}, \tag{3.25}$$

respectively. *Extrinsic LLRs* for info symbols and extrinsic LLRs for code symbols are defined as

$$L(U_k|\boldsymbol{z}_{u,\backslash k},\boldsymbol{z}_x) := \ln\frac{\Pr(U_k=+1|\boldsymbol{Z}_{u,\backslash k}=\boldsymbol{z}_{u,\backslash k},\boldsymbol{Z}_x=\boldsymbol{z}_x)}{\Pr(U_k=-1|\boldsymbol{Z}_{u,\backslash k}=\boldsymbol{z}_{u,\backslash k},\boldsymbol{Z}_x=\boldsymbol{z}_x)}, \tag{3.26}$$

$$L(X_n|\boldsymbol{z}_u,\boldsymbol{z}_{x,\backslash n}) := \ln\frac{\Pr(X_n=+1|\boldsymbol{Z}_u=\boldsymbol{z}_u,\boldsymbol{Z}_{x,\backslash n}=\boldsymbol{z}_{x,\backslash n})}{\Pr(X_n=-1|\boldsymbol{Z}_u=\boldsymbol{z}_u,\boldsymbol{Z}_{x,\backslash n}=\boldsymbol{z}_{x,\backslash n})}, \tag{3.27}$$

respectively. As can be seen above, the extrinsic LLR for a symbol is equal to the a-posteriori LLR for this symbol excluding the channel LLR for this symbol in the condition. Hence, an extrinsic LLR represents a special a-posteriori LLR. As an LLR with value 0 means "no information", we have the equalities

$$\begin{aligned} L(U_k|\boldsymbol{z}_{u,\backslash k},\boldsymbol{z}_x) &= L(U_k|\boldsymbol{z}_{u,0\to k},\boldsymbol{z}_x),\\ L(X_n|\boldsymbol{z}_u,\boldsymbol{z}_{x,\backslash n}) &= L(X_n|\boldsymbol{z}_u,\boldsymbol{z}_{x,0\to n}). \end{aligned}$$

This relation is used for defining extrinsic LogAPP decoding using LogAPP decoding.

The following decoding functions compute a-posteriori LLRs and extrinsic LLRs when the pre-decoding values are conditionally independent LLRs (cf. (3.21) and (3.22)). Notice that LogAPP decoding complies with soft-output decoding according to Definition 3.3, and extrinsic LogAPP decoding complies with extrinsic soft-output decoding according to Definition 3.4.

Definition 3.9 (LogAPP Decoding)
Let $\mathcal{C}_{\mathsf{syxt}}$ denote the systematically extended code of a binary linear code $\mathcal{C}$. An info-symbol LogAPP decoder and a code-symbol LogAPP decoder are mappings

$$\begin{aligned} \mathbf{logapp}_{\mathsf{info}} : \mathbb{R}^K \times \mathbb{R}^N &\to \mathbb{R}^K\\ [\boldsymbol{z}_u,\boldsymbol{z}_x] &\mapsto \boldsymbol{v}_u,\\ \mathbf{logapp}_{\mathsf{code}} : \mathbb{R}^K \times \mathbb{R}^N &\to \mathbb{R}^N\\ [\boldsymbol{z}_u,\boldsymbol{z}_x] &\mapsto \boldsymbol{v}_x, \end{aligned}$$

respectively, realizing the LogAPP decoding functions[17]

$$\mathrm{logapp}_{\mathsf{info},k}(\boldsymbol{z}_u,\boldsymbol{z}_x) := \max^{\star}_{\substack{[\boldsymbol{u}\,\boldsymbol{x}]\in\\ \mathcal{C}_{\mathsf{syxt}}(u_k=+1)}}\left(\tfrac{1}{2}\boldsymbol{z}_u\boldsymbol{u}^{\mathsf{T}}+\tfrac{1}{2}\boldsymbol{z}_x\boldsymbol{x}^{\mathsf{T}}\right) - \max^{\star}_{\substack{[\boldsymbol{u}\,\boldsymbol{x}]\in\\ \mathcal{C}_{\mathsf{syxt}}(u_k=-1)}}\left(\tfrac{1}{2}\boldsymbol{z}_u\boldsymbol{u}^{\mathsf{T}}+\tfrac{1}{2}\boldsymbol{z}_x\boldsymbol{x}^{\mathsf{T}}\right),$$

$$\mathrm{logapp}_{\mathsf{code},n}(\boldsymbol{z}_u,\boldsymbol{z}_x) := \max^{\star}_{\substack{[\boldsymbol{u}\,\boldsymbol{x}]\in\\ \mathcal{C}_{\mathsf{syxt}}(x_n=+1)}}\left(\tfrac{1}{2}\boldsymbol{z}_u\boldsymbol{u}^{\mathsf{T}}+\tfrac{1}{2}\boldsymbol{z}_x\boldsymbol{x}^{\mathsf{T}}\right) - \max^{\star}_{\substack{[\boldsymbol{u}\,\boldsymbol{x}]\in\\ \mathcal{C}_{\mathsf{syxt}}(x_n=-1)}}\left(\tfrac{1}{2}\boldsymbol{z}_u\boldsymbol{u}^{\mathsf{T}}+\tfrac{1}{2}\boldsymbol{z}_x\boldsymbol{x}^{\mathsf{T}}\right),$$

[17] The description via vector products follows [Sor02]. The notation for subsets of codes, $\mathcal{C}_{\mathsf{syxt}}(u_k=\pm1)$, is defined in Section 3.1.

$k = 0, 1, \ldots, K-1$, $n = 0, 1, \ldots, N-1$. An extrinsic info-symbol LogAPP decoder and an extrinsic code-symbol LogAPP decoder are mappings

$$\begin{aligned} \mathbf{logapp}^{\mathsf{ext}}_{\mathsf{info}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^K \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{w}_u, \\ \mathbf{logapp}^{\mathsf{ext}}_{\mathsf{code}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^N \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{w}_x, \end{aligned}$$

respectively, realizing the extrinsic LogAPP decoding functions

$$\begin{aligned} \mathrm{logapp}^{\mathsf{ext}}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) &:= \mathrm{logapp}_{\mathsf{info},k}(\boldsymbol{z}_{u,\theta \to k}, \boldsymbol{z}_x), \\ \mathrm{logapp}^{\mathsf{ext}}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) &:= \mathrm{logapp}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_{x,\theta \to n}), \end{aligned}$$

$k = 0, 1, \ldots, K-1$, $n = 0, 1, \ldots, N-1$. —

As required by extrinsic decoding functions (cf. Definition 3.4), $w_{u,k}$ does not depend on $z_{u,k}$, and $w_{x,n}$ does not depend on $z_{x,n}$.

An important fact is worth pointing out: LogAPP decoding yields LLRs, i.e.,

$$\begin{aligned} v_{u,k} = \mathrm{logapp}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) &= L(U_k | \boldsymbol{z}_u, \boldsymbol{z}_x), \\ v_{x,n} = \mathrm{logapp}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) &= L(X_n | \boldsymbol{z}_u, \boldsymbol{z}_x) \end{aligned}$$

and

$$\begin{aligned} w_{u,k} = \mathrm{logapp}^{\mathsf{ext}}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) &= L(U_k | \boldsymbol{z}_{u,\setminus k}, \boldsymbol{z}_x), \\ w_{x,n} = \mathrm{logapp}^{\mathsf{ext}}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) &= L(X_n | \boldsymbol{z}_u, \boldsymbol{z}_{x,\setminus n}), \end{aligned}$$

only if the word of pre-decoding values, $[\boldsymbol{z}_u, \boldsymbol{z}_x]$, consists of conditionally independent LLRs (cf. (3.21) and (3.22)). Otherwise, the pre-decoding values are not correctly interpreted by the decoder, and the decoding results cannot be guaranteed to be LLRs. This is relevant especially in the context of iterative decoding, where the pre-decoding values are only "approximately" conditionally independent LLRs (cf. Section 4.5).

One advantage of using LLRs is a simple relation between a-posteriori LLRs, $v_{u,k}$ and $v_{x,n}$, extrinsic LLRs, $w_{u,k}$ and $w_{x,n}$, and channel LLRs (which may also be interpreted as a-priori LLRs, as discussed above), $z_{u,k}$ and $z_{x,n}$:

$$\begin{aligned} v_{u,k} &= w_{u,k} + z_{u,k}, \\ v_{x,n} &= w_{x,n} + z_{x,n}, \end{aligned} \tag{3.28}$$

$k = 0, 1, \ldots, K-1$, $n = 0, 1, \ldots, N-1$, [HOP96, RHV97].

Example 3.7
Consider the repetition code of length $N = 3$ ($K = 1$) from Example 3.1. Let the channel LLRs by given as $z_{u,0} = 0$ (no information about info symbols available at the decoder) and $z_{x,0}, z_{x,1}, z_{x,2} \in \mathbb{R}$. For the systematically extended code $\mathcal{R}_{3,\mathsf{syxt}}$, the two subsets with respect to the info symbol U_0 are given by

$$\begin{aligned} \mathcal{R}_{3,\mathsf{syxt}}(u_0 = +1) &= \{[+1, +1, +1, +1]\}, \\ \mathcal{R}_{3,\mathsf{syxt}}(u_0 = -1) &= \{[-1, -1, -1, -1]\}, \end{aligned}$$

each containing only one code word. Thus, the LogAPP decoding function for info symbol U_0 is given by

$$\begin{aligned} v_{u,0} &= \mathrm{logapp}_{\mathsf{info},0}(0, \boldsymbol{z}_x) \\ &= \tfrac{1}{2}(+z_{x,0} + z_{x,1} + z_{x,2}) - \tfrac{1}{2}(-z_{x,0} - z_{x,1} - z_{x,2}) \\ &= z_{x,0} + z_{x,1} + z_{x,2}. \end{aligned}$$

$\diamond$

Example 3.8
Consider the (nonsystematically encoded) single parity check code of length $N = 3$ ($K = 2$) from Example 3.2. Let the channel LLRs by given as $\boldsymbol{z}_u = [z_{u,0}, z_{u,1}] = \mathbf{0}$ (no information about info symbols) and $z_{x,0}, z_{x,1}, z_{x,2} \in \mathbb{R}$. For the systematically extended code $\mathcal{S}_{3,\mathsf{syxt}}$, the two subsets with respect to info symbol U_1 are given by

$$\begin{aligned} \mathcal{S}_{3,\mathsf{syxt}}(u_1 = +1) &= \{[+1,+1,+1,+1,+1], [-1,+1,-1,-1,+1]\}, \\ \mathcal{S}_{3,\mathsf{syxt}}(u_1 = -1) &= \{[+1,-1,+1,-1,-1], [-1,-1,-1,+1,-1]\}, \end{aligned}$$

each containing two code words. Thus, the LogAPP decoding function for info symbol U_1 is given by

$$\begin{aligned} v_{u,1} &= \mathrm{logapp}_{\mathsf{info},1}(\mathbf{0}, \boldsymbol{z}_x) \\ &= \max^{\star}\Big\{\tfrac{1}{2}(+z_{x,0} + z_{x,1} + z_{x,2}), \tfrac{1}{2}(-z_{x,0} - z_{x,1} + z_{x,2})\Big\} \\ &\quad - \max^{\star}\Big\{\tfrac{1}{2}(+z_{x,0} - z_{x,1} - z_{x,2}), \tfrac{1}{2}(-z_{x,0} + z_{x,1} - z_{x,2})\Big\}. \end{aligned}$$

$\diamond$

Example 3.9
Consider again the single parity check code of length $N = 3$ ($K = 2$) from Example 3.2. Let the channel LLRs (again) be given as $\boldsymbol{z}_u = [z_{u,0}, z_{u,1}] = \mathbf{0}$ (no information about info symbols) and $z_{x,0}, z_{x,1}, z_{x,2} \in \mathbb{R}$. When we want to decode to code symbols and have no information about info symbols, it is sufficient to consider the (original) code

$$\mathcal{S}_3 = \{[+1,+1,+1], [-1,+1,-1], [+1,-1,-1], [-1,-1,+1]\}.$$

The two subsets with respect to code symbol X_0 are

$$\begin{aligned} \mathcal{S}_3(x_0 = +1) &= \{[+1,+1,+1], [+1,-1,-1]\}, \\ \mathcal{S}_3(x_0 = -1) &= \{[-1,+1,-1], [-1,-1,+1]\}. \end{aligned}$$

The extrinsic LogAPP decoding function for code symbol X_0 is given by

$$\begin{aligned} v_{x,0} &= \mathrm{logapp}^{\mathsf{ext}}_{\mathsf{code},0}(\mathbf{0}, \boldsymbol{z}_x) = \mathrm{logapp}_{\mathsf{code},0}(\mathbf{0}, \boldsymbol{z}_{x,\not{0}\to 0}) \\ &= \max^{\star}\Big\{\tfrac{1}{2}(+z_{x,1} + z_{x,2}), \tfrac{1}{2}(-z_{x,1} - z_{x,2})\Big\} \\ &\quad - \max^{\star}\Big\{\tfrac{1}{2}(+z_{x,1} - z_{x,2}), \tfrac{1}{2}(-z_{x,1} + z_{x,2})\Big\}. \end{aligned}$$

For computing an extrinsic LLR when all code symbols are coupled by only one parity-check equation, the operator $\boxplus$ was introduced in [HOP96] (cf. Appendix C). Using this operator, we may equivalently write

$$\begin{aligned} v_{x,0} &= \mathrm{logapp}^{\mathsf{ext}}_{\mathsf{code},0}(\mathbf{0}, \boldsymbol{z}_x) \\ &= z_{x,1} \boxplus z_{x,2} := 2\tanh^{-1}\Big(\tanh\big(\tfrac{z_{x,1}}{2}\big) \cdot \tanh\big(\tfrac{z_{x,2}}{2}\big)\Big). \end{aligned}$$

$\diamond$

A-posteriori LLRs are sufficient statistics, i.e., they "carry" all available information about the corresponding symbols. Hence, LogAPP decoding is *optimal with respect to* two aspects:

(a) hard decisions of the post-decoding values minimize the symbol error probability;

(b) the mutual information between a symbol and its corresponding post-decoding value is maximal.

Furthermore, as the soft-outputs are LLRs, they may be used as "useful" pre-decoding values by a subsequent processing stage, when neglecting the statistical dependencies (these may be reduced by interleaving); i.e., they can be correctly interpreted. This property of being interpretable does not immediately follow from (a) and (b). For example, when multiplying the post-decoding values by 100, the resulting values are still sufficient statistics, and (a) and (b) are still fulfilled, but they are definitely not LLRs. The two properties of being optimal and interpretable are utilized in Chapter 5 for estimating transmission quality parameters.

3.4.2 MaxLogAPP Decoding

The LogAPP decoding functions according to Definition 3.9 contain the $\max^\star$ function, defined in (3.23). Replacing the $\max^\star$ functions by a simple maximum function leads to MaxLogAPP decoding[18]. As taking the maximum represents both an upper bound and a close approximation of the $\max^\star$ function (cf. Appendix C),

$$\max_{i\in\mathcal{I}}{}^{\star}\, a_i \quad \gtrapprox \quad \max_{i\in\mathcal{I}} a_i, \tag{3.29}$$

MaxLogAPP decoding may be interpreted as an approximation of LogAPP decoding. On the other hand, MaxLogAPP decoding is optimal with respect to word-wise decoding, and it may thus also be seen independently from LogAPP decoding. This less known interpretation is discussed at the end of this section.

Similar to the case of LogAPP decoding, MaxLogAPP decoding complies with soft-output decoding according to Definition 3.3, and extrinsic MaxLogAPP decoding complies with extrinsic soft-output decoding according to Definition 3.4.

[18] The "Max" in "MaxLogAPP" refers to the maximum function replacing the $\max^\star$ function. Some authors use MaxLogAPP and MaxLogMAP equivalently. However, similarly to the use of "LogMAP" and "LogAPP" (cf. Footnote 16), MaxLogMAP should only be used if hard decisions are included.

Definition 3.10 (MaxLogAPP Decoder)
Let $\mathcal{C}_{\mathsf{syxt}}$ denote the systematically extended code of a binary linear code $\mathcal{C}$. An info-symbol MaxLogAPP decoder and a code-symbol MaxLogAPP decoder are mappings

$$\begin{aligned}\mathbf{maxlogapp}_{\mathsf{info}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^K \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{v}_u,\end{aligned}$$

$$\begin{aligned}\mathbf{maxlogapp}_{\mathsf{code}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^N \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{v}_x,\end{aligned}$$

respectively, realizing the MaxLogAPP decoding functions

$$\mathrm{maxlogapp}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) := \max_{\substack{[\boldsymbol{u}\,\boldsymbol{x}]\in \\ \mathcal{C}_{\mathsf{syxt}}(u_k=+1)}} \left(\tfrac{1}{2}\boldsymbol{z}_u\boldsymbol{u}^{\mathsf{T}} + \tfrac{1}{2}\boldsymbol{z}_x\boldsymbol{x}^{\mathsf{T}}\right) - \max_{\substack{[\boldsymbol{u}\,\boldsymbol{x}]\in \\ \mathcal{C}_{\mathsf{syxt}}(u_k=-1)}} \left(\tfrac{1}{2}\boldsymbol{z}_u\boldsymbol{u}^{\mathsf{T}} + \tfrac{1}{2}\boldsymbol{z}_x\boldsymbol{x}^{\mathsf{T}}\right),$$

$$\mathrm{maxlogapp}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) := \max_{\substack{[\boldsymbol{u}\,\boldsymbol{x}]\in \\ \mathcal{C}_{\mathsf{syxt}}(x_n=+1)}} \left(\tfrac{1}{2}\boldsymbol{z}_u\boldsymbol{u}^{\mathsf{T}} + \tfrac{1}{2}\boldsymbol{z}_x\boldsymbol{x}^{\mathsf{T}}\right) - \max_{\substack{[\boldsymbol{u}\,\boldsymbol{x}]\in \\ \mathcal{C}_{\mathsf{syxt}}(x_n=-1)}} \left(\tfrac{1}{2}\boldsymbol{z}_u\boldsymbol{u}^{\mathsf{T}} + \tfrac{1}{2}\boldsymbol{z}_x\boldsymbol{x}^{\mathsf{T}}\right),$$

$k = 0, 1, \ldots, K-1$, $n = 0, 1, \ldots, N-1$. An extrinsic info-symbol MaxLogAPP decoder and an extrinsic code-symbol MaxLogAPP decoder are mappings

$$\begin{aligned}\mathbf{maxlogapp}^{\mathsf{ext}}_{\mathsf{info}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^K \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{w}_u,\end{aligned}$$

$$\begin{aligned}\mathbf{maxlogapp}^{\mathsf{ext}}_{\mathsf{code}} : \mathbb{R}^K \times \mathbb{R}^N &\rightarrow \mathbb{R}^N \\ [\boldsymbol{z}_u, \boldsymbol{z}_x] &\mapsto \boldsymbol{w}_x,\end{aligned}$$

respectively, realizing the extrinsic MaxLogAPP decoding functions

$$\begin{aligned}\mathrm{maxlogapp}^{\mathsf{ext}}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) &:= \mathrm{maxlogapp}_{\mathsf{info},k}(\boldsymbol{z}_{u,\theta\to k}, \boldsymbol{z}_x), \\ \mathrm{maxlogapp}^{\mathsf{ext}}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) &:= \mathrm{maxlogapp}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_{x,\theta\to n}),\end{aligned}$$

$k = 0, 1, \ldots, K-1$, $n = 0, 1, \ldots, N-1$. —

As required by extrinsic decoding functions (cf. Definition 3.4), $w_{u,k}$ does not depend on $z_{u,k}$, and $w_{x,n}$ does not depend on $z_{x,n}$.

Assume that the pre-decoding values are conditionally independent LLRs (cf. (3.21) and (3.22)). On the one hand, MaxLogAPP decoding approximates LogAPP decoding, i.e.,

$$\begin{aligned}v_{u,k} = \mathrm{maxlogapp}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) &\approx \mathrm{logapp}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) = L(U_k|\boldsymbol{z}_u, \boldsymbol{z}_x), \\ v_{x,n} = \mathrm{maxlogapp}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) &\approx \mathrm{logapp}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) = L(X_n|\boldsymbol{z}_u, \boldsymbol{z}_x),\end{aligned}$$

because of the approximation of the $\max^\star$ function by a maximization, as mentioned above. On the other hand, the post-decoding value for info symbols and code symbols

may be written as

$$v_{u,k} = \text{maxlogapp}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) = \ln \frac{\max\limits_{[\boldsymbol{u}\,\boldsymbol{x}]\in\mathcal{C}_{\mathsf{syxt}}(u_k=+1)} \Pr([\boldsymbol{u}\,\boldsymbol{x}]|\boldsymbol{z}_u, \boldsymbol{z}_x)}{\max\limits_{[\boldsymbol{u}\,\boldsymbol{x}]\in\mathcal{C}_{\mathsf{syxt}}(u_k=-1)} \Pr([\boldsymbol{u}\,\boldsymbol{x}]|\boldsymbol{z}_u, \boldsymbol{z}_x)}, \tag{3.30}$$

$$v_{x,n} = \text{maxlogapp}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) = \ln \frac{\max\limits_{[\boldsymbol{u}\,\boldsymbol{x}]\in\mathcal{C}_{\mathsf{syxt}}(x_n=+1)} \Pr([\boldsymbol{u}\,\boldsymbol{x}]|\boldsymbol{z}_u, \boldsymbol{z}_x)}{\max\limits_{[\boldsymbol{u}\,\boldsymbol{x}]\in\mathcal{C}_{\mathsf{syxt}}(x_n=-1)} \Pr([\boldsymbol{u}\,\boldsymbol{x}]|\boldsymbol{z}_u, \boldsymbol{z}_x)}. \tag{3.31}$$

Thus, the post-decoding value for a symbol gives the difference between the logarithmic probabilities of two code words, namely the most likely code word with this symbol being $+1$ and the most likely code word with this symbol being -1. Similarly, the extrinsic post-decoding values for info symbols and code symbols may be written as

$$w_{u,k} = \text{maxlogapp}^{\mathsf{ext}}_{\mathsf{info},k}(\boldsymbol{z}_u, \boldsymbol{z}_x) = \ln \frac{\max\limits_{[\boldsymbol{u}\,\boldsymbol{x}]\in\mathcal{C}_{\mathsf{syxt}}(u_k=+1)} \Pr([\boldsymbol{u}\,\boldsymbol{x}]|\boldsymbol{z}_{u,\setminus k}, \boldsymbol{z}_x)}{\max\limits_{[\boldsymbol{u}\,\boldsymbol{x}]\in\mathcal{C}_{\mathsf{syxt}}(u_k=-1)} \Pr([\boldsymbol{u}\,\boldsymbol{x}]|\boldsymbol{z}_{u,\setminus k}, \boldsymbol{z}_x)}, \tag{3.32}$$

$$w_{x,n} = \text{maxlogapp}^{\mathsf{ext}}_{\mathsf{code},n}(\boldsymbol{z}_u, \boldsymbol{z}_x) = \ln \frac{\max\limits_{[\boldsymbol{u}\,\boldsymbol{x}]\in\mathcal{C}_{\mathsf{syxt}}(x_n=+1)} \Pr([\boldsymbol{u}\,\boldsymbol{x}]|\boldsymbol{z}_u, \boldsymbol{z}_{x,\setminus n})}{\max\limits_{[\boldsymbol{u}\,\boldsymbol{x}]\in\mathcal{C}_{\mathsf{syxt}}(x_n=-1)} \Pr([\boldsymbol{u}\,\boldsymbol{x}]|\boldsymbol{z}_u, \boldsymbol{z}_{x,\setminus n})}. \tag{3.33}$$

For MaxLogAPP decoding, the relation between post-decoding values, $v_{u,k}$ and $v_{x,n}$, extrinsic post-decoding values, $w_{u,k}$ and $w_{x,n}$, and pre-decoding values (channel LLRs, which may be interpreted as a-priori LLRs, as discussed above), $z_{u,k}$, $z_{x,n}$ is the same as that for LogAPP decoding given in (3.28):

$$\begin{aligned} v_{u,k} &= w_{u,k} + z_{u,k}, \\ v_{x,n} &= w_{x,n} + z_{x,n}, \end{aligned} \tag{3.34}$$

$k = 0, 1, \ldots, K-1$ and $n = 0, 1, \ldots, N-1$.

Example 3.10
Consider the repetition code of length $N = 3$ ($K = 1$) from Example 3.1 and Example 3.7. As the subsets of $\mathcal{C}_{\mathsf{syxt}}$ contain only one code word, $\max^\star$ is identical to maximization, and accordingly, the LogAPP decoding function and the MaxLogAPP decoding function for this code are identical:

$$\begin{aligned} v_{u,0} &= \text{maxlogapp}_{\mathsf{info},0}(0, \boldsymbol{z}_x) \\ &= z_{x,0} + z_{x,1} + z_{x,2}. \end{aligned}$$

$\diamond$

Example 3.11
Consider the single parity check code of length $N = 3$ ($K = 2$) from Example 3.9. The extrinsic MaxLogAPP decoding function for code symbol X_0 is given by

$$\begin{aligned} v_{x,0} &= \text{maxlogapp}_{\mathsf{code},0}(\boldsymbol{0}, \boldsymbol{z}_x) = \text{maxlogapp}_{\mathsf{code},0}(\boldsymbol{0}, \boldsymbol{z}_{x,0\to 0}) \\ &= \max\Big\{\tfrac{1}{2}(+z_{x,1} + z_{x,2}), \tfrac{1}{2}(-z_{x,1} - z_{x,2})\Big\} \\ &\quad - \max\Big\{\tfrac{1}{2}(+z_{x,1} - z_{x,2}), \tfrac{1}{2}(-z_{x,1} + z_{x,2})\Big\}. \end{aligned}$$

As all code symbols are coupled by a single parity-check equation, the extrinsic post-decoding values may also be computed using an approximation for the operator $\boxplus$ [HOP96], denoted by[19] $\boxplus\!\!\!\!|\;$ (cf. Appendix C):

$$\begin{aligned} v_{x,0} &= \mathrm{maxlogapp}^{\mathsf{ext}}_{\mathsf{code},0}(\mathbf{0}, \boldsymbol{z}_x) \\ &= z_{x,1} \boxplus\!\!\!\!| \; z_{x,2} := \mathrm{sgn}(z_{x,1}) \cdot \mathrm{sgn}(z_{x,2}) \cdot \min\bigl\{|z_{x,1}|, |z_{x,2}|\bigr\}. \end{aligned}$$

$\diamond$

3.4.3 Optimality of MaxLogAPP Decoding

The post-decoding values may be used to obtain estimates for the info symbols or the code symbols, as described at the end of Section 3.2.2. LogAPP decoding is optimal with respect to the symbol error rate, and MaxLogAPP decoding is optimal with respect to the word error rate. As this property of MaxLogAPP decoding is less known, it is proved in the following.

Let $[\boldsymbol{u}^\bullet \boldsymbol{x}^\bullet] \in \mathcal{C}_{\mathsf{syxt}}$ denote the most likely code word of the systematically extended code $\mathcal{C}_{\mathsf{syxt}}$, i.e., the code word such that

$$\Pr([\boldsymbol{u}^\bullet \boldsymbol{x}^\bullet] | \boldsymbol{z}_u, \boldsymbol{z}_x) \geq \Pr([\boldsymbol{u}\,\boldsymbol{x}] | \boldsymbol{z}_u, \boldsymbol{z}_x) \quad \text{for all } [\boldsymbol{u}\,\boldsymbol{x}] \in \mathcal{C}_{\mathsf{syxt}},$$

or equivalently,

$$\tfrac{1}{2}\boldsymbol{z}_u \boldsymbol{u}^{\bullet\mathsf{T}} + \tfrac{1}{2}\boldsymbol{z}_x \boldsymbol{x}^{\bullet\mathsf{T}} \geq \tfrac{1}{2}\boldsymbol{z}_u \boldsymbol{u}^{\mathsf{T}} + \tfrac{1}{2}\boldsymbol{z}_x \boldsymbol{x}^{\mathsf{T}} \quad \text{for all } [\boldsymbol{u}\,\boldsymbol{x}] \in \mathcal{C}_{\mathsf{syxt}}.$$

Consider now (3.30). Let $\hat{u}_k$ denote the hard decision based on the post-decoding value $v_{u,k}$. Let further $[\boldsymbol{u}^+ \boldsymbol{x}^+]$ denote the code word maximizing the numerator, and let $[\boldsymbol{u}^- \boldsymbol{x}^-]$ denote the code word maximizing the denominator; notice that $u_k^+ = +1$ and $u_k^- = -1$. One of these two code words is the (overall) most likely code word $[\boldsymbol{u}^\bullet \boldsymbol{x}^\bullet]$, and so we may distinguish two cases:

$$\begin{aligned} [\boldsymbol{u}^+ \boldsymbol{x}^+] = [\boldsymbol{u}^\bullet \boldsymbol{x}^\bullet] \quad &\Rightarrow \quad v_{u,k} \geq 0, \\ [\boldsymbol{u}^- \boldsymbol{x}^-] = [\boldsymbol{u}^\bullet \boldsymbol{x}^\bullet] \quad &\Rightarrow \quad v_{u,k} \leq 0; \end{aligned}$$

the value of $v_{u,k}$ results from (3.30). If $v_{u,k} \neq 0$, then in either case, $\hat{u}_k = u_k^\bullet$. If $v_{u,k} = 0$, then $[\boldsymbol{u}^+ \boldsymbol{x}^+]$ and $[\boldsymbol{u}^- \boldsymbol{x}^-]$ are equiprobable, and $u_k^\bullet$ may arbitrarily be chosen; accordingly, $\hat{u}_k = -1$ and $\hat{u}_k = +1$ are equivalent. As these considerations hold for all u_k and, similarly, for all x_k, we have the result: the word $[\hat{\boldsymbol{u}}\,\hat{\boldsymbol{x}}]$ estimated from the post-decoding values of an MaxLogAPP decoder is the most likely code word. Thus MaxLogAPP decoding with subsequent hard decisions minimizes the word error rate (similarly to Viterbi decoding [Vit67, For73]).

An interesting property of MaxLogAPP decoding is its independence of the signal-to-noise ratio (SNR), when the communication channel is an BI-AWGNC [WHW00]. To be precise, only the hard-decisions are independent from the SNR, but not the soft-values. This property follows from two facts: (a) In the case of an BI-AWGNC, the channel LLR

[19] The S-shaped vertical line refers to the approximation.

depends linearly from the SNR of the channel, see Example 3.3. (b) MaxLogAPP decoding involves only maximizations over linear sums of channel LLRs, see Definition 3.10. Therefore, the post-decoding soft-values are also linearly dependent from the channel SNR. As the channel SNR is a positive value, the signs of the post-decoding soft-values are independent from the channel SNR, and thus so are the hard-decisions.

3.5 Concatenated Codes

Powerful codes that can be decoded with comparatively low complexity may be constructed by code concatenation. Examples of such concatenated codes are product codes [Eli54], generalized concatenated codes [ZSB99], or serially concatenated codes [For66]. This section deals with codes formed by concatenating convolutional codes or block codes, called constituent codes in this context, separated by interleavers. These codes are iteratively decoded using only symbol-by-symbol soft-output decoders for the constituent codes. Coding schemes using these kinds of codes and employing iterative decoders show high coding gains, in particular at low signal-to-noise ratios. Properly designed, they allow performance close to the Shannon limit.

The principles of this code construction and of the corresponding iterative decoder were introduced in [LHH92, LYHH93, BGT93, BG96]; product codes (which may be seen as serially concatenated codes) were addressed in [LYHH93], and parallel concatenated codes were addressed in [BGT93]. Due to the iterative decoder structure, these concatenated codes are also called turbo codes[20], and the iterative decoders are also called turbo decoders [BGT93]. In the sequel, we refer to these kinds of codes simply as concatenated codes.

Three special ingredients of coding schemes comprising concatenated codes and iterative decoders are responsible for the extraordinary performance:

- *Recursive convolutional encoders* are employed as constituent encoders [BGT93].
- *Soft-input soft-output decoders* are used to decode the constituent codes [LYHH93, BGT93].
- *Extrinsic soft-values*[21] are exchanged between the constituent decoders [LYHH93, BGT93].

The resulting codes show relatively good distance properties, and the iterative decoders enable near optimum decoding with relatively low decoding complexity.

These coding schemes are distinguished according to the kind of concatenation, which may be parallel, serial, or hybrid, and according to the number of concatenated codes. Parallel and serially concatenated codes comprising two constituent codes are addressed in the following two sections. Hybrid concatenated coding schemes are discussed in [DP97]. Information on multiple concatenated codes may be found in [BDMP98a, Hue04, Brä04]. First information theoretic results about whether concatenated codes can be capacity achieving are given in [Say03].

[20] Some authors use the term "turbo codes" only for parallel concatenated codes.

[21] This exchange of extrinsic soft-values is called "partial factor MAP filtering" in [LYHH93].

3.5.1 Parallel Concatenated Codes

Parallel concatenated codes (PCCs) were introduced in [BGT93, BG96]. We refer to the overall encoder as PCC encoder and to the overall (iterative) decoder as PCC decoder.

The original PCC, as introduced in [BGT93], is a systematic code, i.e., the PCC code word contains all info symbols. Replacing some systematic symbols by parity symbols leads to partially systematic PCCs, introduced in [LH00a]. Partially systematic PCCs have two special properties: (a) the number of low-weight code words decreases with the number of systematic symbols; thus, the error probability is smaller at high SNR; (b) the iterative decoder converges at lower signal-to-noise ratios if not too many systematic symbols are replaced; thus, the error probability is also smaller at low SNR. Hence, properly designed coding schemes with partially systematic PCCs outperform those with systematic PCCs. A concept similar to that of partially systematic PCCs, called code doping, was proposed in [tB00a, tB01b] for triggering the decoding convergence for codes of very long lengths.

In the sequel, we restrict ourselves to the basic encoding and decoding principles. Further information about design of PCCs, regarding constituent codes and interleavers, as well as analysis of the corresponding iterative decoders can be found in [BM96b,BM96a, RHV97, HW99, VY00, tB01c, Bre02, BMD03].

Encoder

The encoder for a PCC is depicted in Fig. 3.4. It comprises two binary linear encoders (ENC1 and ENC2), an interleaver (π), and a parallel-to-serial converter (P/S). The two encoders are referred to as Encoder 1 and Encoder 2, and they are called constituent encoders; the overall encoder is called PCC encoder.

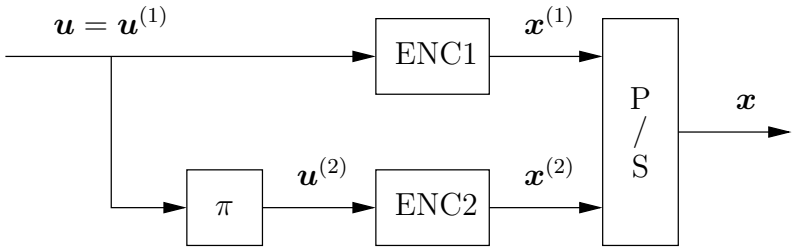


Figure 3.4: PCC encoder.

The PCC encoder maps an info word $\boldsymbol{u}$ to a code word $\boldsymbol{x}$ as follows: On the one hand, the info word $\boldsymbol{u}^{(1)} = \boldsymbol{u}$ is encoded by Encoder 1 to the code word $\boldsymbol{x}^{(1)}$. On the other hand, the info word is interleaved[22] to $\boldsymbol{u}^{(2)} = \mathrm{perm}_\pi\, \boldsymbol{u}$, according to a permutation function π, and is then encoded by Encoder 2 to the code word $\boldsymbol{x}^{(2)}$. The overall code word, called PCC word, is given by $\boldsymbol{x} = [\boldsymbol{x}^{(1)}\, \boldsymbol{x}^{(2)}]$. The info and code words are represented over $\mathbb{B}$.

The code generated by Encoder 1 is denoted by $\mathcal{C}_1$; it has info word length K, code word length N_1, and rate $R_1 := K/N_1$. Similarly, the code generated by Encoder 2 is

[22]For notation, see Appendix B.

denoted by $\mathcal{C}_2$; it has (the same) info word length K, code word length N_2, and rate $R_2 := K/N_2$. The codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are referred to as constituent codes. The code formed by parallel concatenation of the constituent codes, i.e., the set of code words $\boldsymbol{x}$, is called the parallel concatenated code (PCC), and it is denoted by $\mathcal{C}_{\mathsf{PCC}}$; it has info word length K, code word length $N = N_1 + N_2$, and code rate

$$R := \frac{K}{N} = \frac{R_1 R_2}{R_1 + R_2}.$$

Notice that $\mathcal{C}_{\mathsf{PCC}}$ is a subcode of $\mathcal{C}_1 \times \mathcal{C}_2$

As the constituent encoders and the PCC encoder are linear encoders, they can be described using generator matrices. For writing info and code words over $\mathbb{F}_2$, we use the notation introduced in Section 3.1 (for $b \in \mathbb{B}$, $\breve{b} = \mathrm{bpsk}^{-1}(b) \in \mathbb{F}_2$).

Let the generator matrices for $\mathcal{C}_1$ and $\mathcal{C}_2$ be denoted by

$$\boldsymbol{G}_1 \in \mathbb{F}_2^{K \times N_1}, \qquad \boldsymbol{G}_2 \in \mathbb{F}_2^{K \times N_2},$$

respectively. Typically, the constituent encoders are chosen such that $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ have a low trellis complexity (see e.g. [LV95, LKFF98, Var98]), to allow for low-complexity decoding. The interleaving is described by a permutation matrix (cf. Appendix B)

$$\boldsymbol{P} \in \mathbb{F}_2^{K \times K}$$

corresponding to the permutation function π such that

$$\mathrm{perm}_\pi \, \breve{\boldsymbol{u}} = \breve{\boldsymbol{u}} \boldsymbol{P}.$$

Using the generator matrices and the permutation matrix, the encoding of the constituent codes may be written as

$$\begin{aligned} \breve{\boldsymbol{u}}^{(1)} &= \breve{\boldsymbol{u}}, & \breve{\boldsymbol{u}}^{(2)} &= \breve{\boldsymbol{u}} \boldsymbol{P}, \\ \breve{\boldsymbol{x}}^{(1)} &= \breve{\boldsymbol{u}}^{(1)} \boldsymbol{G}_1 = \breve{\boldsymbol{u}} \boldsymbol{G}_1, & \breve{\boldsymbol{x}}^{(2)} &= \breve{\boldsymbol{u}}^{(2)} \boldsymbol{G}_2 = \breve{\boldsymbol{u}} \boldsymbol{P} \boldsymbol{G}_2. \end{aligned}$$

As

$$\breve{\boldsymbol{x}} = [\breve{\boldsymbol{x}}^{(1)} \, \breve{\boldsymbol{x}}^{(2)}],$$

the generator matrix for the PCC encoder is given by

$$\boldsymbol{G} := \begin{bmatrix} \boldsymbol{G}_1 & \boldsymbol{P} \boldsymbol{G}_2 \end{bmatrix} \in \mathbb{F}_2^{K \times N}, \tag{3.35}$$

and the overall encoding can be written as

$$\breve{\boldsymbol{x}} = \breve{\boldsymbol{u}} \boldsymbol{G}.$$

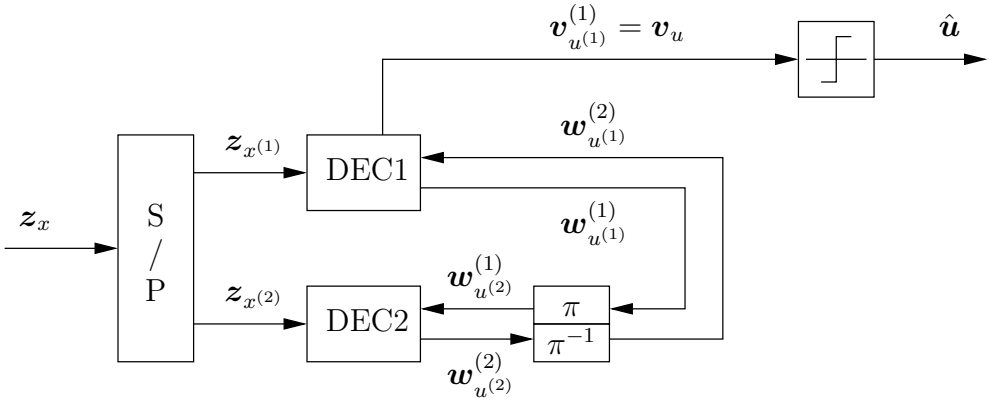


Figure 3.5: PCC decoder.

Iterative Decoder

An iterative decoder for a PCC is depicted in Fig. 3.5. It comprises a serial-to-parallel converter (S/P), two decoders (DEC1 and DEC2), an interleaver (π), and a deinterleaver (π^{-1}). The two decoders, referred to as Decoder 1 and Decoder 2, correspond to the two constituent encoders, and they are called constituent decoders. Commonly, extrinsic LogAPP or extrinsic MaxLogAPP decoders are employed as constituent decoders. The overall decoder is called PCC decoder.

The channel LLR word $\boldsymbol{z}_x$, corresponding to the PCC word $\boldsymbol{x}$, is separated into $\boldsymbol{z}_{x^{(1)}}$ and $\boldsymbol{z}_{x^{(2)}}$ corresponding to the constituent code words $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$, respectively; these are given to the corresponding constituent decoders Decoder 1 and Decoder 2. For iterative decoding, the two constituent decoders exchange extrinsic values for info symbols, $\boldsymbol{w}^{(1)}_{u^{(2)}} = \mathrm{perm}_\pi\, \boldsymbol{w}^{(1)}_{u^{(1)}}$ and $\boldsymbol{w}^{(2)}_{u^{(2)}} = \mathrm{perm}_\pi\, \boldsymbol{w}^{(2)}_{u^{(1)}}$. After the last iteration, Decoder 1 computes the word of complete post-decoding values $\boldsymbol{v}^{(1)}_{u^{(1)}}$ corresponding to $\boldsymbol{u}^{(1)}$. This represents the overall post-decoding word $\boldsymbol{v}_u = \boldsymbol{v}^{(1)}_{u^{(1)}}$, corresponding to $\boldsymbol{u}$, computed by the PCC decoder. A subsequent hard decision gives the estimated info word $\hat{\boldsymbol{u}}$. Notice the notation for soft-values: the subindex indicates which info or code word the soft-values correspond to, and the superindex indicates by which decoder the soft-values were computed. The decoding process is discussed in more detail in the sequel.

Each iteration of the *decoding process* consists of two steps, called half-iterations. Decoder 1 operates in the first half-iteration, and Decoder 2 operates in the second half-iteration.

Decoder 1: The soft-values available to Decoder 1 are $\boldsymbol{z}_{x^{(1)}}$ and

$$\boldsymbol{w}^{(2)}_{u^{(1)}} := \mathrm{perm}_\pi^{-1}\, \boldsymbol{w}^{(2)}_{u^{(2)}}. \tag{3.36}$$

In the first iteration, $\boldsymbol{w}^{(2)}_{u^{(2)}}$ is set to the all-zero word, and in the other iterations, it is the decoding result of Decoder 2 from the previous half-iteration.

The transmission model assumed by Decoder 1 is as follows (cf. Section 3.2): Info word $\boldsymbol{u}^{(1)}$ was encoded by Encoder 1 to code word $\boldsymbol{x}^{(1)}$, so that $[\boldsymbol{u}^{(1)}\, \boldsymbol{x}^{(1)}] \in \mathcal{C}_{1,\mathsf{syxt}}$.

The words $\boldsymbol{u}^{(1)}$ and $\boldsymbol{x}^{(1)}$ were transmitted over BISMCs and the resulting words of channel LLRs are $\boldsymbol{w}^{(2)}_{u^{(1)}}$ and $\boldsymbol{z}_{x^{(1)}}$, respectively.

Using the pre-decoding words $\boldsymbol{w}^{(2)}_{u^{(1)}}$ and $\boldsymbol{z}_{x^{(1)}}$, and taking into account the code constraints due to $\mathcal{C}_{1,\mathsf{syxt}}$, Decoder 1 computes extrinsic values for its info symbols:

$$\boldsymbol{w}^{(1)}_{u^{(1)}} := \mathbf{dec}^{\mathsf{ext}}_{\mathsf{info}}(\boldsymbol{w}^{(2)}_{u^{(1)}}, \boldsymbol{z}_{x^{(1)}} \,\|\, \mathcal{C}_{1,\mathsf{syxt}}). \tag{3.37}$$

Decoder 2: The soft-values available to Decoder 2 are $\boldsymbol{z}_{x^{(2)}}$ and

$$\boldsymbol{w}^{(1)}_{u^{(2)}} := \mathrm{perm}_{\pi}\, \boldsymbol{w}^{(1)}_{u^{(1)}}. \tag{3.38}$$

The word $\boldsymbol{w}^{(1)}_{u^{(1)}}$ is the decoding result of Decoder 1 from the previous half-iteration.

The transmission model assumed by Decoder 2 is as follows (cf. Section 3.2): Info word $\boldsymbol{u}^{(2)}$ was encoded by Encoder 2 to code word $\boldsymbol{x}^{(2)}$, so that $[\boldsymbol{u}^{(2)}\, \boldsymbol{x}^{(2)}] \in \mathcal{C}_{2,\mathsf{syxt}}$. The words $\boldsymbol{u}^{(2)}$ and $\boldsymbol{x}^{(2)}$ were transmitted over BISMCs and the resulting words of channel LLRs are $\boldsymbol{w}^{(1)}_{u^{(2)}}$ and $\boldsymbol{z}_{x^{(2)}}$, respectively.

Using the pre-decoding words $\boldsymbol{w}^{(1)}_{u^{(2)}}$ and $\boldsymbol{z}_{x^{(2)}}$, and taking into account the code constraints due to $\mathcal{C}_{2,\mathsf{syxt}}$, Decoder 2 computes extrinsic values for its info symbols:

$$\boldsymbol{w}^{(2)}_{u^{(2)}} := \mathbf{dec}^{\mathsf{ext}}_{\mathsf{info}}(\boldsymbol{w}^{(1)}_{u^{(2)}}, \boldsymbol{z}_{x^{(2)}} \,\|\, \mathcal{C}_{2,\mathsf{syxt}}). \tag{3.39}$$

The iterative decoding scheme is constituted by (3.36), (3.37), (3.38), and (3.39).

After the last iteration, Decoder 1 computes complete post-decoding values for its info symbols, using the same assumptions as given above:

$$\boldsymbol{v}^{(1)}_{u^{(1)}} := \mathbf{dec}_{\mathsf{info}}(\boldsymbol{w}^{(2)}_{u^{(1)}}, \boldsymbol{z}_{x^{(1)}} \,\|\, \mathcal{C}_{1,\mathsf{syxt}}). \tag{3.40}$$

These values represent the final post-decoding values of the PCC decoder, i.e.,

$$\boldsymbol{v}_u := \boldsymbol{v}^{(1)}_{u^{(1)}}.$$

Alternatively, Decoder 2 may be used in an analogous way to generate the final post-decoding values of the PCC decoder.

The iterative decoder typically converges quickly. In some cases, however, it converges only slowly, or it does not converge at all. (Some aspects regarding optimality are discussed in the next section.) Thus, the decoding performance can be improved for a given average number of iterations by terminating the iteration as soon as convergence or non-convergence is detected. For that purpose, various *stopping criteria* have been proposed in literature [Rob94, HOP96, NS97, Hám98, SLF99, WWE00, MDP00, LH01]. All stopping criteria follow the same principle: After each half or each full iteration, a function of the extrinsic values is computed; the iteration is terminated either if the function value reaches a certain threshold, or if the difference between two subsequent function values (from one iteration to the next) reaches a certain threshold. For simulation, a *genie criterion* may be employed: The iteration is terminated as soon as the estimated info word is equal to the transmitted info word. The resulting word error rate represents a lower bound on word error rates achievable with real stopping criteria.

EXIT Chart

The behavior of the iterative decoder may be visualized using the EXIT chart method [tB01c]. Presuming very large code lengths, the EXIT chart method allows for an analysis of the iterative decoder and a very accurate prediction of the decoding threshold, which is the parameter of the worst channel (for a given channel model) such that quasi error-free transmission can be assured. For an AWGN channel, the decoding threshold is typically given in terms of signal-to-noise threshold, and for a BSC, the decoding threshold is typically given in terms of the crossover probability. As for a given channel model, there is a one-to-one correspondence between the channel parameter and the channel capacity, the decoding threshold may equivalently be given in terms of the channel capacity. This representation is preferred in this thesis.

The EXIT chart for a PCC depicts the info-symbol EXIT functions for the two constituent decoders (cf. Section 3.3). For the computation or the EXIT functions, the a-priori channels are typically modeled as AWGN channels. The values of symbol-wise mutual information associated to the inputs and to the outputs of the two constituent decoders are

$$\begin{aligned} I_{\mathsf{ext}}^{(1)} := I(U^{(1)}; W_{u^{(1)}}^{(1)}) &= I(U^{(2)}; W_{u^{(2)}}^{(1)}) =: I_{\mathsf{apri}}^{(2)}, \\ I_{\mathsf{ext}}^{(2)} := I(U^{(2)}; W_{u^{(2)}}^{(2)}) &= I(U^{(1)}; W_{u^{(1)}}^{(2)}) =: I_{\mathsf{apri}}^{(1)}; \end{aligned}$$

the equalities hold because interleaving does not change mutual information.

Regarding these equalities, the EXIT functions for Decoder 1 and Decoder 2 are given as

$$\mathrm{itf}^{(1)} : I_{\mathsf{ext}}^{(2)} \mapsto I_{\mathsf{ext}}^{(1)}, \tag{3.41}$$

$$\mathrm{itf}^{(2)} : I_{\mathsf{ext}}^{(1)} \mapsto I_{\mathsf{ext}}^{(2)}; \tag{3.42}$$

the mutual information of the communication channel, $I_{\mathsf{ch}} := I(X; Z_X)$, is used as parameter. The iterative decoder can converge only if the two EXIT functions do not have an intersection. For details about the EXIT chart method for PCCs, we refer the reader to [tB01c].

The EXIT chart method is illustrated for a low-density parity-check code in Example 3.15.

Optimality of the Decoder

The use of LogAPP decoders as constituent decoders is often considered as optimal in literature. Employing LogAPP decoding in (3.36), (3.37), (3.38), and (3.39), the iterative decoder is given by the following set of equations:

$$\boldsymbol{w}_{u^{(1)}}^{(2)} := \mathrm{perm}_{\pi}^{-1}\, \boldsymbol{w}_{u^{(2)}}^{(2)}, \tag{3.43a}$$

$$\boldsymbol{w}_{u^{(1)}}^{(1)} := \mathbf{logapp}_{\mathsf{info}}^{\mathsf{ext}}(\boldsymbol{w}_{u^{(1)}}^{(2)}, \boldsymbol{z}_{x^{(1)}} \,\|\, \mathcal{C}_{1,\mathsf{syxt}}), \tag{3.43b}$$

$$\boldsymbol{w}_{u^{(2)}}^{(1)} := \mathrm{perm}_{\pi}\, \boldsymbol{w}_{u^{(1)}}^{(1)}, \tag{3.43c}$$

$$\boldsymbol{w}_{u^{(2)}}^{(2)} := \mathbf{logapp}_{\mathsf{info}}^{\mathsf{ext}}(\boldsymbol{w}_{u^{(2)}}^{(1)}, \boldsymbol{z}_{x^{(2)}} \,\|\, \mathcal{C}_{2,\mathsf{syxt}}). \tag{3.43d}$$

The words $\boldsymbol{z}_{x^{(1)}}$ and $\boldsymbol{z}_{x^{(2)}}$ are assumed to be pre-decoding words consisting of conditionally independent LLRs (cf. (3.21) and (3.22)). The optimality of the resulting iterative decoding scheme is discussed in the sequel.

First, we address the role of LogAPP decoders during iterative decoding. We start with the very first iteration, where $\boldsymbol{w}^{(2)}_{u^{(1)}} = \mathbf{0}$. Since the pre-decoding word $[\boldsymbol{w}^{(2)}_{u^{(1)}}\, \boldsymbol{z}^{(1)}_{x}]$ consists of conditionally independent LLRs, the extrinsic word $\boldsymbol{w}^{(1)}_{u^{(1)}}$ computed by Decoder 1, (3.43b), consists of LLRs; but for obvious reasons, these LLRs are not conditionally independent. Due to interleaving, (3.43c), the local dependencies in $\boldsymbol{w}^{(1)}_{u^{(2)}}$ are smaller than that in $\boldsymbol{w}^{(1)}_{u^{(1)}}$, but they are not removed. The extrinsic word $\boldsymbol{w}^{(2)}_{u^{(2)}}$ computed by Decoder 2 in the second half-iteration, (3.43d), contains values that are neither LLRs nor independent. Therefore, the deinterleaved word $\boldsymbol{w}^{(2)}_{u^{(1)}}$ does not have these properties either. In all following decoding steps, the outputs of Decoder 1 and Decoder 2 will not be LLRs.

We see from this discussion that only the decoding step in the first half-iteration is optimal in the sense of LogAPP decoding, whereas the others are not. Hence, though LogAPP decoding is optimal with respect to minimum symbol error probability and maximal symbol-wise mutual information for single decoders (cf. Section 3.4.1), it is not optimal with respect to these criteria when used for constituent decoders within a PCC decoders. Nevertheless, it may be optimum in some other sense. A possible improvement, namely the "correction" of the extrinsic values to LLRs, is investigated in Section 4.5.

Not only the behavior of LogAPP decodes during iterative decoding is of interest, but also the solution of the set of LogAPP decoding equations given in (3.43). Let $\boldsymbol{w}^{(1)*}_{u^{(1)}}$ and $\boldsymbol{w}^{(2)*}_{u^{(1)}}$ denote two extrinsic words that fulfill (3.43) simultaneously; for convenience, we define $\boldsymbol{w}^{*} := [\boldsymbol{w}^{(1)*}_{u^{(1)}}\, \boldsymbol{w}^{(2)*}_{u^{(1)}}]$. The vector $\boldsymbol{w}^{*}$ represents a solution, i.e., a *fixed point* of (3.43). The info word corresponding to this fixed point is denoted by $\hat{\boldsymbol{u}}^{*}$. On the other hand, consider an overall LogAPP decoder for the PCC:

$$\boldsymbol{v}^{\bullet}_{u} := \mathbf{logapp}_{\mathsf{info}}(\mathbf{0}, [\boldsymbol{z}_{x^{(1)}}\, \boldsymbol{z}_{x^{(2)}}] \,\|\, \mathcal{C}_{\mathsf{PCC,syxt}}).$$

The info word corresponding to this optimal solution is denoted by $\hat{\boldsymbol{u}}^{\bullet}$.

A fundamental question is the *relation between a fixed-point solution and an optimal solution*, i.e., the relation between $\hat{\boldsymbol{u}}^{*}$ and $\hat{\boldsymbol{u}}^{\bullet}$. Since the iterative decoder achieves very low error rates, the two solutions are identical in most cases. On the other hand, a remarkable result could be proved for parallel concatenated single parity check codes [FB03]: A fixed point always exists and it is unique, so that the iterative decoder always converges to this fixed point. Simulations showed that the error rates for iterative decoding and those for optimal decoding are different, so that we can conclude: the estimated info word corresponding to the fixed point and that corresponding to the optimal solution are not always the same. Therefore, using LogAPP decoders as constituent decoders is not optimal even if the solution is determined in a single step, corresponding to a fixed point, rather than iteratively.

This gives rise to two questions: (1) When employing LogAPP decoding functions, what does a fixed point mean? (2) Are there decoding functions such that a fixed point corresponds to the optimal solution? Whereas the first question is still open, the latter

question is addressed in [Sor02]. Constituent decoders are constructed such that the iterative decoder converges to the optimal solution whenever it converges at all. Thus, it can be proved that the most likely info word is found when noise on the communication channel is low. A similar proof does not exist for iterative decoders employing LogAPP decoders. Further information about fixed points and convergence of the iterative decoding process can be found in [Moq02, IEE01].

Code Structure and Parity-Check Matrices

The iterative decoder does not operate on the PCC $\mathcal{C}_{\mathsf{PCC}}$, but on a code that results from extending the original code $\mathcal{C}_{\mathsf{PCC}}$ with its info symbols. This code is called embedding code[23] of $\mathcal{C}_{\mathsf{PCC}}$, and it is denoted by $\mathcal{C}_{\mathsf{PCC,emb}}$. Using the generator and parity-check matrices of the constituent codes, we derive parity-check matrices of $\mathcal{C}_{\mathsf{PCC}}$ and $\mathcal{C}_{\mathsf{PCC,emb}}$. The structures of these parity-check matrices are then related to the structures of the iterative decoder.

For the constituent codes $\mathcal{C}_1$ and $\mathcal{C}_2$, let

$$\boldsymbol{G}_1 \in \mathbb{F}_2^{K \times N_1}, \qquad \boldsymbol{G}_2 \in \mathbb{F}_2^{K \times N_2}$$

denote the generator matrices, and let

$$\boldsymbol{H}_1 \in \mathbb{F}_2^{(N_1-K) \times N_1}, \qquad \boldsymbol{H}_2 \in \mathbb{F}_2^{(N_2-K) \times N_2}$$

denote parity-check matrices, respectively. For simplicity, $K < N_1$ and $K < N_2$ is assumed [24]. Similarly to (3.10), let inverse generator matrices $\boldsymbol{A}_1^{\mathsf{T}}$ and $\boldsymbol{A}_2^{\mathsf{T}}$ for $\mathcal{C}_1$ and $\mathcal{C}_2$ be defined as

$$\begin{aligned} \boldsymbol{A}_1 \in \mathbb{F}_2^{K \times N_1} \quad &: \quad \boldsymbol{G}_1 \boldsymbol{A}_1^{\mathsf{T}} = \boldsymbol{I}, \\ \boldsymbol{A}_2 \in \mathbb{F}_2^{K \times N_2} \quad &: \quad \boldsymbol{G}_2 \boldsymbol{A}_2^{\mathsf{T}} = \boldsymbol{I}, \end{aligned}$$

respectively.

The embedding code $\mathcal{C}_{\mathsf{PCC,emb}}$ employed for iterative decoding of $\mathcal{C}_{\mathsf{PCC}}$ is the systematically extended code of $\mathcal{C}_{\mathsf{PCC}}$ (cf. Section 3.1), i.e., the code words of $\mathcal{C}_{\mathsf{PCC,emb}}$ comprise the info word and the PCC word:

$$\boldsymbol{x}_{\mathsf{emb}} := [\boldsymbol{u}\, \boldsymbol{x}] = [\boldsymbol{u}\, \boldsymbol{x}^{(1)}\, \boldsymbol{x}^{(2)}].$$

Thus, the generator matrix for $\mathcal{C}_{\mathsf{PCC,emb}}$ is given by

$$\boldsymbol{G}_{\mathsf{emb}} := \begin{bmatrix} \boldsymbol{I} & \boldsymbol{G} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{G}_1 & \boldsymbol{P}\boldsymbol{G}_2 \end{bmatrix} \in \mathbb{F}_2^{K \times (K+N_1+N_2)}. \tag{3.44}$$

Since $\boldsymbol{G}_{\mathsf{emb}}$ is a systematic generator matrix, a parity-check matrix for $\mathcal{C}_{\mathsf{PCC,emb}}$ can be determined straight-forward as

$$\boldsymbol{H}_{\mathsf{emb}} := \begin{bmatrix} \boldsymbol{G}^{\mathsf{T}} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{G}_1^{\mathsf{T}} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{G}_2^{\mathsf{T}} \boldsymbol{P}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \in \mathbb{F}_2^{(N_1+N_2) \times (K+N_1+N_2)}. \tag{3.45}$$

[23] In the case of PCCs (but only there), the embedding code is equal to the systematically extended code.

[24] A generalization is straight-forward.

Regarding $(\boldsymbol{P}\boldsymbol{G}_2)(\boldsymbol{P}\boldsymbol{A}_2)^{\mathsf{T}} = \boldsymbol{I}$ and (3.12), we obtain a second parity-check matrix for $\mathcal{C}_{\mathsf{PCC,emb}}$:

$$\boldsymbol{H}_{\mathsf{emb},A} := \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{H}_2 \end{bmatrix} \in \mathbb{F}_2^{(N_1+N_2)\times(K+N_1+N_2)}. \tag{3.46}$$

Notice that all rows of $\boldsymbol{H}_{\mathsf{emb},A}$ are linearly independent.

Starting with matrix $\boldsymbol{H}_{\mathsf{emb},A}$, a parity-check matrix for $\mathcal{C}_{\mathsf{PCC}}$ can be derived as follows. First, we add in $\boldsymbol{H}_{\mathsf{emb},A}$ the first row (of matrices) to the third row (of matrices):

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_1 & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{H}_2 \end{bmatrix};$$

the resulting matrix still has full rank. When splitting the check equation[25]

$$\begin{bmatrix} \breve{\boldsymbol{u}} & \breve{\boldsymbol{x}} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_1 & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{H}_2 \end{bmatrix}^T = \boldsymbol{0}$$

into two parts, as

$$\begin{aligned} \begin{bmatrix} \breve{\boldsymbol{u}} & \breve{\boldsymbol{x}} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A}_1 & \boldsymbol{0} \end{bmatrix}^T &= \boldsymbol{0} \quad \text{and} \\ \begin{bmatrix} \breve{\boldsymbol{u}} & \breve{\boldsymbol{x}} \end{bmatrix} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_1 & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{H}_2 \end{bmatrix}^T &= \boldsymbol{0}, \end{aligned}$$

we see immediately that

$$\boldsymbol{H}_A := \begin{bmatrix} \boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{A}_1 & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_2 \end{bmatrix} \in \mathbb{F}_2^{(N_1+N_2-K)\times(N_1+N_2)} \tag{3.47}$$

represents a parity-check matrix for $\mathcal{C}_{\mathsf{PCC}}$. Notice that $\boldsymbol{H}_A$ comprises only the parity-check matrices and inverse generator matrices of the constituent codes, which is indicated by index $\boldsymbol{A}$.

The structures of the parity-check matrices are directly related to the structure of the iterative decoder. Consider first the parity check matrix for the embedding code, given in (3.45):

$$\boldsymbol{H}_{\mathsf{emb}} = \begin{bmatrix} \boldsymbol{G}_1^{\mathsf{T}} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{G}_2^{\mathsf{T}}\boldsymbol{P}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{matrix} \longleftarrow \\ \longleftarrow \end{matrix} \begin{matrix} \text{Decoder 1} \\ \text{Decoder 2} \end{matrix}$$

The parity-check constraints defined by the upper part are taken into account by Decoder 1, and those defined by the lower part are taken into account by Decoder 2. These

[25] Notice the notation for binary symbols in $\mathbb{F}_2$ and in $\mathbb{B}$, as introduced in Section 3.1.

two sets of parity-check constraints are coupled only via the info symbol positions, namely the first K columns. Similarly, the two constituent decoders are coupled by exchanging information about these info symbols.

Conversely, these relations may be exploited to find an iterative decoding structure for a given binary linear code. First, the parity check matrix of the given code has to be converted into the form of $\boldsymbol{H}_{\mathsf{emb}}$, given in (3.45). Such a conversion is always possible (if necessary, using an equivalent code), and it is generally not unique. Then, the matrices corresponding to $\boldsymbol{G}_1$, $\boldsymbol{G}_2$, and $\boldsymbol{P}$ can be determined. The problem is that the matrices corresponding to $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are required to have low trellis complexity such that soft-output decoding is feasible. If this is the case, we can interpret the given code as a PCC and decode it using the iterative decoding structure of this PCC. A first approach of this kind was proposed in [Ung03] for decoding of Reed-Solomon codes over extensions fields of $\mathbb{F}_2$.

A more general approach may be based on the parity-check matrix for the PCC, given in (3.47). If a given binary linear code has a parity-check matrix that has a structure similar to $\boldsymbol{H}_A$, we may again determine matrices $\boldsymbol{G}_1$, $\boldsymbol{G}_2$, and $\boldsymbol{P}$ such that this code can be interpreted as a PCC. If $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ show low trellis complexity, the iterative decoding structure of the corresponding PCC can be applied for decoding. As "matching" a parity-check matrix of a given code to the structure of $\boldsymbol{H}_{\mathsf{emb}}$ gives fewer degrees of freedom than "matching" it to the structure of $\boldsymbol{H}_A$, the latter method may be applicable to a larger class of codes than the former.

Besides finding iterative decoders for linear binary codes, the derived parity-check matrices for the PCC and its embedding code may also be used for code analysis or code design.

3.5.2 Serially Concatenated Codes

Motivated by the extraordinary performance of parallel concatenated codes, as considered in the previous section, serially concatenated codes (SCCs) that are constructed in a similar way were investigated in [BDMP98b]. In some special cases, PCCs and SCCs and the corresponding iterative decoders are even (almost) equivalent, as shown in [HtBH01]. We refer to the overall encoder as SCC encoder and to the overall (iterative) decoder as SCC decoder.

In the sequel, we restrict ourselves to the basic encoding and decoding principles. Further information about analysis and design of SCCs, regarding constituent codes and interleavers, and the corresponding iterative decoders may be found in [BDMP98a, BDMP98b, HW99, VY00, tB00b, tB01a].

Encoder

The encoder for a serially concatenated code (SCC) comprises two binary linear encoders (ENC1 and ENC2), and an interleaver (π), as shown in Fig. 3.6. The two encoders are referred to as Encoder 1 and Encoder 2, and they are called constituent encoders; the overall encoder is called SCC encoder.

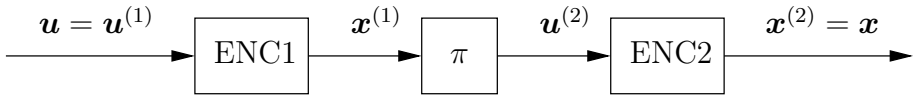


Figure 3.6: SCC encoder.

The SCC encoder maps an info word $\boldsymbol{u}$ to a code word $\boldsymbol{x}$ as follows: The info word $\boldsymbol{u}^{(1)} = \boldsymbol{u}$ is encoded by Encoder 1 to the code word $\boldsymbol{x}^{(1)}$. This code word is interleaved[26], according to a permutation function π, to $\boldsymbol{u}^{(2)} = \operatorname{perm}_\pi \boldsymbol{x}^{(1)}$ and then encoded by Encoder 2 to the code word $\boldsymbol{x}^{(2)}$. This represents the overall code word $\boldsymbol{x} = \boldsymbol{x}^{(2)}$, called SCC word. The info and code words are represented over $\mathbb{B}$.

The code generated by Encoder 1 is denoted by $\mathcal{C}_1$, and it is also called the outer code; it has info word length K, code word length N_1, and rate $R_1 := K/N_1$. Similarly, the code generated by Encoder 2 is denoted by $\mathcal{C}_2$, and it is also called the inner code; it has info word length N_1, code word length N_2, and rate $R_2 := N_1/N_2$. The codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are referred to as constituent codes. The code formed by serial concatenation of the constituent codes, i.e., the set of code words $\boldsymbol{x}$, is called the serially concatenated code (SCC), and it is denoted by $\mathcal{C}_{\mathsf{SCC}}$; it has info word length K, code word length $N = N_2$, and code rate

$$R := \frac{K}{N} = R_1 R_2.$$

Notice that $\mathcal{C}_{\mathsf{SCC}}$ is a subcode of $\mathcal{C}_2$.

As the constituent encoders and the SCC encoder are linear encoders, they can be described using generator matrices. For writing info and code words over $\mathbb{F}_2$, we use the notation introduced in Section 3.1 (for $b \in \mathbb{B}$, $\breve{b} = \operatorname{bpsk}^{-1}(b) \in \mathbb{F}_2$).

Let the generator matrices for $\mathcal{C}_1$ and $\mathcal{C}_2$ be denoted by

$$\boldsymbol{G}_1 \in \mathbb{F}_2^{K \times N_1}, \qquad \boldsymbol{G}_2 \in \mathbb{F}_2^{N_1 \times N_2},$$

respectively. Typically, the constituent encoders are chosen such that $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ have a low trellis complexity[27] to provide for low decoding complexity. The interleaving is described by a permutation matrix (cf. Appendix B)

$$\boldsymbol{P} \in \mathbb{F}_2^{N_1 \times N_1}$$

corresponding to the permutation function π such that

$$\operatorname{perm}_\pi \breve{\boldsymbol{x}}^{(1)} = \breve{\boldsymbol{x}}^{(1)} \boldsymbol{P}.$$

Using the generator matrices and the permutation matrix, we have for the constituent encodings

$$\breve{\boldsymbol{u}}^{(1)} = \breve{\boldsymbol{u}}, \qquad \breve{\boldsymbol{u}}^{(2)} = \breve{\boldsymbol{x}}^{(1)} \boldsymbol{P},$$
$$\breve{\boldsymbol{x}}^{(1)} = \breve{\boldsymbol{u}}^{(1)} \boldsymbol{G}_1 = \breve{\boldsymbol{u}} \boldsymbol{G}_1, \qquad \breve{\boldsymbol{x}}^{(2)} = \breve{\boldsymbol{u}}^{(2)} \boldsymbol{G}_2 = \breve{\boldsymbol{x}}^{(1)} \boldsymbol{P} \boldsymbol{G}_2.$$

[26] For notation, see Appendix B.

[27] Regarding trellises and their complexity, see [LV95, LKFF98, Var98].

As

$$\breve{\boldsymbol{x}} = \breve{\boldsymbol{x}}^{(2)},$$

the generator matrix for the SCC encoder is given by

$$\boldsymbol{G} := \boldsymbol{G}_1 \boldsymbol{P} \boldsymbol{G}_2 \in \mathbb{F}_2^{K \times N_2}, \tag{3.48}$$

and the overall encoding can be written as

$$\breve{\boldsymbol{x}} = \breve{\boldsymbol{u}} \boldsymbol{G}.$$

Iterative Decoder

An iterative decoder for an SCC is depicted in Fig. 3.7. It comprises two decoders (DEC1 and DEC2), an interleaver (π), and a deinterleaver (π^{-1}). The two decoders, referred to as Decoder 1 and Decoder 2, correspond to the two constituent encoders, and they are called constituent decoders. Commonly, extrinsic LogAPP or extrinsic MaxLogAPP decoders are employed as constituent decoders. The overall decoder is called SCC decoder.

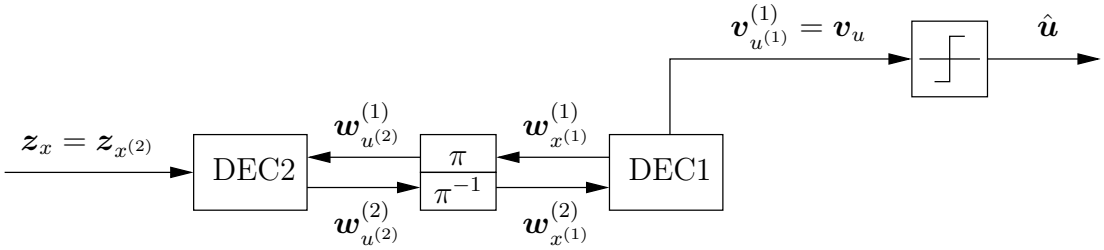


Figure 3.7: SCC decoder.

The channel LLR word $\boldsymbol{z}_x$ corresponding to the SCC code word $\boldsymbol{x}$ is equal to the word of channel LLRs $\boldsymbol{z}_{x^{(2)}}$ corresponding to code word $\boldsymbol{x}^{(2)}$, as $\boldsymbol{x} = \boldsymbol{x}^{(2)}$, i.e., $\boldsymbol{z}_{x^{(2)}} = \boldsymbol{z}_x$; the word $\boldsymbol{z}_{x^{(2)}}$ is given to Decoder 2. For iterative decoding, the two constituent decoders exchange extrinsic values, $\boldsymbol{w}^{(2)}_{x^{(1)}} = \mathrm{perm}_\pi^{-1} \boldsymbol{w}^{(2)}_{u^{(2)}}$ and $\boldsymbol{w}^{(1)}_{u^{(2)}} = \mathrm{perm}_\pi \boldsymbol{w}^{(1)}_{x^{(1)}}$. After the last iteration, Decoder 1 computes complete post-decoding values $\boldsymbol{v}^{(1)}_{u^{(1)}}$ corresponding to $\boldsymbol{u}^{(1)}$. This represents the overall post-decoding word $\boldsymbol{v}_u = \boldsymbol{v}^{(1)}_{u^{(1)}}$, corresponding to $\boldsymbol{u}$, computed by the SCC decoder. A subsequent hard decision gives the estimated info word $\hat{\boldsymbol{u}}$. Notice the notation for soft-values: the subindex indicates which info or code word the soft-values correspond to, and the superindex indicates by which decoder the soft-values were computed. The decoding process is now discussed in more detail.

Each iteration of the *decoding process* consists of two steps, called half-iterations. Decoder 2 operates in the first half-iteration, and Decoder 1 operates in the second half-iteration.

Decoder 2: The soft-values available to Decoder 2 are $\boldsymbol{z}_{x^{(2)}}$ and

$$\boldsymbol{w}^{(1)}_{u^{(2)}} := \mathrm{perm}_\pi \boldsymbol{w}^{(1)}_{x^{(1)}}. \tag{3.49}$$

In the first iteration, $\boldsymbol{w}^{(1)}_{u^{(1)}}$ is set to the all-zero word, and in the other iterations, it is the decoding result of Decoder 1 from the previous half-iteration.

The transmission model assumed by Decoder 2 is as follows (cf. Section 3.2): Info word $\boldsymbol{u}^{(2)}$ was encoded by Encoder 2 to code word $\boldsymbol{x}^{(2)}$, so that $[\boldsymbol{u}^{(2)}\,\boldsymbol{x}^{(2)}] \in \mathcal{C}_{2,\mathsf{syxt}}$. The words $\boldsymbol{u}^{(2)}$ and $\boldsymbol{x}^{(2)}$ were transmitted over BISMCs, and the resulting words of channel LLRs are $\boldsymbol{w}^{(1)}_{u^{(2)}}$ and $\boldsymbol{z}_{x^{(2)}}$, respectively.

Using the pre-decoding words $\boldsymbol{w}^{(1)}_{u^{(2)}}$ and $\boldsymbol{z}_{x^{(2)}}$ and taking into account the code constraints due to $\mathcal{C}_{2,\mathsf{syxt}}$, Decoder 2 computes extrinsic values for its info symbols:

$$\boldsymbol{w}^{(2)}_{u^{(2)}} := \mathbf{dec}^{\mathsf{ext}}_{\mathsf{info}}(\boldsymbol{w}^{(1)}_{u^{(2)}}, \boldsymbol{z}_{x^{(2)}} \,\|\, \mathcal{C}_{2,\mathsf{syxt}}). \tag{3.50}$$

Decoder 1: The soft-values available to Decoder 1 are

$$\boldsymbol{w}^{(2)}_{x^{(1)}} := \mathrm{perm}_{\pi}^{-1}\, \boldsymbol{w}^{(2)}_{u^{(2)}}. \tag{3.51}$$

The word $\boldsymbol{w}^{(2)}_{u^{(2)}}$ is the decoding result of Decoder 2 from the previous half-iteration. Notice that no soft-values for info symbols are available.

The transmission model assumed by Decoder 1 is as follows (cf. Section 3.2): Info word $\boldsymbol{u}^{(1)}$ was encoded by Encoder 1 to code word $\boldsymbol{x}^{(1)}$, so that $\boldsymbol{x}^{(1)} \in \mathcal{C}_1$ and $[\boldsymbol{u}^{(1)}\,\boldsymbol{x}^{(1)}] \in \mathcal{C}_{1,\mathsf{syxt}}$. Only the code word $\boldsymbol{x}^{(1)}$ was transmitted over a BISMC, and the resulting word of channel LLRs is $\boldsymbol{w}^{(2)}_{x^{(1)}}$. As no info word $\boldsymbol{u}^{(1)}$ is transmitted, we set $\boldsymbol{z}_{u^{(1)}} = \mathbf{0}$.

Using the pre-decoding word $\boldsymbol{w}^{(2)}_{x^{(1)}}$ and taking into account the code constraints due to $\mathcal{C}_1$, Decoder 1 computes extrinsic values for its code symbols:

$$\boldsymbol{w}^{(1)}_{x^{(1)}} := \mathbf{dec}^{\mathsf{ext}}_{\mathsf{code}}(\boldsymbol{w}^{(2)}_{x^{(1)}} \,\|\, \mathcal{C}_1). \tag{3.52}$$

The iterative decoding scheme is constituted by (3.49), (3.50), (3.51), and (3.52).

After the last iteration, Decoder 1 computes complete post-decoding values for its info symbols, using the same assumptions as given above:

$$\boldsymbol{v}^{(1)}_{u^{(1)}} := \mathbf{dec}_{\mathsf{info}}(\mathbf{0}, \boldsymbol{w}^{(2)}_{x^{(1)}} \,\|\, \mathcal{C}_{1,\mathsf{syxt}}). \tag{3.53}$$

These values represent the final post-decoding values of the PCC decoder, i.e.,

$$\boldsymbol{v}_u := \boldsymbol{v}^{(1)}_{u^{(1)}}.$$

Notice that for this decoding operation, $\mathcal{C}_{1,\mathsf{syxt}}$ is taken into account.

Similarly to the PCC decoder, *stopping criteria* may be employed for detection of convergence or nonconvergence in order to improve the decoding performance for a given average number of iterations. In general, all stopping criteria for PCC decoders may be applied for SCCs in an appropriately adapted way. An additional stopping criterion is the following [PAT04]: After each (full or half) iteration, the complete post-decoding values corresponding to $\boldsymbol{x}^{(1)}$ are computed and hard decided to $\hat{\boldsymbol{x}}^{(1)}$. The iteration is terminated

as soon as $\hat{\boldsymbol{x}}^{(1)} \in \mathcal{C}_1$. This stopping criterion is similar to that commonly used for LDPC codes (see Section 3.6).

Serially concatenated codes may be constructed such that they operate close to the channel capacity. For example, using repetition codes as outer codes and high rate convolutional codes as inner codes leads to this property [tB00b, HtBH01]. A special class are repeat-accumulate codes: repetition codes are used as outer codes and an accumulator, i.e., a recursive rate-1 memory-1 convolutional encoder, is used as inner code; when slightly modified, even these simple codes may achieve the channel capacity [DJM98, tBK03]. The codes are called systematic repeat-accumulate codes if also the systematic symbols of the repetition codes are transmitted over the communication channel.

EXIT Chart

The EXIT chart for the serially concatenated code depicts the code-symbol EXIT function for Decoder 1 and the info-symbol EXIT function for Decoder 2 (cf. Section 3.3). For the computation of the EXIT functions, the a-priori channels are typically modeled as AWGN channels.

The values of symbol-wise mutual information associated to the inputs and to the outputs of the two constituent decoders are

$$\begin{aligned} I_{\mathsf{ext}}^{(1)} := I(X^{(1)}; W_{x^{(1)}}^{(1)}) &= I(U^{(2)}; W_{u^{(2)}}^{(1)}) =: I_{\mathsf{apri}}^{(2)}, \\ I_{\mathsf{ext}}^{(2)} := I(U^{(2)}; W_{u^{(2)}}^{(2)}) &= I(X^{(1)}; W_{x^{(1)}}^{(2)}) =: I_{\mathsf{apri}}^{(1)}; \end{aligned}$$

the equalities hold because interleaving does not change mutual information.

Regarding these equalities, the EXIT function for Decoder 1 is given as

$$\mathrm{itf}^{(1)} \;:\; I_{\mathsf{ext}}^{(2)} \mapsto I_{\mathsf{ext}}^{(1)}, \tag{3.54}$$

where the pre-decoding information about the info symbols $U^{(1)}$ is assumed to be zero; the EXIT function for Decoder 2 is given as

$$\mathrm{itf}^{(2)} \;:\; I_{\mathsf{ext}}^{(1)} \mapsto I_{\mathsf{ext}}^{(2)}, \tag{3.55}$$

where the mutual information of the communication channel, $I_{\mathsf{ch}} := I(X^{(2)}; Z_{x^{(2)}})$, is used as parameter. The iterative decoder can converge only if the two EXIT functions do not have an intersection. For details about the EXIT chart method for SCCs, we refer the reader to [tB01a].

The EXIT chart method is illustrated for a low-density parity-check code in Example 3.15.

Optimality of the Decoder

Similar to the case of PCCs, the use of LogAPP decoders as constituent decoders is often considered as optimal in literature. Employing LogAPP decoding in (3.49), (3.50), (3.51),

and (3.52), the iterative decoder is given by the following set of equations:

$$\boldsymbol{w}^{(1)}_{u^{(2)}} := \mathrm{perm}_{\pi}\, \boldsymbol{w}^{(1)}_{x^{(1)}}, \tag{3.56a}$$

$$\boldsymbol{w}^{(2)}_{u^{(2)}} := \mathbf{dec}^{\mathsf{ext}}_{\mathsf{info}}(\boldsymbol{w}^{(1)}_{u^{(2)}}, \boldsymbol{z}_{x^{(2)}} \,\|\, \mathcal{C}_{2,\mathsf{syxt}}), \tag{3.56b}$$

$$\boldsymbol{w}^{(2)}_{x^{(1)}} := \mathrm{perm}^{-1}_{\pi}\, \boldsymbol{w}^{(2)}_{u^{(2)}}, \tag{3.56c}$$

$$\boldsymbol{w}^{(1)}_{x^{(1)}} := \mathbf{dec}^{\mathsf{ext}}_{\mathsf{code}}(\boldsymbol{w}^{(2)}_{x^{(1)}} \,\|\, \mathcal{C}_1). \tag{3.56d}$$

The pre-decoding word $\boldsymbol{z}_{x^{(2)}}$ is assumed to consist of conditionally independent LLRs (cf. (3.21) and (3.22))

The considerations on the optimality of the PCC decoder apply in a similar way to the SCC decoder. An additional problem arises from the fact that Decoder 1 decodes to code symbols, (3.56d). Since code symbols depend on each other due to the code structure, so do also the corresponding post-decoding values. Therefore, the values of $\boldsymbol{w}^{(1)}_{x^{(1)}}$ are definitely not conditionally independent from each other. Notice that this basic problem occurs for every soft-output decoder, independently of the applied decoding principle (like LogAPP or MaxLogAPP).

Hence, similar to the PCC case, though LogAPP decoding is optimal with respect to minimum symbol error probability and maximal symbol-wise mutual information (cf. Section 3.4.1) for single decoders, it is not optimal with respect to these criteria when used for constituent decoders of iterative decoders for SCCs.

Code Structure and Parity-Check Matrices

The iterative decoder does not operate on the SCC $\mathcal{C}_{\mathsf{SCC}}$, but on a code that results from extending the original code $\mathcal{C}_{\mathsf{SCC}}$ with the code symbols of the first constituent code. This code is called the embedding code of $\mathcal{C}_{\mathsf{SCC}}$, and it is denoted by $\mathcal{C}_{\mathsf{SCC,emb}}$. Using the generator and the parity-check matrices of the constituent codes, we derive parity-check matrices of $\mathcal{C}_{\mathsf{SCC}}$ and $\mathcal{C}_{\mathsf{SCC,emb}}$. The structures of these parity-check matrices are then related to the structures of the iterative decoders.

For the constituent codes $\mathcal{C}_1$ and $\mathcal{C}_2$, let

$$\boldsymbol{G}_1 \in \mathbb{F}_2^{K \times N_1}, \qquad \boldsymbol{G}_2 \in \mathbb{F}_2^{N_1 \times N_2}$$

denote the generator matrices, and let

$$\boldsymbol{H}_1 \in \mathbb{F}_2^{(N_1-K) \times N_1}, \qquad \boldsymbol{H}_2 \in \mathbb{F}_2^{(N_2-N_1) \times N_2}$$

denote parity-check matrices, respectively. Similarly to (3.12), let inverse generator matrices $\boldsymbol{A}_1^{\mathsf{T}}$ and $\boldsymbol{A}_2^{\mathsf{T}}$ for $\mathcal{C}_1$ and $\mathcal{C}_2$ be defined as

$$\begin{aligned} \boldsymbol{A}_1 \in \mathbb{F}_2^{K \times N_1} \quad &: \quad \boldsymbol{G}_1 \boldsymbol{A}_1^{\mathsf{T}} = \boldsymbol{I}, \\ \boldsymbol{A}_2 \in \mathbb{F}_2^{N_1 \times N_2} \quad &: \quad \boldsymbol{G}_2 \boldsymbol{A}_2^{\mathsf{T}} = \boldsymbol{I}, \end{aligned}$$

respectively.

The embedding code $\mathcal{C}_{\mathsf{SCC,emb}}$ employed for iterative decoding of $\mathcal{C}_{\mathsf{SCC}}$ is formed by the code words of the first constituent code and the SCC words:

$$\boldsymbol{x}_{\mathsf{emb}} := [\boldsymbol{x}^{(1)}\, \boldsymbol{x}].$$

Thus, the generator matrix for $\mathcal{C}_{\mathsf{emb}}$ is given by

$$\boldsymbol{G}_{\mathsf{emb}} := \begin{bmatrix} \boldsymbol{G}_1 & \boldsymbol{G}_1\boldsymbol{P}\boldsymbol{G}_2 \end{bmatrix} \in \mathbb{F}_2^{K\times(N_1+N_2)}. \tag{3.57}$$

Regarding $(\boldsymbol{P}\boldsymbol{G}_2)(\boldsymbol{P}\boldsymbol{A}_2)^{\mathsf{T}} = \boldsymbol{I}$ and (3.12), we may write the parity-check equations for $\mathcal{C}_1$ and $\mathcal{C}_{2,\mathsf{syxt}}$ as

$$\breve{\boldsymbol{x}}^{(1)}\boldsymbol{H}_1^{\mathsf{T}} = \boldsymbol{0}, \qquad\qquad \begin{bmatrix}\breve{\boldsymbol{x}}^{(1)} & \breve{\boldsymbol{x}}\end{bmatrix}\begin{bmatrix}\boldsymbol{I} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_2\end{bmatrix}^{\mathsf{T}} = \boldsymbol{0},$$

or simply as a single equation:

$$\begin{bmatrix}\breve{\boldsymbol{x}}^{(1)} & \breve{\boldsymbol{x}}\end{bmatrix}\begin{bmatrix}\boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_2\end{bmatrix}^{\mathsf{T}} = \boldsymbol{0}. \tag{3.58}$$

It can easily be seen that

$$\boldsymbol{H}_{\mathsf{emb},A} := \begin{bmatrix}\boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_2\end{bmatrix} \in \mathbb{F}_2^{(N_1-K+N_2)\times(N_1+N_2)} \tag{3.59}$$

is a parity-check matrix for $\mathcal{C}_{\mathsf{SCC,emb}}$. Notice that $\boldsymbol{H}_{\mathsf{emb},A}$ contains no redundant rows.

Starting with $\boldsymbol{H}_{\mathsf{emb},A}$, we derive a parity-check matrix for $\mathcal{C}_{\mathsf{SCC}}$. First, we apply the following row operations:

$$\begin{bmatrix}\boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_2\end{bmatrix} \xrightarrow{(a)} \begin{bmatrix}\boldsymbol{I} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_2\end{bmatrix} \xrightarrow{(b)} \begin{bmatrix}\boldsymbol{I} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_1\boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_2\end{bmatrix};$$

(a) the first and the second row are exchanged; (b) the first row is left-multiplied by $\boldsymbol{H}_1$, and then it is added to the second row. The resulting matrix has full rank, and thus the parity-check equation (3.58) can equivalently be written as

$$\begin{bmatrix}\breve{\boldsymbol{x}}^{(1)} & \breve{\boldsymbol{x}}\end{bmatrix}\begin{bmatrix}\boldsymbol{I} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_1\boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_2\end{bmatrix}^{\mathsf{T}} = \boldsymbol{0}.$$

When removing the parts corresponding to code word $\breve{\boldsymbol{x}}^{(1)}$, we obtain

$$\breve{\boldsymbol{x}}\begin{bmatrix}\boldsymbol{H}_1\boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{H}_2\end{bmatrix}^{\mathsf{T}} = \boldsymbol{0}.$$

Thus, we have found a parity-check matrix for $\mathcal{C}_{\mathsf{SCC}}$:

$$\boldsymbol{H}_A := \begin{bmatrix} \boldsymbol{H}_1 \boldsymbol{P} \boldsymbol{A}_2 \\ \boldsymbol{H}_2 \end{bmatrix} \in \mathbb{F}_2^{(N_2-K)\times N_2}. \tag{3.60}$$

Notice that $\boldsymbol{H}_A$ comprises only the parity-check matrices and the inverse generator matrix of the second constituent code, which is indicated by index A.

The structures of the parity-check matrices are directly related to the structure of the iterative decoder. Consider first the parity check matrix for the embedding code, given in (3.59):

$$\boldsymbol{H}_{\mathsf{emb},A} = \left.\begin{bmatrix} \boldsymbol{H}_1 & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{P}\boldsymbol{A}_2 \\ \boldsymbol{0} & \boldsymbol{H}_2 \end{bmatrix}\right\} \begin{array}{l} \longleftarrow \text{ Decoder 1} \\ \\ \longleftarrow \text{ Decoder 2} \end{array}$$

The parity-check constraints defined by the first row are taken into account by Decoder 1, and those defined by the second and the third row are taken into account by Decoder 2. These two sets of parity check constraints are coupled only via the positions corresponding to the code symbols of $\mathcal{C}_1$, namely the first N_1 columns. Similarly, the two constituent decoders are coupled by exchanging information about these code symbols. Notice that the info symbols are not used for the iteration, only for the final decision.

Similar to the case of PCCs, these relations may be exploited to derive an iterative decoder for a given binary linear code. The parity check matrix of the given code has to be "matched" to $\boldsymbol{H}_{\mathsf{emb}}$ of the embedding code or to $\boldsymbol{H}_A$ of the SCC. Again, the problem is to find fitting matrices $\boldsymbol{G}_1$, $\boldsymbol{G}_2$, $\boldsymbol{P}$, where $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ have to show low trellis complexity.

Besides finding iterative decoders for linear binary codes, the derived parity-check matrices for the SCC and its embedding code may also be used for code analysis or code design.

3.6 Low-Density Parity-Check Codes

Low-density parity-check codes (LDPCCs) are defined by parity-check matrices that contain only few ones and thus have "low density". The parity-check constraints may be graphically represented by a graph[28], called factor graph. LDPCCs are iteratively decoded using message passing, also called belief propagation, on this graph. Due to the sparseness of the parity-check matrix, this iterative decoding algorithm is near optimum.

Regular LDPCCs were introduced by Gallager [Gal62, Gal63] and later rediscovered and generalized to irregular LDPCCs in [MN97, Mac99]. The iterative decoding algorithm also goes back to Gallager; it was reinvented and generalized in [WLK95, Wib96] to iterative decoding on graphs. A general framework for this kind of decoding algorithms is given in [Tan81, KFL01, For01, For03, Loe04]. Coding schemes comprising irregular LDPCCs and iterative decoders can achieve channel capacity [RU01a, RSU01, CRU01].

In the sequel, we restrict ourselves to regular LDPCCs and revise the basic encoding and decoding principles. Further information may be found in the literature given above. Whereas PCCs and SCCs are defined by their encoder structures, LDPCCs are defined

[28] The code constraints of any linear code can be represented in a factor graph.

by their parity-check matrices. Accordingly, we start with the parity-check matrices and the factor graphs.

Code Structure, Parity-Check Matrices, and Factor Graph

Consider a parity-check matrix

$$\boldsymbol{H} = \big[\boldsymbol{H}_{m,n}\big]_{\substack{m\in\mathcal{M}\\ n\in\mathcal{N}}} \in \mathbb{F}_2^{M\times N} \tag{3.61}$$

that contains only a small number of ones, i.e., a matrix with "low density"; the column and row index sets of $\boldsymbol{H}$ are denoted by

$$\mathcal{M} := \{0, 1, \ldots, M-1\}, \qquad \mathcal{N} := \{0, 1, \ldots, N-1\}.$$

(Notice that the indices start with 0.) The index set $\mathcal{N}$ is also used for LDPCC words $\boldsymbol{x}$. A regular $(d_\mathsf{v}, d_\mathsf{c})$ LDPCC of length N is defined by a low-density *parity-check matrix* $\boldsymbol{H}$ that has d_v ones in each column and d_c ones in each row[29]:

$$\mathcal{C}_{\mathsf{LDPCC}} := \{\breve{\boldsymbol{x}} \in \mathbb{F}_2^N : \breve{\boldsymbol{x}}\boldsymbol{H}^\mathsf{T} = \boldsymbol{0}\}. \tag{3.62}$$

The value d_v is called *variable node degree*, and the value d_c is called *check node degree*.

The *design rate* R_d is the code rate which can be "expected" from the size of $\boldsymbol{H}$:

$$R_d := \frac{N-M}{N} = 1 - \frac{d_\mathsf{v}}{d_\mathsf{c}}.$$

The design rate follows from $Md_\mathsf{c} = Nd_\mathsf{v}$, which is equal to the number of ones in $\boldsymbol{H}$. As $\boldsymbol{H}$ may contain redundant rows, the actual *code rate* R is lower-bounded by the design rate:

$$R \geq R_d = 1 - \frac{d_\mathsf{v}}{d_\mathsf{c}}.$$

As the code rate must be larger than zero, we have the condition $d_\mathsf{v} < d_\mathsf{c}$ for the variable node degree and the check node degree.

Example 3.12
The parity-check matrix

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

defines an LDPCC with $d_\mathsf{v} = 2$ and $d_\mathsf{c} = 3$. (Due to the small size, matrix $\boldsymbol{H}$ shows no low-density, of course.) The design rate is $R_d = 1 - 2/3 = 1/3$. As only three rows of $\boldsymbol{H}$ are linearly independent (the last row is the sum of the first three rows), the actual code rate is $R = 1/2$. ◇

[29] In parity-check matrices of irregular LDPCCs, the number of ones per row and the number of ones per column is not constant.

The parity-check equation

$$\breve{\boldsymbol{x}}\boldsymbol{H}^{\mathsf{T}} = \boldsymbol{0}$$

can be represented in a bipartite graph of variable nodes, check nodes, and edges between variable nodes and check nodes. This graph is called *factor graph* [KFL01, Loe04], and it is addressed in the sequel.

To start with, we write the rows of the parity-check matrix $\boldsymbol{H}$ as

$$\boldsymbol{H}_m := \left[H_{m,0}, H_{m,1}, \ldots, H_{m,N-1}\right],$$

$m \in \mathcal{M}$, such that

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_0 \\ \vdots \\ \boldsymbol{H}_{M-1\cdot} \end{bmatrix}$$

The parity-check equation corresponding to the m-th row,

$$\breve{\boldsymbol{x}}\,\boldsymbol{H}_m^{\mathsf{T}} = 0,$$

is called check equation m. For each code symbol x_n, we define a *variable node* n, $n \in \mathcal{N}$, and for each check equation m, we define a *check node* m, $m \in \mathcal{M}$. Hence, the set of variable nodes and the set of check nodes can be identified with $\mathcal{N}$ and $\mathcal{M}$, respectively. (Due to the one-to-one correspondence, we do not distinguish between nodes and indices.)

A variable node n and a check node m are connected by an *edge* (m, n) if code symbol x_n participates in check equation m. As this is the case if the parity-check matrix $\boldsymbol{H}$ has a one in the corresponding position, i.e., if $H_{m,n} = 1$, the set of edges is given by

$$\mathcal{E} := \{(m, n) \in \mathcal{M} \times \mathcal{N} : H_{m,n} = 1\}.$$

The *factor graph* of $\boldsymbol{H}$ is the graph defined by the triplet $(\mathcal{M}, \mathcal{N}, \mathcal{E})$, and it represents graphically the parity check equations, and thus the code constraints[30]. More information about factor graphs, including a formal and more general definition, can be found in [Tan81, WLK95, KFL01, Loe04].

Example 3.13
We continue Example 3.12. The parity-check equation $\breve{\boldsymbol{x}}\boldsymbol{H}^{\mathsf{T}} = \boldsymbol{0}$ can be written as

$$\begin{aligned}
\breve{x}_0 \oplus \breve{x}_1 \oplus \breve{x}_3 &= 0 \quad (m = 0) \\
\breve{x}_0 \oplus \breve{x}_2 \oplus \breve{x}_4 &= 0 \quad (m = 1) \\
\breve{x}_1 \oplus \breve{x}_2 \oplus \breve{x}_5 &= 0 \quad (m = 2) \\
\breve{x}_3 \oplus \breve{x}_4 \oplus \breve{x}_5 &= 0 \quad (m = 3).
\end{aligned}$$

The number of each check equation is given in parentheses. Notice that $d_{\mathsf{c}} = 3$ code symbols participate in each parity-check equation, and that each code symbol participates in $d_{\mathsf{v}} = 2$ parity-check equations. The factor graph of $\boldsymbol{H}$, representing this system of equations, is given in Fig. 3.8. ◇

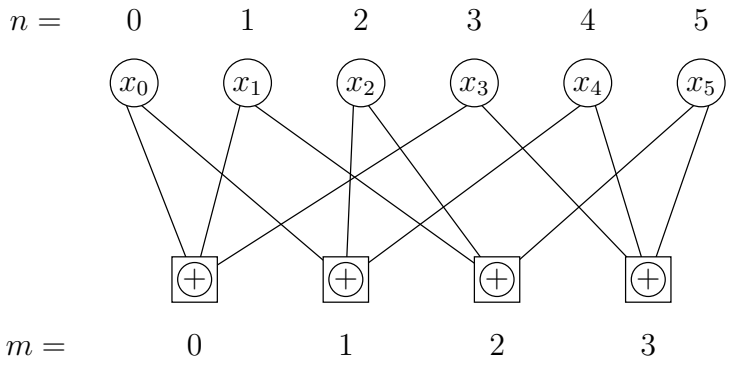


Figure 3.8: Factor graph for the LDPCC from Example 3.12 and Example 3.13.

The iterative decoder for the LDPCC $\mathcal{C}_{\mathsf{LDPCC}}$ operates on a code that results from extending the original code $\mathcal{C}_{\mathsf{LDPCC}}$; this is similar to decoding of PCCs and SCCs (see previous section). This code is called embedding code of $\mathcal{C}_{\mathsf{LDPCC}}$, and it is denoted by $\mathcal{C}_{\mathsf{LDPCC,emb}}$. The embedding code is defined via code constraints of repetition codes and of single parity-check codes, and it is addressed in the sequel.

For each code symbol x_n participating in check equation m, $m \in \mathcal{M}$, we define a replica $x'_{m,n} \in \mathbb{B}$ representing a code symbol of the embedding code. Notice that symbol $x'_{m,n}$ corresponds to edge (m,n) of the factor graph and thus, to the one in $\boldsymbol{H}$ at position (m,n). As the factor graph has $N' := |\mathcal{E}|$ edges, the number of additional (new) symbols $x'_{m,n}$ is also N'. Due to the definition, the code symbols $x'_{m,n}$ are required to fulfill two kinds of constraints.

The first kind of constraints is related to the variable nodes of the factor graph and can be described using repetition codes. For each variable node n, $n \in \mathcal{N}$, we define the set of check nodes connected to that variable node as

$$\mathcal{M}_n := \{m \in \mathcal{M} : H_{m,n} = 1\}.$$

Notice that $\mathcal{M}_n \subset \mathcal{M}$ and $|\mathcal{M}_n| = d_{\mathsf{v}}$. Furthermore, we associate a code word

$$\boldsymbol{x}^{(\mathsf{v},n)} := [x_0^{(\mathsf{v},n)}, \ldots, x_{d_{\mathsf{v}}-1}^{(\mathsf{v},n)}] := \left[x'_{m,n}\right]_{m \in \mathcal{M}_n} \tag{3.63}$$

with each variable node n, comprising all symbols $x'_{m,n}$ associated with edges of this variable node. (The index v indicates the association with a variable node.) As those symbols are equal to x_n by definition, we can express the given constraints as

$$\boldsymbol{x}^{(\mathsf{v},n)} \in \mathcal{R}_{d_{\mathsf{v}}}, \qquad [x_n\, \boldsymbol{x}^{(\mathsf{v},n)}] \in \mathcal{R}_{d_{\mathsf{v}},\mathsf{syxt}}, \tag{3.64}$$

where $\mathcal{R}_{d_{\mathsf{v}}}$ denotes the repetition code of length d_{v}, and $\mathcal{R}_{d_{\mathsf{v}},\mathsf{syxt}}$ denotes its systematically extended code. Writing the concatenation of all variable-node code words as

$$\boldsymbol{x}^{(\mathsf{v})} := [\boldsymbol{x}^{(\mathsf{v},0)} \ \ldots \ \boldsymbol{x}^{(\mathsf{v},N-1)}], \tag{3.65}$$

[30] In literature, the factor graph is often associated with the code defined by a parity-check matrix. In fact, it is associated with a particular parity-check matrix of the code.

we can express the code word of the embedding code as

$$\boldsymbol{x}_{\mathsf{emb}} := [\boldsymbol{x}\, \boldsymbol{x}^{(\mathsf{v})}]. \tag{3.66}$$

Combining (3.64) and (3.65), the constraints related to the variable nodes are given by

$$\boldsymbol{x}^{(\mathsf{v})} \in \mathcal{R}_{d_\mathsf{v}}^N, \qquad [\boldsymbol{x}\, \boldsymbol{x}^{(\mathsf{v})}] \in (\mathcal{R}_{d_\mathsf{v}}^N)_{\mathsf{syxt}}; \tag{3.67}$$

$(\mathcal{R}_{d_\mathsf{v}}^N)_{\mathsf{syxt}}$ denotes the systematically extended code of $\mathcal{R}_{d_\mathsf{v}}^N$, where the systematic symbols are gathered in the first positions of the code word.

The second kind of constraints are related to the check nodes of the factor graph and can be described using single parity check codes. For each check node m, $m \in \mathcal{M}$, we define the set of variable nodes connected to that check node as

$$\mathcal{N}_m := \{n \in \mathcal{N} : H_{m,n} = 1\}.$$

Notice that $\mathcal{N}_m \subset \mathcal{N}$ and $|\mathcal{N}_m| = d_\mathsf{c}$. Furthermore, we associate a code word

$$\boldsymbol{x}^{(\mathsf{c},m)} := [x_0^{(\mathsf{c},m)}, \ldots, x_{d_\mathsf{c}-1}^{(\mathsf{c},m)}] := \left[x'_{m,n}\right]_{n\in\mathcal{N}_m} \tag{3.68}$$

with each check node m, comprising all symbols $x'_{m,n}$ associated with edges of this check node. (The index c indicates the association with a check node.) As those symbols fulfill a parity check equation by definition, we can express the given constraints as

$$\boldsymbol{x}^{(\mathsf{c},m)} \in \mathcal{S}_{d_\mathsf{c}}, \tag{3.69}$$

where $\mathcal{S}_{d_\mathsf{c}}$ denotes the single parity check code of length d_c. The concatenation of all check-node code words is written as

$$\boldsymbol{x}^{(\mathsf{c})} := [\boldsymbol{x}^{(\mathsf{c},0)} \ \ldots\ \boldsymbol{x}^{(\mathsf{c},M-1)}]. \tag{3.70}$$

Combining (3.69) and (3.70), the constraints related to the check nodes are given by

$$\boldsymbol{x}^{(\mathsf{c})} \in \mathcal{S}_{d_\mathsf{c}}^M. \tag{3.71}$$

The code words $\boldsymbol{x}^{(\mathsf{v})}$ and $\boldsymbol{x}^{(\mathsf{c})}$ contain the same symbols, only in a different ordering. For conversion, we define[31] a permutation function π such that

$$\boldsymbol{x}^{(\mathsf{c})} = \mathrm{perm}_\pi\, \boldsymbol{x}^{(\mathsf{v})}. \tag{3.72}$$

Thus, $\boldsymbol{x}^{(\mathsf{c})}$ may be interpreted as an interleaved version of $\boldsymbol{x}^{(\mathsf{v})}$.

Summarizing the above considerations, we define the *embedding code* of $\mathcal{C}_{\mathsf{LDPCC}}$ as

$$\begin{aligned}\mathcal{C}_{\mathsf{emb}} := \big\{[\boldsymbol{x}\, \boldsymbol{x}^{(\mathsf{v})}] \in \mathbb{B}^{N+N'} : \boldsymbol{x}^{(\mathsf{c})} = \mathrm{perm}_\pi\, \boldsymbol{x}^{(\mathsf{v})};\\ [\boldsymbol{x}\, \boldsymbol{x}^{(\mathsf{v})}] \in (\mathcal{R}_{d_\mathsf{v}}^N)_{\mathsf{syxt}}, \boldsymbol{x}^{(\mathsf{c})} \in \mathcal{S}_{d_\mathsf{c}}^M\big\}.\end{aligned} \tag{3.73}$$

The repetition code of length d_v, $\mathcal{R}_{d_\mathsf{v}}$, and the single parity check code of length d_c, $\mathcal{S}_{d_\mathsf{c}}$, are called constituent codes. It can easily be seen that

$$\boldsymbol{x} \in \mathcal{C}_{\mathsf{LDPCC}} \quad \Leftrightarrow \quad [\boldsymbol{x}\, \boldsymbol{x}^{(\mathsf{v})}] \in \mathcal{C}_{\mathsf{emb}}.$$

This relation is exploited for decoding.

[31] For notation, see Appendix B.

Example 3.14
We continue Example 3.13. As each element $H_{m,n}$ corresponds to one code symbol $x'_{m,n}$ of the embedding code, these symbols may be determined, symbolically, by replacing the ones in $\boldsymbol{H}$:

$$\begin{bmatrix} x'_{0,0} & x'_{0,1} & & x'_{0,3} & & \\ x'_{1,0} & & x'_{1,2} & & x'_{1,4} & \\ & x'_{2,1} & x'_{2,2} & & & x'_{2,5} \\ & & & x'_{3,3} & x'_{3,4} & x'_{3,5} \end{bmatrix}.$$

The code words associated with variable nodes (corresponding to the "columns" in the matrix given above) are

$$\boldsymbol{x}^{(\mathsf{v},0)} = [x'_{0,0}, x'_{1,0}], \qquad \boldsymbol{x}^{(\mathsf{v},1)} = [x'_{0,1}, x'_{2,1}], \qquad \boldsymbol{x}^{(\mathsf{v},2)} = [x'_{1,2}, x'_{2,2}],$$
$$\boldsymbol{x}^{(\mathsf{v},3)} = [x'_{0,3}, x'_{3,3}], \qquad \boldsymbol{x}^{(\mathsf{v},4)} = [x'_{1,4}, x'_{3,4}], \qquad \boldsymbol{x}^{(\mathsf{v},5)} = [x'_{2,5}, x'_{3,5}],$$

and the code words associated with check nodes (corresponding to the "rows" in the matrix given above) are

$$\boldsymbol{x}^{(\mathsf{c},0)} = [x'_{0,0}, x'_{0,1}, x'_{0,3}], \qquad \boldsymbol{x}^{(\mathsf{c},1)} = [x'_{1,0}, x'_{1,2}, x'_{1,4}],$$
$$\boldsymbol{x}^{(\mathsf{c},2)} = [x'_{2,1}, x'_{2,2}, x'_{2,5}], \qquad \boldsymbol{x}^{(\mathsf{c},3)} = [x'_{3,3}, x'_{3,4}, x'_{3,5}].$$

The constituent codes of the embedding code $\mathcal{C}_{\mathsf{emb}}$ are the repetition code of length $d_{\mathsf{v}} = 2$, $\mathcal{R}_2$, and the single parity check code of length $d_{\mathsf{c}} = 3$, $\mathcal{S}_3$. The code constraints of the embedding code are given by

$$[x_0\, \boldsymbol{x}^{(\mathsf{v},0)}], [x_1\, \boldsymbol{x}^{(\mathsf{v},1)}], [x_2\, \boldsymbol{x}^{(\mathsf{v},2)}], [x_3\, \boldsymbol{x}^{(\mathsf{v},3)}], [x_4\, \boldsymbol{x}^{(\mathsf{v},4)}], [x_5\, \boldsymbol{x}^{(\mathsf{v},5)}] \in \mathcal{R}_{2,\mathsf{syxt}}$$

(each "column" preceeded by the corresponding symbol x_n, regarded as info symbol, is a code word of the systematically extended repetition code) and

$$\boldsymbol{x}^{(\mathsf{c},0)}, \boldsymbol{x}^{(\mathsf{c},1)}, \boldsymbol{x}^{(\mathsf{c},2)}, \boldsymbol{x}^{(\mathsf{c},3)} \in \mathcal{S}_3$$

(each "row" is a code word of the single parity check code).

The code constraints are depicted in Fig. 3.9. ◇

Encoder

LDPCCs are typically, but not necessarily, systematically encoded. For finding a generator matrix, the parity-check matrix $\boldsymbol{H}$ may be converted into the form $[\boldsymbol{B}\, \boldsymbol{I}]$ by applying row operations and (if necessary) column permutations; $\boldsymbol{I}$ denotes the identity matrix, and $\boldsymbol{B}$ is any matrix over $\mathbb{F}_2$. A systematic generator matrix of the code (or of an equivalent code if columns were permuted) is then given by

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{B}^{\mathsf{T}} \end{bmatrix} \in \mathbb{F}_2^{K \times N},$$

such that the encoding can be written as

$$\breve{\boldsymbol{x}} = \breve{\boldsymbol{u}}\boldsymbol{G}.$$

We refer to the encoder as LDPCC encoder.

Since the parity-check matrix generally has no special structure, the encoding complexity is not linear in the code length. Further information about encoding of LDPCCs can be found in [RU01b].

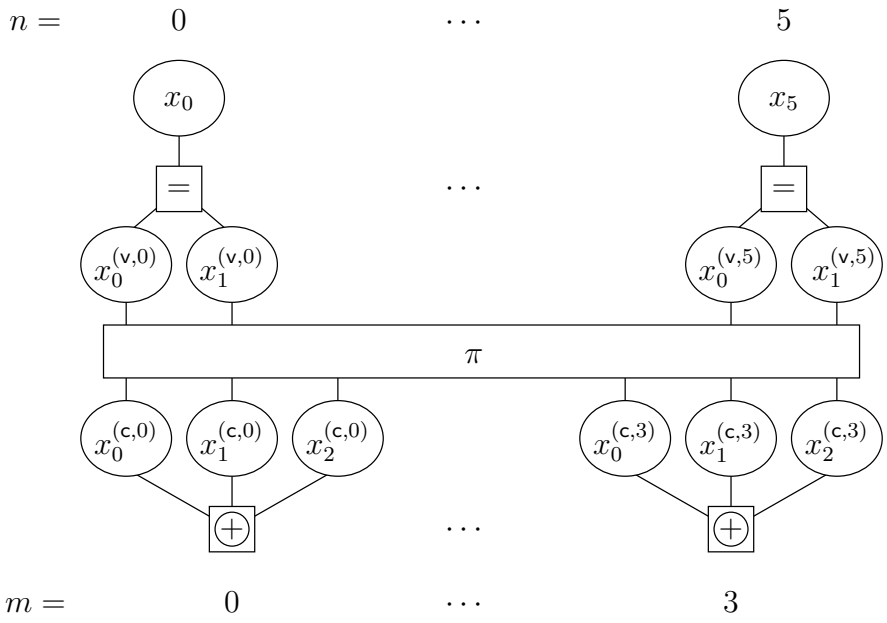


Figure 3.9: Graphical representation of the embedding code from Example 3.14 corresponding to the LDPCC. (The factor graph of the original LDPCC is depicted in Fig. 3.8).

Iterative Decoder

Iterative decoding of LDPCCs is usually described by belief propagation (also called message passing) on the factor graph. In the sequel, the iterative decoding algorithm is presented from a slightly different point of view, based on the embedding code defined above. Although the actual decoding operations are the same, the relation to iterative decoding of PCCs and SCCs becomes clearer.

The iterative decoder for an LDPCC is depicted in Fig. 3.10. It comprises two decoders (DEC1 and DEC2), an interleaver (π), a deinterleaver (π^{-1}), and an inverse LDPCC encoder (ENC^{-1}). The two decoders, referred to as Decoder 1 and Decoder 2, correspond to the two types of (virtual) codes, namely the repetition codes and the single parity check codes, and they are called constituent decoders. Commonly, extrinsic LogAPP or extrinsic MaxLogAPP decoders are employed as constituent decoders. Using LogAPP decoders corresponds to the sum-prod algorithm, and using MaxLogAPP decoders corresponds to the min-sum algorithm. The overall decoder is called LDPCC decoder. Notice the reverse similarity of the decoder to the SCC decoder.

The channel LLR word $\boldsymbol{z}_x$ corresponding to the LDPCC word $\boldsymbol{x}$ is given to Decoder 1. For iterative decoding, Decoder 1 and Decoder 2 exchange extrinsic values $\boldsymbol{w}^{(1)}_{x^{(\mathsf{c})}} = \mathrm{perm}_\pi\, \boldsymbol{w}^{(1)}_{x^{(\mathsf{v})}}$ and $\boldsymbol{w}^{(2)}_{x^{(\mathsf{v})}} = \mathrm{perm}^{-1}_\pi\, \boldsymbol{w}^{(2)}_{x^{(\mathsf{c})}}$ corresponding to the code words $\boldsymbol{x}^{(\mathsf{c})} = \mathrm{perm}_\pi\, \boldsymbol{x}^{(\mathsf{v})}$. Decoder 1 operates on code word $\boldsymbol{x}^{(\mathsf{v})}$ and takes into account only the constraints due to the repetition codes; Decoder 2 operates on code word $\boldsymbol{x}^{(\mathsf{c})}$ and takes into account only the constraints due to the single parity check codes. After the last iteration, Decoder 1 computes the word of complete post-decoding values $\boldsymbol{v}_x$, corresponding to the LDPCC word $\boldsymbol{x}$. A hard-decision gives the estimated LDPCC word $\hat{\boldsymbol{x}}$.

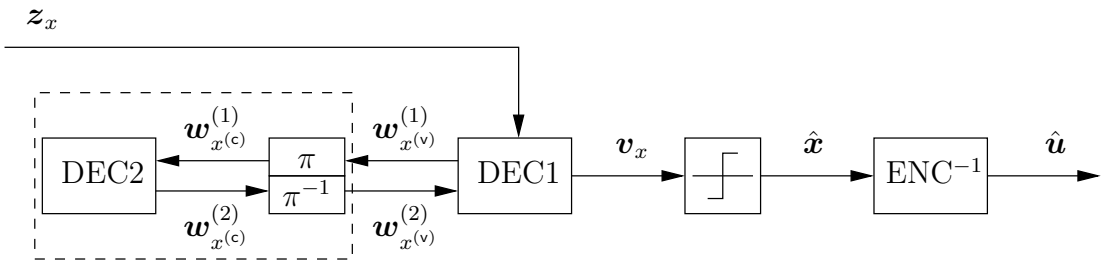


Figure 3.10: LDPCC decoder.

By inverting the LDPCC encoding, the estimated info word $\hat{\boldsymbol{u}}$ is obtained. Notice the notation for soft-values: the subindex indicates which info or code word the soft-values correspond to, and the superindex indicates by which decoder the soft-values were computed. The decoding process is discussed in more detail in the sequel.

Each iteration of the *decoding process* consists of two steps, called half-iterations. Decoder 1 operates in the first half-iteration, and Decoder 2 operations in the second half-iteration.

Decoder 1: The soft-values available to Decoder 1 are $\boldsymbol{z}_x$ and

$$\boldsymbol{w}^{(2)}_{x^{(\mathsf{v})}} := \mathrm{perm}_{\pi}^{-1}\, \boldsymbol{w}^{(2)}_{x^{(\mathsf{c})}} \tag{3.74}$$

In the first iteration, $\boldsymbol{w}^{(2)}_{x^{(\mathsf{c})}}$ is set to the all-zero word, and in the other iterations, it is the decoding result of Decoder 2 from the previous half-iteration. Following (3.65), we use the division

$$\boldsymbol{w}^{(2)}_{x^{(\mathsf{v})}} = [\boldsymbol{w}^{(2)}_{x^{(\mathsf{v},0)}} \ \ldots \ \boldsymbol{w}^{(2)}_{x^{(\mathsf{v},N-1)}}].$$

The transmission model assumed by Decoder 1 is as follows for all $n \in \mathcal{N}$: Symbol x_n (of $\boldsymbol{x}$) was encoded by a repetition encoder to code word $\boldsymbol{x}^{(\mathsf{v},n)}$, so that $[x_n\, \boldsymbol{x}^{(\mathsf{v},n)}] \in \mathcal{R}_{d_\mathsf{v},\mathsf{syxt}}$. The symbol x_n and the word $\boldsymbol{x}^{(\mathsf{v},n)}$ were transmitted over BISMCs, and the resulting words of channel LLRs are $z_{x,n}$ and $\boldsymbol{w}^{(2)}_{x^{(\mathsf{v},n)}}$, respectively.

Using the pre-decoding words $z_{x,n}$ and $\boldsymbol{w}^{(2)}_{x^{(\mathsf{v},n)}}$ and taking into account the code constraints due to $\mathcal{R}_{d_\mathsf{v},\mathsf{syxt}}$, Decoder 1 computes extrinsic values for its code symbols:

$$\boldsymbol{w}^{(1)}_{x^{(\mathsf{v},n)}} := \mathbf{dec}^{\mathsf{ext}}_{\mathsf{code}}(z_{x,n}, \boldsymbol{w}^{(2)}_{x^{(\mathsf{v},n)}} \,\|\, \mathcal{R}_{d_\mathsf{v},\mathsf{syxt}}), \tag{3.75}$$

for all $n \in \mathcal{N}$. Notice that the N decoding operations are independent. Following (3.65), the overall output word is formed as

$$\boldsymbol{w}^{(1)}_{x^{(\mathsf{v})}} = [\boldsymbol{w}^{(1)}_{x^{(\mathsf{v},0)}} \ \ldots \ \boldsymbol{w}^{(1)}_{x^{(\mathsf{v},N-1)}}].$$

Decoder 2: The soft-values available to Decoder 2 are

$$\boldsymbol{w}^{(1)}_{x^{(\mathsf{c})}} := \mathrm{perm}_{\pi}\, \boldsymbol{w}^{(1)}_{x^{(\mathsf{v})}}. \tag{3.76}$$

The word $\boldsymbol{w}^{(1)}_{x^{(\mathsf{v})}}$ is the decoding result of Decoder 1 from the previous half-iteration. Notice that no soft-values for (local) info symbols[32] are available. Following (3.70), we use the division

$$\boldsymbol{w}^{(1)}_{x^{(\mathsf{c})}} = [\boldsymbol{w}^{(1)}_{x^{(\mathsf{c},0)}} \ \ldots \ \boldsymbol{w}^{(1)}_{x^{(\mathsf{c},M-1)}}].$$

The transmission model assumed by Decoder 2 is as follows for all $m \in \mathcal{M}$: A (virtual) info word $\boldsymbol{u}^{(\mathsf{c},m)}$ was encoded by a single parity check encoder to code word $\boldsymbol{x}^{(\mathsf{c},m)}$, so that $[\boldsymbol{u}^{(\mathsf{c},m)}\, \boldsymbol{x}^{(\mathsf{c},m)}] \in \mathcal{S}_{d_\mathsf{c},\mathsf{syxt}}$. Only the code word $\boldsymbol{x}^{(\mathsf{c},m)}$ was transmitted over an BISMC, and the resulting word of channel LLRs is $\boldsymbol{w}^{(1)}_{x^{(\mathsf{c},m)}}$. As the info word $\boldsymbol{u}^{(\mathsf{c},m)}$ was not transmitted, we set the corresponding word of channel LLRs to $\boldsymbol{z}_{u^{(\mathsf{c},m)}} = \mathbf{0}$.

Using the pre-decoding word $\boldsymbol{w}^{(1)}_{x^{(\mathsf{c},m)}}$ and taking into account the code constraints due to $\mathcal{S}_{d_\mathsf{c}}$, Decoder 2 computes extrinsic values for its code symbols:

$$\boldsymbol{w}^{(2)}_{x^{(\mathsf{c},m)}} := \mathbf{dec}^{\mathsf{ext}}_{\mathsf{code}}(\boldsymbol{w}^{(1)}_{x^{(\mathsf{c},m)}} \,\|\, \mathcal{S}_{d_\mathsf{c}}), \tag{3.77}$$

for all $m \in \mathcal{M}$. Notice that the M decoding operations are independent. Following (3.70), the output word is formed as

$$\boldsymbol{w}^{(2)}_{x^{(\mathsf{c})}} = [\boldsymbol{w}^{(2)}_{x^{(\mathsf{c},0)}} \ \ldots \ \boldsymbol{w}^{(2)}_{x^{(\mathsf{c},M-1)}}].$$

The iterative decoding scheme is constituted by (3.74), (3.75), (3.76), and (3.77).

After each iteration, Decoder 1 computes complete post-decoding values for its (local) info symbols x_n, using the same assumptions as given above:

$$v_{x,n} := \mathrm{dec}_{\mathsf{info}}(z_{x,n}, \boldsymbol{w}^{(2)}_{x^{(\mathsf{v},n)}} \,\|\, \mathcal{R}_{d_\mathsf{v},\mathsf{syxt}}), \tag{3.78}$$

for all $n \in \mathcal{N}$. Notice again that the N decoding operations are independent. The complete post-decoding word is then formed as

$$\boldsymbol{v}_x := [v_{x,0}, v_{x,1}, \ldots, v_{x,N-1}]$$

and hard decided to the estimated code word $\hat{\boldsymbol{x}}$. If $\hat{\boldsymbol{x}} \in \mathcal{C}_{\mathsf{LDPCC}}$, the iteration is terminated, and the estimated info word $\hat{\boldsymbol{u}}$ corresponding to $\hat{\boldsymbol{x}}$ is determined. This *stopping criterion* makes obvious that iterative decoding aims at finding the most likely code word, and thus performs sequences (word) estimation rather than symbol-by-symbol estimation.

The decoding operations are particularly simple due to the simple constituent codes, namely repetition codes and single parity check codes (cf. examples in Section 3.4): When employing *LogAPP decoding*, (3.75) and (3.77) result as[33]

$$w^{(1)}_{x^{(\mathsf{v},n)},i} = \mathrm{logapp}^{\mathsf{ext}}_{\mathsf{code},i}(z_{x,n}, \boldsymbol{w}^{(2)}_{x^{(\mathsf{v},n)}} \,\|\, \mathcal{R}_{d_\mathsf{v},\mathsf{syxt}}) = z_{x,n} + \sum_{\substack{l=0 \\ l\neq i}}^{d_\mathsf{v}-1} w^{(2)}_{x^{(\mathsf{v},n)},l}\,,$$

$$w^{(2)}_{x^{(\mathsf{c},m)},i} = \mathrm{logapp}^{\mathsf{ext}}_{\mathsf{code},i}(\boldsymbol{w}^{(1)}_{x^{(\mathsf{c},m)}} \,\|\, \mathcal{S}_{d_\mathsf{c}}) = \boxplus_{\substack{l=0 \\ l\neq i}}^{d_\mathsf{c}-1} w^{(1)}_{x^{(\mathsf{c},m)},l}\,.$$

[32] Info symbols of the constituent code $\mathcal{S}_{d_\mathsf{c}}$.

[33] The operator $\boxplus$ is defined in Appendix C.2.

This is equivalent to the sum-prod algorithm. When employing *MaxLogAPP decoding*, (3.75) and (3.77) result as[34]

$$w^{(1)}_{x^{(\mathsf{v},n)},i} = \mathrm{maxlogapp}^{\mathsf{ext}}_{\mathsf{code},i}(z_{x,n}, \boldsymbol{w}^{(2)}_{x^{(\mathsf{v},n)}} \,\|\, \mathcal{R}_{d_\mathsf{v},\mathsf{syxt}}) = z_{x,n} + \sum_{\substack{l=0 \\ l\neq i}}^{d_\mathsf{v}-1} w^{(2)}_{x^{(\mathsf{v},n)},l} \,,$$

$$w^{(2)}_{x^{(\mathsf{c},m)},i} = \mathrm{maxlogapp}^{\mathsf{ext}}_{\mathsf{code},i}(\boldsymbol{w}^{(1)}_{x^{(\mathsf{c},m)}} \,\|\, \mathcal{S}_{d_\mathsf{c}}) = \boxplus_{\substack{l=0 \\ l\neq i}}^{d_\mathsf{c}-1} w^{(1)}_{x^{(\mathsf{c},m)},l} \,.$$

This is equivalent to the min-sum algorithm. For both LogAPP decoding and MaxLogAPP decoding, (3.78) results as

$$v_{x,n} = \mathrm{maxlogapp}_{\mathsf{info}}(z_{x,n}, \boldsymbol{w}^{(2)}_{x^{(\mathsf{v},n)}} \,\|\, \mathcal{R}_{d_\mathsf{v},\mathsf{syxt}}) = z_{x,n} + \sum_{l=0}^{d_\mathsf{v}-1} w^{(2)}_{x^{(\mathsf{v},n)},l} \,.$$

EXIT Chart

The EXIT chart for the LDPC code depicts the code-symbol EXIT functions for the two constituent decoders (cf. Section 3.3). For the computation of the EXIT functions, the a-priori channels are typically modeled as AWGN channels.

The values of symbol-wise mutual information associated to the inputs and to the outputs of the two constituent decoders are

$$\begin{aligned} I^{(\mathsf{v})}_{\mathsf{ext}} := I(X^{(\mathsf{v})}; W^{(1)}_{x^{(\mathsf{v})}}) &= I(X^{(\mathsf{c})}; W^{(1)}_{x^{(\mathsf{c})}}) =: I^{(\mathsf{c})}_{\mathsf{apri}}, \\ I^{(\mathsf{c})}_{\mathsf{ext}} := I(X^{(\mathsf{c})}; W^{(2)}_{x^{(\mathsf{c})}}) &= I(X^{(\mathsf{v})}; W^{(2)}_{x^{(\mathsf{v})}}) =: I^{(\mathsf{v})}_{\mathsf{apri}}; \end{aligned}$$

the equalities hold because interleaving does not change mutual information.

Regarding these equalities, the EXIT function for Decoder 1 (variable-node decoder) is given as

$$\mathrm{itf}^{(\mathsf{v})} : I^{(\mathsf{c})}_{\mathsf{ext}} \mapsto I^{(\mathsf{v})}_{\mathsf{ext}}, \tag{3.79}$$

where the mutual information of the communication channel, $I_{\mathsf{ch}} := I(X; Z_X)$, is used as parameter; the EXIT function for Decoder 2 (check-node decoder) is given as

$$\mathrm{itf}^{(\mathsf{c})} : I^{(\mathsf{v})}_{\mathsf{ext}} \mapsto I^{(\mathsf{c})}_{\mathsf{ext}}. \tag{3.80}$$

The iterative decoder can converge only if the two EXIT functions do not have an intersection.

For details about the EXIT chart method for LDPCCs, we refer the reader to [AKtB04].

[34] The operator $\boxplus$ is defined in Appendix C.2.

Example 3.15
Consider an LDPCC with variable node degree $d_{\mathsf{v}} = 3$ and check node degree $d_{\mathsf{c}} = 4$, and the transmission over a BEC. The EXIT charts for two values of channel information, $I_{\mathsf{ch}} = 0.36$ and $I_{\mathsf{ch}} = 0.45$, are depicted in Fig. 3.11. In addition to the EXIT functions, the figures show the decoding trajectories, which illustrate the evolution of the extrinsic mutual information during iterative decoding. The decoding trajectory gets stuck for the lower value of channel information, and it approaches the upper right corner for the larger value of channel information. The latter case corresponds to error-free decoding. The decoding threshold is between $I_{\mathsf{ch}} = 0.36$ and $I_{\mathsf{ch}} = 0.45$, and it corresponds to the case where the two EXIT functions touch each other such that the decoding trajectory can slip through. ◇

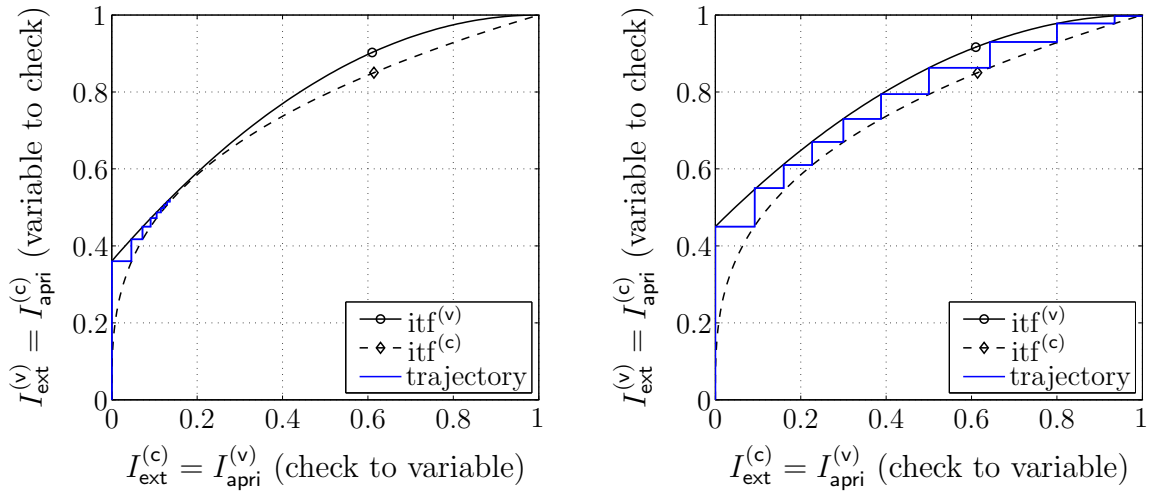


Figure 3.11: EXIT charts, Example 3.15. For each $I_{\mathsf{ch}} = 0.36$ (left figure) and $I_{\mathsf{ch}} = 0.45$ (right figure), the EXIT function for the variable-node decoder itf$^{(\mathsf{v})}$, the EXIT function for the check-node decoder itf$^{(\mathsf{c})}$, and a decoding trajectory are depicted.

Optimality of the Decoder

The considerations on the optimality of the PCC and the SCC decoder apply to the LDPCC decoder in a similar way. As opposed to the SCC decoder, both constituent decoders, namely the one for the repetition codes and the one for the single parity check codes, decode to "local" code symbols. Thus, the computed extrinsic values are never independent, in contradiction to the assumptions. Due to the low density of the parity-check matrix, the pre-decoding values remain independent for the first few iterations, but after a certain number of iterations, they do depend on each other (otherwise, not all code constraints would have been taken into account).

Hence, using LogAPP decoders for the constituent codes may be optimal for each individual decoding operation, but it is not clear if it is also optimal for the iterative decoding process.

3.7 Summary

This chapter began with some properties of linear binary codes and have defined the corresponding systematically extended codes. Based on these, a general decoding model has been introduced. As a basic property, the superchannels from encoder input to decoder output are symmetric if linear channel encoders, symmetric channels, and symmetric symbol-by-symbol soft-output decoders are employed. LogAPP decoding and MaxLogAPP decoding have been given as examples for symmetric soft-output decoding principles.

Three iteratively decodable channel codes have been discussed: parallel concatenated codes, serially concatenated codes, and low-density parity-check codes. For each type of code, the encoder, the iterative decoder, and code structures in terms of parity-check matrices have been considered. The iterative decoders are based on constituent decoders, each of which takes into account only subsets of the overall code constraints. During iterative decoding, the constituent decoders exchange information about symbols: In the case of PCCs, information about info symbols; in the case of SCCs, information about symbols of the outer code; and in the case of LDPCCs, information about symbols of the embedding code. These are three rather different principles of information exchange between constituent decoders.

Information exchange between constituent decoders would not be possible if an iterative decoder operated directly on the actual code. For PCCs, SCCs, and LDPCCs, the notion of embedding codes have been introduced. The constituent decoders have been shown to exchange information via the symbols of the embedding code that are not symbols of the original code. For PCCs and SCCs, the parity-check matrices have been determined, and their structures have been related to the decoding schemes. For both PCCs and SCCs, only the concatenation of two constituent codes has been considered; but the concept may easily be extended in a straight-forward way. The derived structures of parity-check matrices for PCCs and SCCs may give hints as to how to find embedding codes and the corresponding iterative decoders for other linear binary codes.

Chapter 4

Reliability Information

A symbol-by-symbol soft-output decoder computes post-decoding soft-values for each info or code symbol. Thus, it provides not only estimates of the symbols, but also information about the reliability of these estimates; this is called reliability information. This reliability information may be accurate, as in the case of a LogAPP decoder, but it may also be mismatched, as in the case of a MaxLogAPP decoder (cf. Section 3.4).

Several aspects of reliability information are discussed in this chapter. We start with a definition of reliability values associated to soft-values (cf. Chapter 2). Then, a general framework is presented for assessing the quality of soft-values and for improving their quality by means of memoryless post-processing. The average Kullback-Leibler distance is proposed as a criterion suitable for measuring the difference between the magnitudes of the soft-values, corresponding to the interpreted reliability, and the reliability values, corresponding to the true reliabilities. As an application, decoding of parallel concatenated codes (PCC) is considered. The performance is improved by improving the extrinsic soft-values of the LogAPP or MaxLogAPP constituent decoders.

The new contributions are as follows:

(a) A general framework for assessing and improving the quality of soft-values is proposed.

(b) The Kullback-Leibler distance is introduced for measuring the quality of soft-values.

(c) The PCC decoder with LogAPP constituent decoders is improved[1] regarding the speed of convergence by correcting the extrinsic soft-values with respect to a criterion based on the Kullback-Leibler distance.

In addition to this, the meanings of "reliability information" and "reliability value" are discussed and clarified.

4.1 Reliability Values

In this section, we start with a suitable transmission model. Based on this model, the terms "reliability information" and "reliability value" are discussed and some properties

[1] Even though the improvement is small, it shows the suboptimality of the PCC decoder.

are stated.

The general model is depicted in Fig. 4.1. Binary info symbols[2] $U \in \mathbb{B}$ that are independent and uniformly distributed are transmitted over a binary-input symmetric memoryless channel (BISMC) $U \to V$. The channel outputs $V \in \mathbb{R}$ are required to be "LLR-like"; i.e., when estimating the transmitted info symbol u on basis of a soft-value v, the error probability is assumed to be large for small magnitudes of v, and it is assumed to be small for large magnitudes of v. A precise definition of "LLR-like" is given below. The channel output V is considered as an observation and the corresponding a-posteriori LLR $L \in \mathbb{R}$, is computed. For a given channel output v, we define

$$l := L(U|V = v) = \ln \frac{\Pr(U = +1|V = v)}{\Pr(U = -1|V = v)}. \tag{4.1}$$

Notice that the actual transition probabilities of the channel $U \to V$ have to be known for this computation. In practice, the distributions may be approximated by histograms determined by simulation.

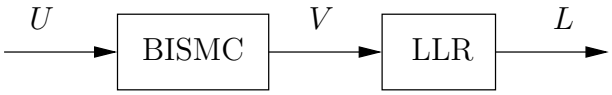


Figure 4.1: Transmission model for assessing reliability values.

The following definition specifies the intuitive explanation of "LLR-like" given above:

Definition 4.1 (LLR-like Soft-Values)
Assume a BISMC $U \to V$. The soft-value V is called LLR-like if $L(U|V = v)$ is monotonically increasing in v. —

This property has two implications: first, the channel $U \to L$ is a BISMC, and second, the sign of v is the ML (and MAP) estimate for U. Thus, if a soft-value is LLR-like, the soft-value and the corresponding LLR differ only in their absolute values, and this difference is "well-behaved" in some sense. If a soft-value is LLR-like but no LLR, we may call it *mismatched LLR* in the sequel[3].

This model is very general and includes in particular the following three scenarios:

1. A symbol is transmitted over an AWGN channel and the channel output is converted to an LLR assuming a wrong SNR. The channel $U \to V$ models the channel between the symbol and the mismatched LLR. (See also Example 4.1.)

2. Assume a transmission system with linear channel encoding and MaxLogAPP decoding. Then, the channel $U \to V$ models the channel between an info symbol and its mismatched a-posteriori LLR.

[2]The considerations hold similarly for code symbols.

[3]Since the term "mismatched LLR" provides an intuitive feeling of the correct meaning, we use it in spite it is self-contradictory.

3. Assume a transmission system with linear channel encoding and iterative decoding. Then, the channel $U \to V$ models the channel between an info or a code symbol and its mismatched extrinsic LLR.

Notice that possible dependencies between info or code symbols and possible dependencies between soft-values are explicitly not taken into account, as we investigate only the symbol-wise reliability of soft-values.

Example 4.1
Consider the conversion of the output y of a binary-input AWGN channel with input $x \in \mathbb{B}$ into the corresponding LLR $v := L(X|y)$. (This corresponds to the first of the above scenarios.) The conversion is given by

$$v = \frac{2}{\sigma_N^2} y,$$

as shown in Example 3.3, and it requires knowledge of the noise variance σ_N^2. When a wrong noise variance is applied, e.g., due to a wrong SNR estimation, the resulting soft-value v is not an LLR. However, it is LLR-like according to Definition 4.1, as can easily be seen. ◇

The LLR device in the model has no effect if the channel output v *is* already an LLR. This intuitive conjecture is proved in the following lemma.

Lemma 4.1
Assume that $v = L(U|Y = y)$ for observations[4] $y \in \mathbb{Y}$. Then, $L(U|V = v) = L(U|Y = y)$, or equivalently,

$$L(U|V = v) = v.$$

Proof. As $v = L(U|Y = y)$, we have

$$p_{U|Y}(u|y) = \frac{1}{1 + e^{-uv}} \tag{4.2}$$

for $u \in \mathbb{B}$ and $v \in \mathbb{V}$ (cf. Appendix C). Consider now the conditional probabilities of $U = u$:

$$\begin{aligned} p_{U|V}(u|v) &\overset{(a)}{=} \int_{y \in \mathbb{Y}} p_{U|Y}(u|y) p_{Y|V}(y|v) dy \\ &\overset{(b)}{=} \int_{y \in \mathbb{Y}} \frac{1}{1 + e^{-uv}} p_{Y|V}(y|v) dy \\ &\overset{(c)}{=} \frac{1}{1 + e^{-uv}} \int_{y \in \mathbb{Y}} p_{Y|V}(y|v) dy \\ &= \frac{1}{1 + e^{-uv}} = p_{U|Y}(u|y). \end{aligned}$$

We have used the following relations: (a) The three random variables U, Y, and V form the Markov chain

$$U \to Y \to V$$

[4]The observations may be scalars or vectors.

due to their definitions. (b) Identity (4.2). (c) If y leads to v, the values of the first factor (in the second line) are identical; otherwise, the second factor (in the second line) is equal to zero. Finally, as $p_{U|V}(u|v) = p_{U|Y}(u|y)$, we also have $L(U|Y = y) = L(U|V = v)$. QED

Due to the property given in Lemma 4.1, the function $L(U|.)$ is idempotent[5].

Remark 4.1
Lemma 4.1 can easily be generalized to include nonbinary symbols U or soft-values that are not LLR-like: the equality $p_{U|V}(u|v) = p_{U|Y}(u|y)$ holds if v is a sufficient statistic of y for U. The proof follows the same lines. As this thesis is restricted to binary symbols and LLR-like soft-values, Lemma 4.1 is sufficient.

The soft-value v may be separated into its sign and its magnitude. Since we assumed symmetry, the sign represents the estimated symbol (also called hard decision), and the magnitude represents the reliability of this estimate:

$$\begin{aligned} \hat{u} &:= \operatorname{sgn}(v), \\ a &:= \operatorname{abs}(v). \end{aligned}$$

The values $\hat{u}$ and a are regarded as realizations of the random variables $\hat{U}$ and A. Notice that A is a subchannel indicator for the channel $U \to V$, and that A induces a decomposition into BSCs (cf. Chapter 2). At first view, it seems to be natural to denote a as "reliability". However, "reliability" implies that the soft-value can be correctly interpreted (cf. Section 3.2). In the general case, a *reliability value* of a symbol U may be defined as a value from which the a-posteriori probabilities of all symbol values can be computed. As in this thesis, soft-values are basically required to be LLRs, we define the reliability of a soft-value as follows.

Definition 4.2 (Reliability of a Soft-Value)
A soft-value $v \in \mathbb{R}$ for a binary symbol $U \in \mathbb{B}$ is called reliable if it is equal to the corresponding LLR, i.e., if $v = L(U|V = v)$. The absolute value of the LLR,

$$\lambda := |L(U|V = v)|,$$

is called reliability value. —

Due to the assumed symmetry, the reliability value may equivalently be defined using the magnitude of the soft-value:

$$\lambda := |L(U|A = a)|.$$

Similarly to Lemma 4.1, we have the following property.

Lemma 4.2
Assume that $a = |L(U|Y = y)|$ *for observations*[6] $y \in \mathbb{Y}$. *Then we have*

$$|L(U|A = a)| = |L(U|Y = y)| = a.$$

[5] A function f is called idempotent if $f(f(x)) = f(x)$.
[6] The observations may be scalars or vectors.

The proof follows immediately from Lemma 4.1 and symmetry arguments. Due to this property, the function $|L(U|.)|$ is idempotent as well (cf. Footnote 5).

Example 4.2
Consider an AWGN channel with input symbols $+1$ and -1 and noise power σ_n^2. Let y denote the channel output, and let a denote its absolute value, $a := |y|$. Then, the mapping from a to the reliability value λ is given by $\lambda(a) = \frac{2}{\sigma_n^2}a$. ◇

As introduced in Section 2.8, the reliability value may be seen as a random variable Λ with realizations λ. Furthermore, the reliability value $\Lambda = \lambda$ represents the reliability of the subchannel corresponding to $A = a$.

Remark 4.2
Reliability is basically not a property of the soft-value itself. It is rather a question whether a soft-value represents what it is supposed to represent or in other words, whether the processing unit accepting the soft-value interprets it in the correct way. As all soft-values in this thesis are interpreted as LLRs, the above definition is sufficient.

Definition 4.2 immediately poses two questions about mismatched (non-reliable) LLRs. First, how can the degree of mismatch be measured? And second, how can the degree of mismatch be reduced? These questions are addressed in Section 4.3 and Section 4.4.

4.2 Measurement of Reliability Values

For a given channel $U \to V$, e.g., between the input of a convolutional encoder and the output of the corresponding LogAPP decoder, the reliability values $\lambda(a)$ may be determined by simulation. Notice that V is assumed to be LLR-like and thus real-valued. The following steps may be used:

(a) The channel output alphabet is quantized into a finite number of equidistant intervals, where one interval is symmetric around zero. The centers of these intervals are denoted by v_Q, and the set of centers is denoted by $\mathbb{V}_Q$. Thus, the actual BISMC $U \to V$ is approximated by the BISMC $U \to V_Q$, where $V_Q \in \mathbb{V}_Q$.

(b) The histograms $h_{V_Q|U}(v_Q|{+1})$ and $h_{V_Q|U}(v_Q|{-1})$ are determined by simulation. Taking into account the symmetry $h_{V_Q|U}(v_Q|{+1}) = h_{V_Q|U}(-v_Q|{-1})$, the measurement of only one of the two histograms is sufficient.

(c) As $L(U) = 0$ is assumed, the LLR of the quantized channel output is given by

$$l_Q := L(U|V_Q = v_Q) = L(V_Q = v_Q|U) \approx \ln \frac{h_{V_Q|U}(v_Q|{+1})}{h_{V_Q|U}(v_Q|{-1})}.$$

(d) Let $a_Q := |v_Q|$ and $\lambda_Q := |l_Q|$. Then the reliability value of the quantized channel output is given by the two equivalent expressions

$$\lambda_Q = \left|\ln \frac{h_{V_Q|U}(+a_Q|{+1})}{h_{V_Q|U}(+a_Q|{-1})}\right| = \left|\ln \frac{h_{V_Q|U}(+a_Q|{+1})}{h_{V_Q|U}(-a_Q|{+1})}\right|.$$

Finally, we may approximate $\lambda \approx \lambda_Q$ and $l \approx l_Q$. (Notice that the two approximations are equivalent.) These approximations are unbiased, and the precision depends only on the precision of the histogram measurement.

As $h_{V_Q|U}(v_Q|+1)$ results from measurements, it may be the case that $h_{V_Q|U}(a_Q|+1) = 0$ or $h_{V_Q|U}(-a_Q|+1) = 0$ for certain soft-value magnitudes a_Q. (This holds similarly for $h_{V_Q|U}(v_Q|-1)$.) In either case no reasonable reliability value λ_Q can be computed, and the measurements for the soft-values $v_Q = +a_Q$ and $v_Q = -a_Q$ can be discarded.

4.3 Measurement of Reliability Mismatch

The quality of a soft-value depends on the difference between the soft-value and its corresponding LLR, as soft-values are supposed to be LLRs. If this difference is zero, the soft-value is called reliable according Definition 4.2. If this difference is unequal to zero, the soft-value is called a mismatched soft-value (or a mismatched LLR). In this case, we speak of *reliability mismatch*, as the reliability claimed by the soft-value, its magnitude, differs from the actual reliability value, the absolute value of the LLR: the smaller the value of the reliability mismatch, the better the quality of the soft-value. The converse of the mismatch may be called the *reliability of reliability values*. To avoid confusion, "reliability" will only be used in the sense "reliability value" in the sequel. A major task in this context is to find a suitable and appropriate measure of reliability mismatch.

The reliability mismatch of a soft-value V may be visualized by plotting the LLR $l = L(U|V = v)$ versus the soft-value v, as proposed in [HR90, Hoe97]. Since we assume symmetry, it is sufficient to consider the absolute values, and we may equivalently plot the reliability value $\lambda := |l|$ versus the soft-value magnitude $a := |v|$. This plot may be interpreted using the concept of decomposing a BISMC into BSCs (cf. Section 2.2). As A is a subchannel indicator for the channel $U \to V$, this plot shows the reliability value $\Lambda = \lambda$ of each subchannel $A = a$.

Example 4.3
Consider a terminated convolutional code encoded by the generator polynomials $g_1(D) = 1 + D + D^3$ and $g_2(D) = 1 + D + D^2 + D^3$ with info word length $K = 1000$ and code word length $N = 2006$. The code symbols are transmitted over an AWGN channel with SNR $E_b/N_0 = -2$ dB, 0 dB, +2 dB. The decoder performs MaxLogAPP decoding. As MaxLogAPP decoding only approximates LogAPP decoding (cf. Section 3.4), the post-decoding values for the info symbols are only approximate LLRs. The reliability values are measured using the approach from Section 4.2.

The reliability values λ_Q are plotted versus the soft-value magnitudes a_Q in Fig. 4.2. (Additionally, the probability distributions of the soft-value magnitudes are provided in Fig. 4.3.) Notice that LogAPP decoding results in the 45-degree line, which corresponds to $\lambda_Q = a_Q$ and may be regarded as the ideal curve. The measured curves (for MaxLogAPP decoding) are all below this ideal curve. Thus, the soft-values pretend to be more reliable than they actually are, and MaxLogAPP decoding may be said to be too optimistic [PR96].

As the measured curves are different from the ideal curve, the soft-values are *not* reliable in the sense of Definition 4.2. However, the "visual distance" becomes smaller for increasing SNR. (This is not surprising, because MaxLogAPP decoding approaches LogAPP decoding for increasing SNR.) Therefore, the soft-values at higher SNR may be regarded as "more reliable" (in some sense) than the soft-values at lower SNR. In other words: The soft-values exhibit a better quality when the SNR is larger. ◇

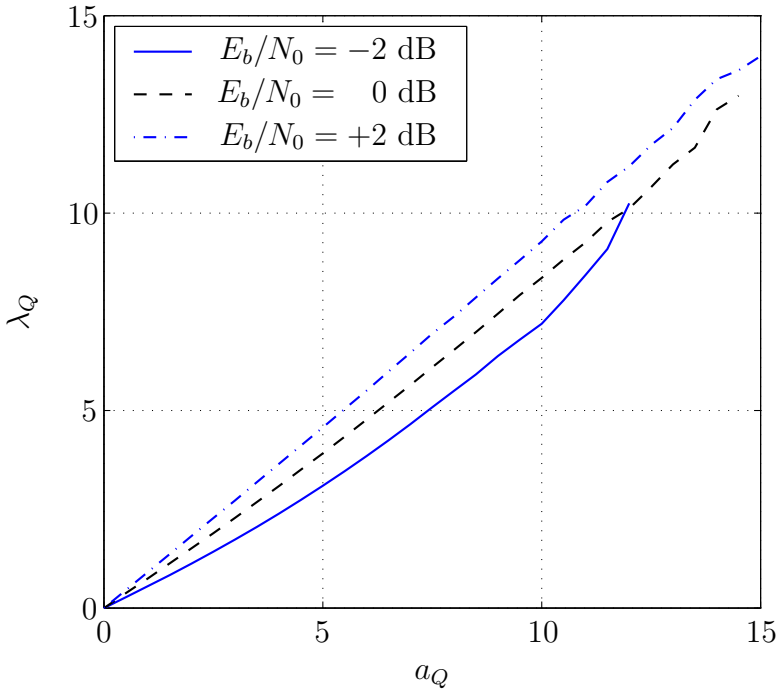


Figure 4.2: Reliability values λ_Q versus soft-value magnitudes a_Q. Convolutional code with MaxLogAPP decoding from Example 4.3; several signal-to-noise ratios.

The discussion of Fig. 4.2 in Example 4.3 motivates to introduce *a measure for the reliability mismatch*. Doing so, the difference between the soft-value magnitude a and its reliability value λ can be determined not only qualitatively, as done in Example 4.3, but also quantitatively. To measure the reliability mismatch, we have to find an appropriate distance between soft-values and their corresponding LLRs.

The following two distance measures are proposed in the literature. The first measure (ABS) compares the average absolute values of L and V [CF02]:

$$d_{\mathsf{ABS}}(L,V) := \Big|\mathrm{E}\{|L|\} - \mathrm{E}\{|V|\}\Big|. \tag{4.3}$$

The second measure (DIF) averages over the absolute difference between L and V [vDJK03]:

$$d_{\mathsf{DIF}}(L,V) := \mathrm{E}\Big\{|L-V|\Big\}. \tag{4.4}$$

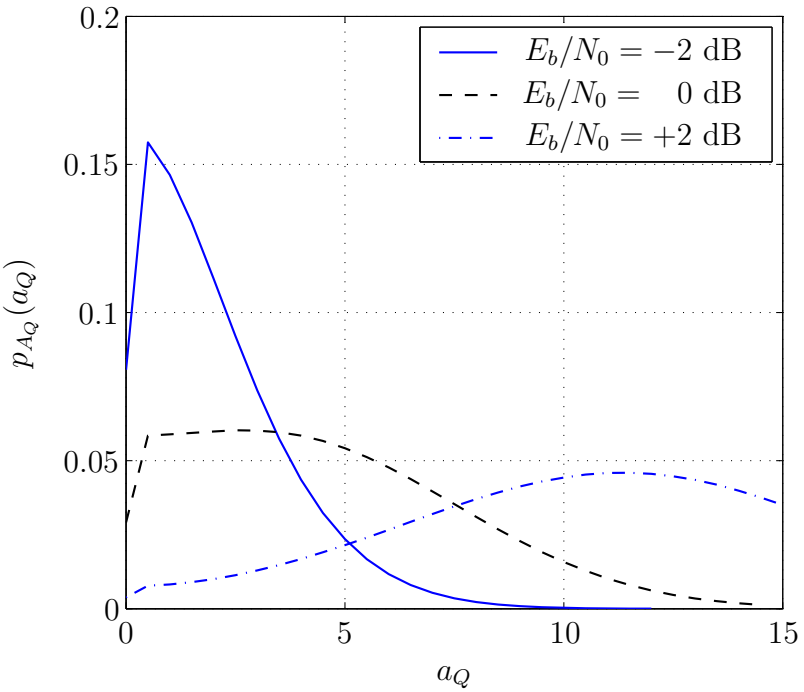


Figure 4.3: Probability mass function $p_{A_Q}(a_Q)$ of soft-value magnitudes a_Q. Convolutional code with MaxLogAPP decoding from Example 4.3; several signal-to-noise ratios.

These two approaches show certain disadvantages. In the former measure, large absolute values are more emphasized than small absolute values, although small absolute values are more critical. The latter measure corresponds to a mean error without taking into account the meaning of the soft-values.

As the quality of soft-values is in some sense related to the associated mutual information (MI), the difference between the mutual information associated to V and that associated to L,

$$d_{\mathsf{MI}}(L,V) := \big|I(U;L) - I(U;V)\big|, \tag{4.5}$$

seems to be a suitable measure of reliability mismatch, at a first glance. However, this is not the case, as can easily be shown. The three random variables involved form the Markov chain $U \to V \to L$, and L is a sufficient statistic of V. Therefore, we have $I(U;L) = I(U;V)$, and thus $d_{\mathsf{MI}}(L,V) = 0$. Since this follows directly from the definition of L, this distance measure is not useful.

Due to the given disadvantages, these three distance measures are not further investigated. Instead, the average Kullback-Leibler distance is proposed as a distance measure. Its use follows from the true meaning of LLRs in a straight-forward way. We start with a single subchannel and then extend the results by taking the expectation with respect to the subchannels.

Consider the subchannel $U \to V|A = a$ with reliability value $\Lambda = \lambda$. Assume that the soft-value is v and that the corresponding LLR is l. Then, we have two probability distributions of the symbol U. On the one hand, we have the probability distribution $p_{U||V}$

that results from the assumption that the soft-value v is an LLR:

$$\begin{aligned} p_{U||V}(+1||v) &:= \frac{1}{1+e^{-v}}, \\ p_{U||V}(-1||v) &:= \frac{1}{1+e^{+v}}. \end{aligned} \tag{4.6}$$

This distribution is called the *interpreted distribution*, as it results from the interpretation of the soft-value as an LLR. On the other hand, we have the distribution $p_{U||L}$ that results from the assumption that the LLR l is an LLR (which is true, of course):

$$\begin{aligned} p_{U||L}(+1||l) &:= \frac{1}{1+e^{-l}}, \\ p_{U||L}(-1||l) &:= \frac{1}{1+e^{+l}}. \end{aligned} \tag{4.7}$$

This distribution is called the *actual distribution* (for obvious reasons). We use the special notation "||" to emphasize the difference to conditional probabilities; it may be read "... based on LLR...". In fact, $p_{U||V}$ and $p_{U||L}$ are functions mapping a real-value, namely the soft-value v or the LLR l, to a probability distribution of a binary random variable, namely the symbol U. Notice that

$$p_{U||V}(u||v) \neq p_{U|V}(u|v)$$

due to the wrong interpretation of the soft-value, whereas

$$p_{U||L}(u||l) = p_{U|L}(u|l) = p_{U|V}(u|v)$$

due to the definition of l; $p_{U|V}(u|v)$ and $p_{U|L}(u|l)$ denote the conditional distributions of U given $V = v$ and $L = l$, respectively.

The distance between the soft-value v and the LLR l is thus in fact a distance between the two underlying probability distributions $p_{U||V}(u||v)$ and $p_{U||L}(u||l)$. A commonly applied distance measure for probability distributions is the *Kullback-Leibler distance* (KLD) (cf. Appendix D). Accordingly, we may define the distance between v and l as the Kullback-Leibler distance between $p_{U||V}(u||v)$ and $p_{U||L}(u||l)$:

$$\begin{aligned} d_{\mathsf{KLD}}(l,v) &:= D\Big(p_{U||L}(u||l)\Big\|p_{U||V}(u||v)\Big) \\ &= \sum_{u\in\mathbb{B}} p_{U||L}(u||l)\,\mathrm{ld}\,\frac{p_{U||L}(u||l)}{p_{U||V}(u||v)}. \end{aligned} \tag{4.8}$$

The Kullback-Leibler distance has two important properties: (i) it is nonnegative, and (ii) it is zero precisely if the two distributions are identical (cf. Appendix D).

Example 4.4
Assume the soft-value $v = 0.8$ has the associated LLR $l \approx 1.4$. The interpreted distribution of U is given according to (4.6):

$$p_{U||V}(+1||0.8) \approx 0.7, \qquad p_{U||V}(-1||0.8) \approx 0.3.$$

Correspondingly, the actual distribution of U is given according to (4.7):

$$p_{U||L}(+1||1.4) \approx 0.8, \qquad p_{U||L}(-1||1.4) \approx 0.2.$$

(The values are only approximated for clarity.) The distance between $l = 1.4$ and $v = 0.8$ is the Kullback-Leibler distance

$$\begin{aligned} d_{\mathsf{KLD}}(1.4, 0.8) &= D\Big([0.8\ 0.2]\Big\|[0.7\ 0.3]\Big) \\ &\approx 0.8 \ln \frac{0.8}{0.7} + 0.2 \ln \frac{0.2}{0.3} \approx 0.032. \end{aligned}$$

◇

Based on this distance between two values v and l, we may define the average distance between the random variables V and L as the corresponding expectation. As l is a function of v, averaging over v is sufficient. The above considerations are summarized in the following definition.

Definition 4.3 (KLD Reliability Mismatch)
Consider a binary symbol U, a soft-value V for this symbol, and the corresponding LLR L. The probability distributions of U resulting from V and L are denoted by $p_{U||V}(u||v)$ and $p_{U||L}(u||l)$ according to (4.6) and (4.7), respectively. The Kullback-Leibler distance (KLD) distance between two realizations v and l is defined as the Kullback-Leibler distance between $p_{U||V}(u||v)$ and $p_{U||L}(u||l)$:

$$d_{\mathsf{KLD}}(l, v) := D\Big(p_{U||L}(u||l)\Big\|p_{U||V}(u||v)\Big)$$

(cf. Equ. (4.8)). The Kullback-Leibler distance between the soft-value V and the LLR L is defined as the average Kullback-Leibler distance

$$d_{\mathsf{KLD}}(L, V) := \mathop{\mathrm{E}}_{v \in \mathbb{R}} \Big\{ d_{\mathsf{KLD}}(l, v) \Big\}.$$

The value $d_{\mathsf{KLD}}(L, V)$ is called the KLD reliability mismatch. —

The KLD reliability mismatch is nonnegative, and it is equal to zero precisely if $V = L$, i.e., if the soft-value is reliable. This follows immediately from the two properties of the Kullback-Leibler distance mentioned above. Due to symmetry, we have the identities

$$\begin{aligned} d_{\mathsf{KLD}}(l, v) &= d_{\mathsf{KLD}}(\lambda, a), \\ d_{\mathsf{KLD}}(L, V) &= d_{\mathsf{KLD}}(\Lambda, A), \end{aligned}$$

with $A = |V|$ and $\Lambda = |L|$. Therefore, we may equivalently speak of the reliability mismatch between Λ and A.

The two measures of reliability mismatch that are based on the difference of the mean values and on the average differences, as given in (4.3) and (4.4), are ad-hoc approaches and do not have particular meanings. As opposed to that, the measure based on the KLD has a strong background from probability theory and follows immediately when taking into account that each LLR actually parameterizes a probability distribution of a binary symbol.

Example 4.5
We continue Example 4.3. The Kullback-Leibler distances $d_{\mathsf{KLD}}(\lambda_Q, a_Q)$ are plotted versus the values a_Q in Fig. 4.4. The average Kullback-Leibler distance between Λ_Q and A_Q can be computed as

$$d_{\mathsf{KLD}}(\Lambda_Q, A_Q) = \sum_{a_Q} p_{A_Q}(a_Q)\, d_{\mathsf{KLD}}(\lambda_Q, a_Q);$$

summation over a_Q is sufficient, as λ_Q is a function of a_Q. The probability distribution of A_Q is depicted in Fig. 4.3. The KLD reliability mismatch for each SNR results as

E_b/N_0	-2 dB	0 dB	$+2$ dB
$d_{\mathsf{KLD}}(\Lambda_Q, A_Q)$	$5.7 \cdot 10^{-2}$	$1.1 \cdot 10^{-2}$	$3.3 \cdot 10^{-4}$

When doing the same simulations with LogAPP decoding, the measured KLD reliability mismatch results to about 10^{-5}. As the theoretical reliability mismatch is equal to zero, this value is only due to the finite precision of the histogram measurements. ◇

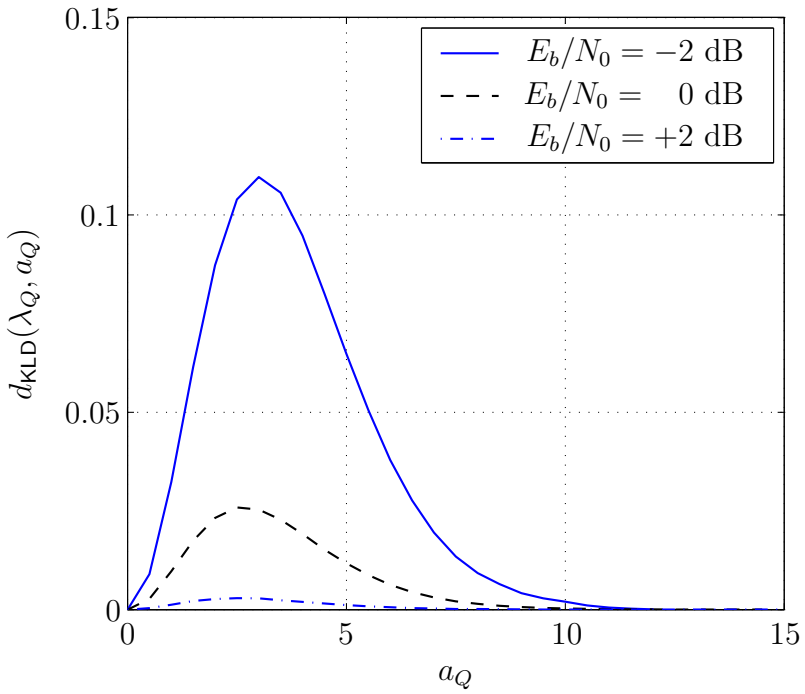


Figure 4.4: Kullback-Leibler distances $d_{\mathsf{KLD}}(\lambda_Q, a_Q)$. Convolutional code with MaxLogAPP decoding from Example 4.5; several signal-to-noise ratios.

4.4 Correction of Reliability Mismatch

Mismatched soft-values may be corrected by memoryless[7] post-processing. Ideally, the soft-values should be reliable after applying such a mapping, i.e., they should be equal to the corresponding LLRs. As this may not be feasible in practice, the soft-values after the mapping are at least required to be closer to the true LLRs than the soft-values before the mapping; i.e., the reliability mismatch after the mapping should be smaller than that before the mapping. A general framework for improving or even correcting soft-values is discussed in this section.

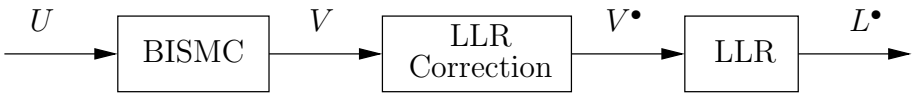


Figure 4.5: Model for reducing the reliability mismatch, and thus for improving reliability values.

Model for Correcting Soft-Values

The model for reducing the reliability mismatch is shown in Fig. 4.5. It extends the one shown in Fig. 4.2 by a correction function (LLR Correction).

An equiprobable binary symbol $U \in \mathbb{B}$ is transmitted over a BISMC $U \to V$ including the actual channel and an approximate conversion to LLRs. This LLR-like output is the soft-value $V \in \mathbb{R}$. A function f (LLR Correction) maps this soft-value to an improved (or corrected) soft-value $V^\bullet \in \mathbb{R}$:

$$\begin{aligned} f : \mathbb{R} &\to \mathbb{R} \\ v &\mapsto v^\bullet = f(v). \end{aligned}$$

This *correction function* f is required to be monotonically increasing and to fulfill the symmetry condition $f(-v) = -f(v)$. Finally, the improved soft-value is converted to the corresponding *true LLR*

$$l^\bullet := L(U|V^\bullet = v^\bullet).$$

We write $l^\bullet$ to distinguish this LLR from $l := L(U|V = v)$. The magnitudes of the soft-values and the LLRs are written as

$$\begin{aligned} a &:= |v|, & a^\bullet &:= |v^\bullet|, \\ \lambda &:= |l|, & \lambda^\bullet &:= |l^\bullet|. \end{aligned}$$

Since we have $a^\bullet = f(a)$ due to symmetry, we may equivalently define f only for nonnegative values and use

$$\begin{aligned} v^\bullet &= \operatorname{sgn}(v) \cdot f(|v|) \\ &= \operatorname{sgn}(v) \cdot f(a). \end{aligned}$$

[7] Mappings with memory are not considered, as such mappings would be closer to decoding rather than to post-processing, and are therefore not within the scope of this section.

This notation explicitly shows that f changes only the magnitudes of v, but not the signs.

Ideal Correction Function

The ideal correction function maps the soft-value to its LLR,

$$f_{\mathsf{LLR}}(v) = L(U|V = v),$$

so that the soft-value magnitude is equal to the reliability value:

$$a^{\bullet} = f_{\mathsf{LLR}}(a) = \lambda.$$

Since U is uniformly distributed, we have

$$L(U|V = v) = L(V = v|U) = \ln \frac{p_{V|U}(v| + 1)}{p_{V|U}(v| - 1)}.$$

The ideal correction function can thus be used only if the conditional distributions $p_{V|U}(v|u)$ are precisely known, i.e., if they can be determined by analysis or by simulation with sufficient precision. These conditional distributions are known for the outputs of the approximated boxplus operator [vDJK03, LS04]; for more complicated cases, they are assumed to be rather involved. A simulation with sufficient precision is usually not feasible: as soft-values with large magnitudes and erroneous sign ($\mathrm{sgn}(v) \neq u$) occur only very rarely, the conditional distributions cannot be determined with sufficient precision. On the other hand, a good approximation of the ideal correction function may already yield a gain in performance. Such correction functions are addressed in the following section.

Parameterized Correction Function

A general approach for finding a good correction function is as follows:

1. A suitable parameterized function is chosen as the correction function.
2. The parameters of this function are optimized such that the reliability mismatch after applying this function is minimized.

The degrees of freedom are the parameterized function and the measure of reliability mismatch. As such measures have already been discussed in the previous section, we focus on the functions here.

We start with a function f of a that depends on a vector of parameters $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \ldots, \alpha_{M-1}]$:

$$a^{\bullet} = f(a; \boldsymbol{\alpha}).$$

This function is required to be monotonically increasing (cf. above), and its shape should be similar to the curve it is matched to, i.e., to the ideal correction function f_{LLR}. As f_{LLR} is usually not available (otherwise we would directly use it, of course), we employ an estimated function $\hat{f}_{\mathsf{LLR}}$ found by measurements (cf. Section 4.2).

When the post-decoding soft-values of MaxLogAPP decoders or of LogAPP decoders within iterative decoders[8] are to be processed, the following parameterized correction functions may be suitable:

$$\begin{aligned} f_1(a;\alpha_0) &:= \alpha_0 \cdot a, \\ f_2(a;\alpha_0,\alpha_1) &:= \min\{\alpha_0 \cdot a, \alpha_1\}, \\ f_3(a;\alpha_0,\alpha_1,\alpha_2) &:= \min\{\alpha_0 \cdot a, \alpha_1 \cdot a + \alpha_2\}, \\ f_4(a;\alpha_0,\alpha_1,\alpha_2) &:= \min{}^{\star}\{\alpha_0 \cdot a, \alpha_1 \cdot a + \alpha_2\}, \end{aligned} \tag{4.9}$$

where $\alpha_0, \alpha_1, \alpha_2 \geq 0$ such that the functions fulfill the requirements. (The function $\min^{\star}$ is defined in Appendix C). Examples of these functions are depicted in Figure 4.6. Function f_1 adapts the magnitudes a in a uniform way by scaling. Function f_2 does the same and additionally clips a, so that the reliability values are limited by α_1. Function f_3 scales the magnitudes in a nonuniform way, i.e., there are two linear regions, as opposed to f_1. Finally, function f_4 is very similar to f_3 but shows a smooth transition between the two linear regions.

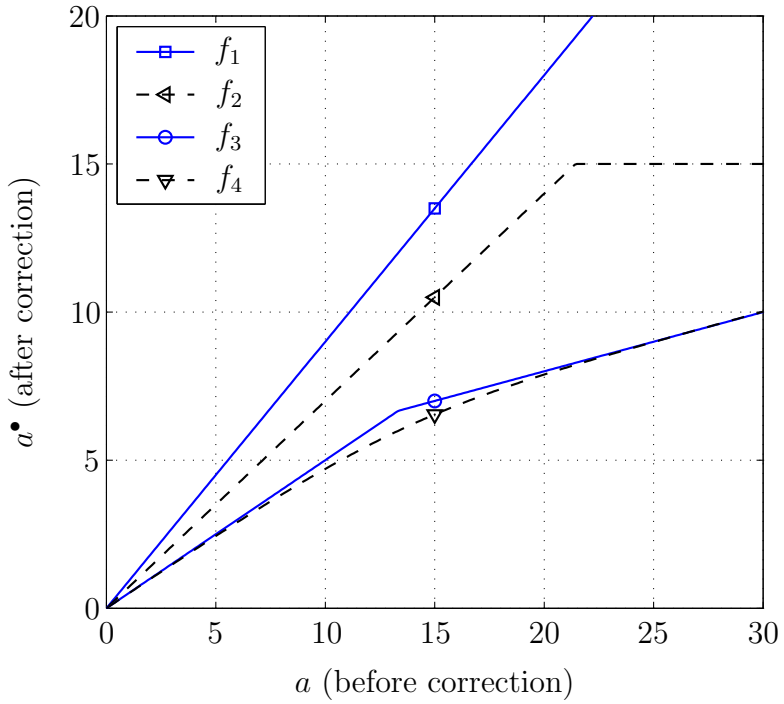


Figure 4.6: Examples for the parameterized correction functions given in (4.9): $a^{\bullet} = f_i(a;\boldsymbol{\alpha})$, $i = 1, 2, 3, 4$.

There is an obvious trade-off: correction functions with many parameters can better approximate the ideal correction function; on the other hand, the more parameters have to be determined, the more difficult is the optimization of each individual parameter. In

[8]LogAPP decoding within iterative decoders does not produce LLRs, as discussed in Section 3.5.

the following examples, we concentrate on the correction function f_1, as this function has shown to be sufficient. In general, however, the appropriate correction function has to be determined individually for each application.

The parameters of the chosen correction function may now be determined such that the reliability mismatch becomes minimal. In the sequel, we focus on the KLD reliability mismatch, given in Definition 4.3. The optimization criterion for the parameter vector $\boldsymbol{\alpha}$ of a function $f(a;\boldsymbol{\alpha})$ is thus

$$d_{\mathsf{KLD}}(V^\bullet, L^\bullet) \quad \xrightarrow{\boldsymbol{\alpha}} \quad \min. \tag{4.10}$$

Notice that $L^\bullet$ is the LLR for U conditioned on $V^\bullet$, as defined in the model, and not the LLR conditioned on V.

Remark 4.3
In practice, the parameters $\boldsymbol{\alpha}$ of a correction function $f(v;\boldsymbol{\alpha})$ may be optimized in a rather straight-forward way. Some subtle details are discussed in the sequel. The considerations are based on the method and notation introduced in Section 4.2. Starting with a BISMC $U \to V$, the LLR-like output $V \in \mathbb{R}$ has been quantized to $V_Q \in \mathbb{V}_Q$, and the histograms $h_{V_Q|U}(v_Q|{+1})$ and $h_{V_Q|U}(v_Q|{-1})$ have been determined by simulation.

As only quantized values $v_Q \in \mathbb{V}_Q$ are available, the corresponding values of function $f(v;\boldsymbol{\alpha})$ are also quantized:

$$v_Q^\bullet := f(v_Q;\boldsymbol{\alpha}) \quad \in \mathbb{V}_Q^\bullet := f(\mathbb{V}_Q;\boldsymbol{\alpha}).$$

The histograms corresponding to $v_Q^\bullet$ are given by

$$\begin{aligned} h_{V_Q^\bullet|U}(v_Q^\bullet|{+1}) &= \sum_{v_Q : v_Q^\bullet} h_{V_Q|U}(v_Q|{+1}), \\ h_{V_Q^\bullet|U}(v_Q^\bullet|{-1}) &= \sum_{v_Q : v_Q^\bullet} h_{V_Q|U}(v_Q|{-1}), \end{aligned}$$

where summation is done over all v_Q with $f(v_Q;\boldsymbol{\alpha}) = v_Q^\bullet$. Thus, we may compute

$$l_Q^\bullet := L(U|V_Q^\bullet = v_Q^\bullet) = L(V_Q^\bullet = v_Q^\bullet|U) \approx \ln \frac{h_{V_Q^\bullet|U}(v_Q^\bullet|{+1})}{h_{V_Q^\bullet|U}(v_Q^\bullet|{-1})}.$$

If the correction function is strictly monotonically increasing, there is a one-to-one mapping between v_Q and $v_Q^\bullet$, and we have

$$L(U|V_Q^\bullet = v_Q^\bullet) = L(U|V_Q = v_Q),$$

which simplifies the optimization of the parameters. The functions f_1, f_3, and f_4 given in (4.9) are strictly monotonically increasing if $\alpha_0 \neq 0$ and $\alpha_1 \neq 0$.

The following example illustrates the concept for correcting soft-values and thus reducing the reliability mismatch.

Example 4.6
We continue Example 4.5 and consider only the case $E_b/N_0 = 0$ dB. For correcting the soft-values, we apply function f_1,

$$a_Q^\bullet := f_1(a_Q; \alpha_0) = \alpha_0 \cdot a_Q,$$

and optimize the parameter α_0 numerically. In Fig. 4.7, the KLD reliability mismatch $d_{\mathsf{KLD}}(\Lambda_Q^\bullet, A_Q^\bullet)$ is plotted versus the parameter α_0. The minimum KLD reliability mismatch is about $8.2 \cdot 10^{-5}$, and it is obtained for $\alpha_0 \approx 0.771$.

The reliability values before and after correction with the optimized function f_1 are shown in Fig. 4.8. The "visual" distance between the corrected curve and the ideal (45-degree) curve is small for small values and larger for large values. Obviously, the "difference" for small values provides the major contribution to the Kullback-Leibler reliability mismatch, as desired (cf. comment on other measures of reliability mismatch). Notice that large values $a_Q^\bullet$ are above the ideal curve; i.e., the corrected soft-values with large magnitude are too pessimistic. ◇

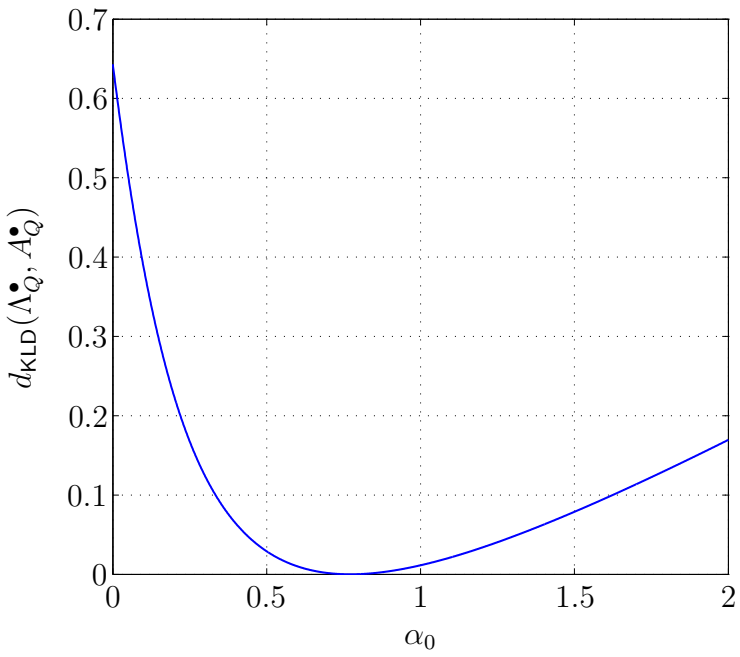


Figure 4.7: Average KLD reliability mismatch $d_{\mathsf{KLD}}(\Lambda_Q^\bullet, A_Q^\bullet)$ versus parameter α_0 of the correction function. Convolutional code with MaxLogAPP decoding from Example 4.6.

The previous examples have shown that MaxLogAPP decoding is suboptimal in the sense that the soft-outputs are not reliable, but they are very close to the LLRs. Accordingly, a small correction is sufficient. However, larger corrections may be required when other decoding schemes are employed. One important example is iterative decoding, where the pre-decoding soft-values delivered to the constituent decoders are usually not conditionally independent LLRs[9] after a few iterations. This issue is addressed in the following section.

[9]Cf. (3.21) and (3.22).

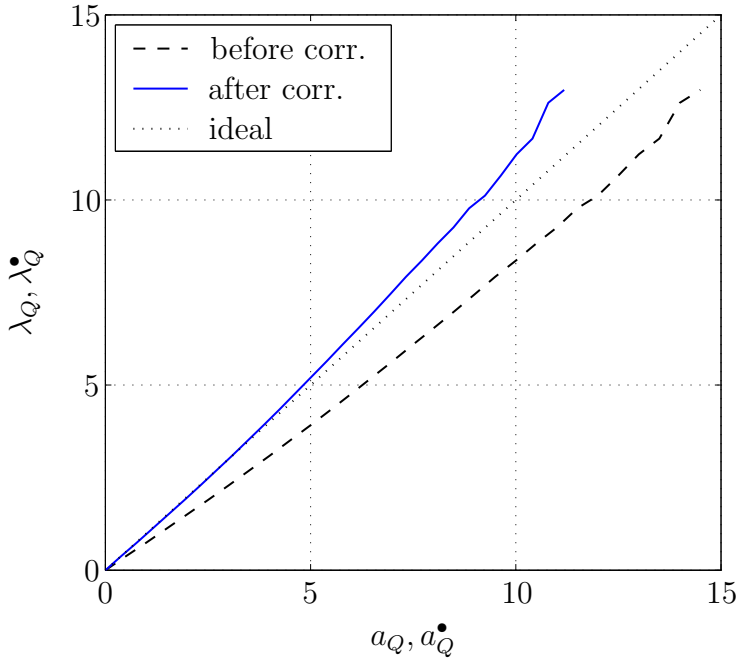


Figure 4.8: Reliability values before correction, λ_Q versus a_Q, and after correction, $\lambda_Q^\bullet$ versus $a_Q^\bullet$. Convolutional code with MaxLogAPP decoding from Example 4.6.

4.5 Application to Iterative Decoding

The parallel concatenated code[10] (PCC) and the corresponding iterative decoder, denoted as PCC decoder, have been explained in Section 3.5.1. The theoretical analysis of the decoding scheme has shown that the extrinsic soft-values that are passed on are not conditionally independent LLRs[11] after the first half-iteration. Thus, starting from the first full-iteration, the extrinsic soft-values are not LLRs, even if LogAPP decoding is applied by the constituent decoders.

A correction function, as discussed in the previous section, may be applied to improve the extrinsic soft-values, so that the reliability mismatch is reduced. Using a carefully chosen correction function, the iterative decoder converges faster or even achieves a lower error rate. This section gives some examples illustrating this concept.

The PCC decoder including the correction functions is depicted in Fig. 4.9 (cf. Fig. 3.5). The correction function $f^{(1)}$ maps the extrinsic values computed by Decoder 1, $\boldsymbol{w}^{(1)}_{u^{(1)}}$, to the values $\boldsymbol{w}^{\bullet(1)}_{u^{(1)}}$; the correction function $f^{(2)}$ maps the extrinsic values computed by Decoder 2, $\boldsymbol{w}^{(2)}_{u^{(1)}}$, to the values $\boldsymbol{w}^{\bullet(2)}_{u^{(1)}}$. As this mapping is memoryless, interleaving has no effect, and we may omit the subindices. The resulting models are depicted in Fig. 4.10, and they follow the model from Fig. 4.5. Notice that the channels between info symbols U and extrinsic values $W^{(1)}$ and $W^{(2)}$ are modeled as BISMCs (cf. Definition 3.5

[10]The presented concept may be applied to a serially concatenated code (SCC) in a similar way.
[11]Cf. (3.21) and (3.22).

and comments thereafter). Similarly to the previous sections, we define

$$l^{(1)} := L(U|W^{(1)} = w^{(1)}), \qquad w^{\bullet(1)} := f^{(1)}(w^{(1)}), \qquad l^{\bullet(1)} := L(U|W^{\bullet(1)} = w^{\bullet(1)}),$$
$$l^{(2)} := L(U|W^{(2)} = w^{(2)}), \qquad w^{\bullet(2)} := f^{(2)}(w^{(2)}), \qquad l^{\bullet(2)} := L(U|W^{\bullet(2)} = w^{\bullet(2)}).$$

The soft-value magnitudes and the reliability values are denoted by

$$a^{(1)} := |w^{(1)}|, \qquad \lambda^{(1)} := |l^{(1)}|, \qquad a^{\bullet(1)} := |w^{\bullet(1)}|, \qquad \lambda^{\bullet(1)} := |l^{\bullet(1)}|,$$
$$a^{(2)} := |w^{(2)}|, \qquad \lambda^{(2)} := |l^{(2)}|, \qquad a^{\bullet(2)} := |w^{\bullet(2)}|, \qquad \lambda^{\bullet(2)} := |l^{\bullet(2)}|.$$

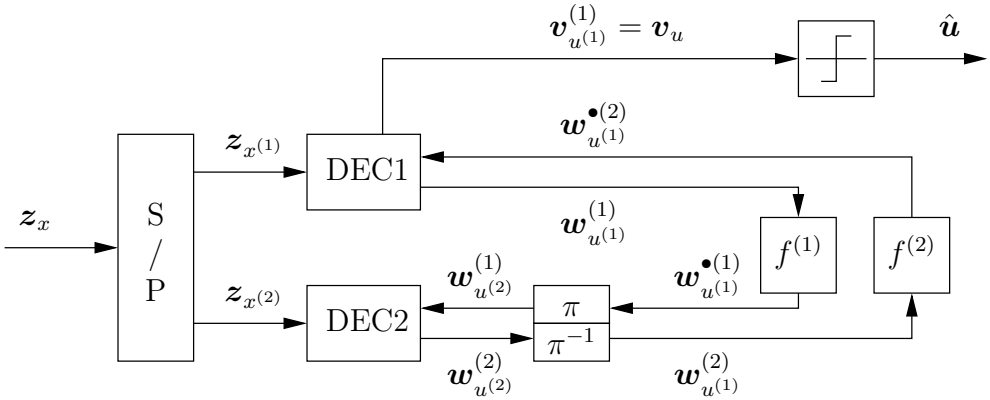


Figure 4.9: PCC decoder with soft-values passed through correction functions $f^{(1)}$ and $f^{(2)}$.

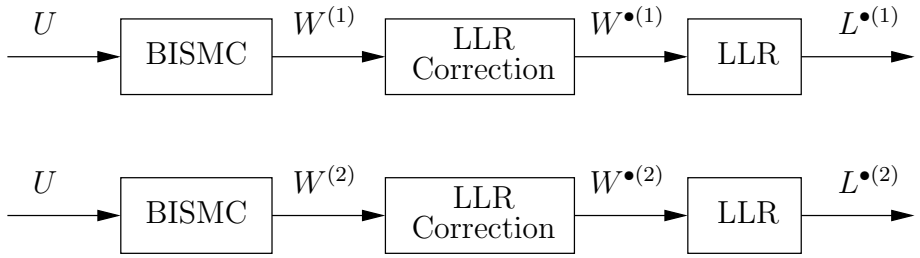


Figure 4.10: Models for improving reliability values for Decoder 1 (upper part) and Decoder 2 (lower part).

As mentioned above, the extrinsic soft-values after the first full-iteration are no LLRs. This may be shown by plotting the reliability values versus the soft-value magnitudes for each iteration and for each constituent decoder.

Example 4.7
Consider a PCC of info word length $K = 250$, code word length $N = 506$, and thus code rate $R = 250/506 = 0.49 \approx 1/2$. We use the constituent codes and the

interleaver specified in the UMTS standard [3GP00]. The two constituent encoders are terminated recursive systematic convolutional (RSC) encoders of rate 1/2 and memory length 3, defined by the generator function

$$\begin{bmatrix} 1 & \dfrac{1+D+D^3}{1+D^2+D^3} \end{bmatrix}.$$

The first and the second constituent code are punctured according to the puncturing matrices

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$$

respectively, to obtain the desired code rate. This puncturing scheme corresponds to the one presented in [BGT93] and results in a systematic PCC.

The code is used for transmission over an AWGN channel with SNR $E_b/N_0 = 2$ dB. The iterative decoder employs LogAPP decoders or MaxLogAPP decoders for the constituent decoders, and performs 10 iterations.

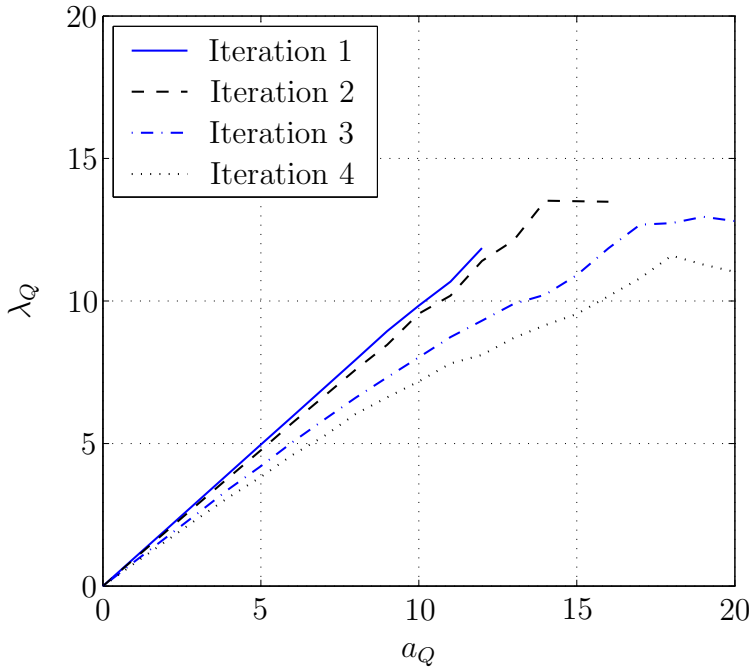
For each constituent decoder and for each iteration, the reliability values are measured using the method from Section 4.2. The results for LogAPP decoding and MaxLogAPP decoding are depicted in Fig. 4.11(a) and Fig. 4.11(b), respectively. Only the first few iterations for Decoder 1 are shown; the reliability values remain about the same afterwards, and those for Decoder 2 are similar. For both decoding algorithms, the soft-value magnitudes are different from the reliability values, and thus the soft-values are not reliable. As expected, the difference is larger when MaxLogAPP decoding is applied. ◇

The extrinsic soft-values may be corrected using correction functions, as presented in the previous section. As a result, an improved performance of the iterative decoder can be expected because each constituent decoder bases its computations on the assumption that the soft-values at its input are in fact LLRs. Improved performance means that the iterative decoder converges faster or that the bit error rate becomes smaller. In the sequel, first the general concept is described and then an example is given for illustration.

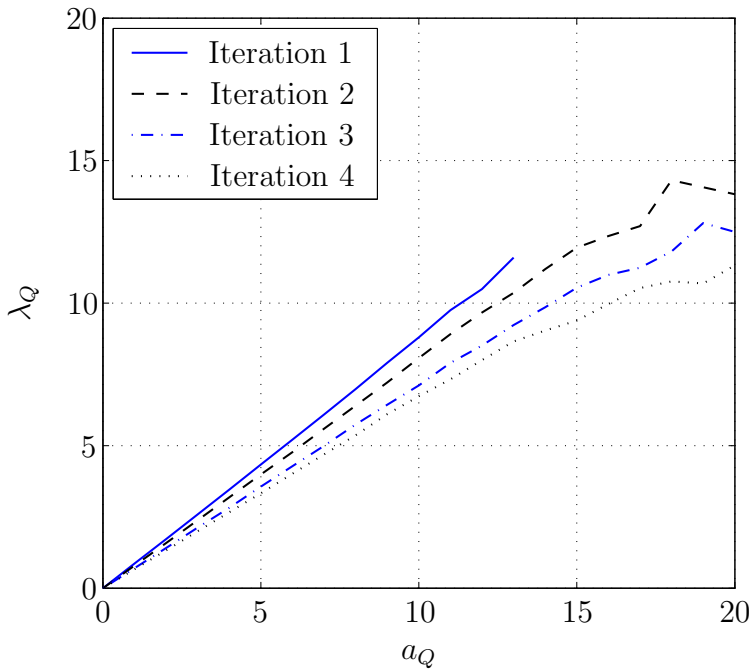
The decoding result of each constituent decoder depends on the previous decoding result of the other constituent decoder. Therefore, the optimal parameters of the correction functions may depend on the iteration. To enable this distinction, we denote the correction function applied to the extrinsic values computed by Decoder 1 in the first half-iteration of iteration i by $f^{(1)[i]}$; correspondingly, we denote the correction function applied to the extrinsic values computed by Decoder 2 in the second half-iteration of iteration i by $f^{(2)[i]}$. These correction functions have to be successively optimized for each half-iteration. The first four steps are as follows:

1. Determine the reliability values for the extrinsic values computed by Decoder 1 in the first half-iteration of iteration 1. Optimize the correction function $f^{(1)[1]}$.

2. Applying $f^{(1)[1]}$, determine the reliability values for the extrinsic values computed by Decoder 2 in the second half-iteration of iteration 1. Optimize the correction function $f^{(2)[1]}$.



(a) LogAPP decoding.



(b) MaxLogAPP decoding.

Figure 4.11: Reliability values λ_Q versus soft-value magnitudes a_Q, constituent decoder 1. PCC with LogAPP and MaxLogAPP decoding from Example 4.7.

3. Applying $f^{(1)[1]}$ and $f^{(2)[1]}$, determine the reliability values for the extrinsic values computed by Decoder 1 in the first half-iteration of iteration 2. Optimize the correction function $f^{(1)[2]}$.

4. Applying $f^{(1)[1]}$, $f^{(2)[1]}$, and $f^{(1)[2]}$, determine the reliability values for the extrinsic values computed by Decoder 2 in the second half-iteration of iteration 2. Optimize the correction function $f^{(2)[2]}$.

The procedure has to be continued up to the maximum number of iterations to be used.

Example 4.8
We continue Example 4.7. To provide for low computational complexity, we choose correction function f_1 from (4.9), i.e.,

$$f^{(d)[i]}(w) = \alpha^{(d)[i]} \cdot w,$$

where $d = 1, 2$ denotes the constituent decoder and i denotes the iteration. The optimal parameters $\alpha^{(d)[i]}$ are successively determined as described above, where the KLD reliability mismatch, and the two measures given in (4.3) (ABS), (4.4) (DIF) are used as optimization criteria. This is done for both LogAPP decoding and MaxLogAPP decoding using 10 iterations.

Table 4.1 lists the parameters optimized with respect to the KLD reliability mismatch and those optimize with respect to the criteria are given in (4.3) and (4.4). The value 1 corresponds to no correction; values less than 1 indicate that the soft-values before correction are too optimistic. The values for MaxLogAPP decoding are smaller than that for LogAPP decoding. This is not surprising, as MaxLogAPP decoding may be seen as an approximation of LogAPP decoding, and thus, the resulting soft-values can be expected to have "poorer" quality.

The decisive criterion whether the correction of the soft-values has been successful is the resulting error rate. In Fig. 4.12, the bit error rate (BER) is plotted versus the iterations for LogAPP and MaxLogAPP decoding, each without correction function and with correction functions optimized with respect to the three criteria. MaxLogAPP decoding is improved for all three criteria. (A similar result was also reported in [vDJK03].) The best performance is achieved when the parameters are optimized according to the KLD reliability mismatch; the other two criteria lead to slightly worse results. LogAPP decoding is only improved if the parameter optimization is performed on basis of the KLD reliability mismatch: the decoder shows a faster convergence; the gain with respect to the number of iterations is between one and three iterations. The other two criteria lead to a bit error rate which is even larger than that without correcting the extrinsic values. ◇

The previous example has given evidence to what was conjectured by theory in Section 3.5.1: iterative decoding with LogAPP constituent decoders is not optimal. Obviously, an improvement is possible, even though the achievable gain may be small. The PCC decoder proposed in [Sor02] may close the gap to optimal decoding.

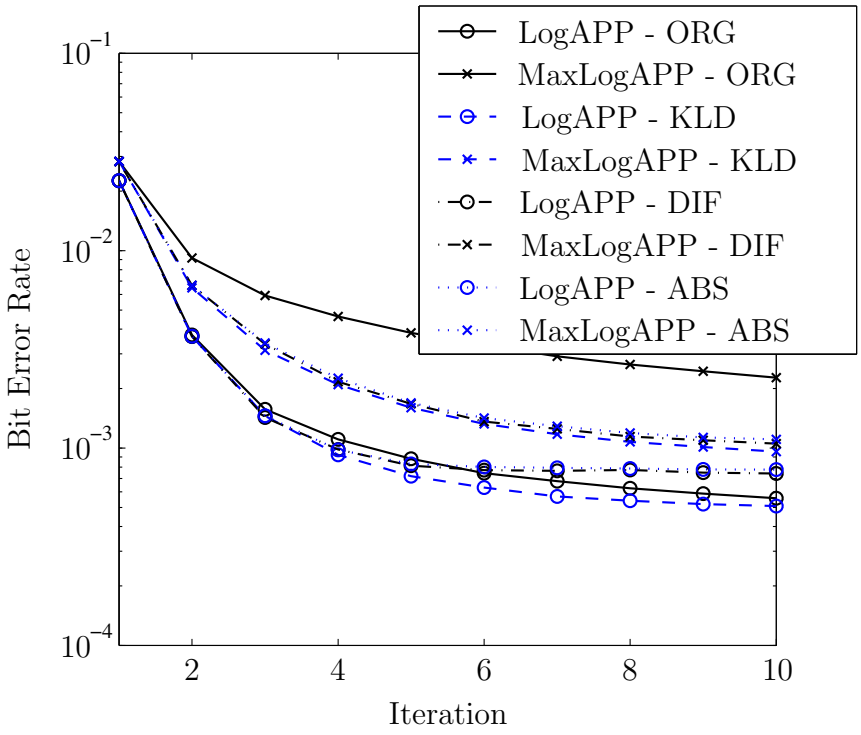


Figure 4.12: Bit error rates versus number of iterations, Example 4.8. PCC with LogAPP and MaxLogAPP decoding, each without (ORG) and with use of correction functions f_1. The correction functions are optimized with respect to the measures of reliability mismatch given in (4.3) (ABS), (4.4) (DIF), and Definition 4.3 (KLD). The values of the optimized parameters are listed in Table 4.1.

4.6 Summary

In this chapter, the notion of reliability information, or to be precise, of reliability values has been discussed. The reliability of soft-values may be measured by simulation and visualized by plotting the LLR versus the soft-value, or equivalently, the reliability value versus the soft-value magnitude. For measuring the difference between soft-values and their LLRs (or equivalently, between soft-value magnitudes and reliability values), the reliability mismatch based on the Kullback-Leibler distance has been introduced. The reliability mismatch may be reduced by post-processing the soft-values using a parameterized correction function. The function parameters can be optimized in such a way that the reliability mismatch after correction becomes minimal. Finally, this concept has been applied to the iterative decoder of a parallel concatenated code. Performance gains in terms of error rates can be achieved for both LogAPP and MaxLogAPP constituent decoders. In particular, the use of LogAPP constituent decoders has been shown by an example to be suboptimal.

Iteration	ORG		KLD		DIF		ABS	
i	$\alpha^{(1)[i]}$	$\alpha^{(2)[i]}$	$\alpha^{(1)[i]}$	$\alpha^{(2)[i]}$	$\alpha^{(1)[i]}$	$\alpha^{(2)[i]}$	$\alpha^{(1)[i]}$	$\alpha^{(2)[i]}$
1	1	1	0.990	0.986	0.991	0.992	0.992	0.992
2	1	1	0.961	0.925	0.945	0.931	0.933	0.902
3	1	1	0.871	0.894	0.771	0.653	0.756	0.746
4	1	1	0.812	0.888	0.784	0.718	0.746	0.723
5	1	1	0.783	0.889	0.637	0.754	0.652	0.714
6	1	1	0.766	0.894	0.599	0.725	0.611	0.732
7	1	1	0.750	0.890	0.651	0.624	0.609	0.728
8	1	1	0.740	0.895	0.668	0.696	0.632	0.707
9	1	1	0.738	0.893	0.655	0.735	0.630	0.690
10	1	1	0.732	0.888	0.601	0.706	0.645	0.698

(a) LogAPP decoding.

Iteration	ORG		KLD		DIF		ABS	
i	$\alpha^{(1)[i]}$	$\alpha^{(2)[i]}$	$\alpha^{(1)[i]}$	$\alpha^{(2)[i]}$	$\alpha^{(1)[i]}$	$\alpha^{(2)[i]}$	$\alpha^{(1)[i]}$	$\alpha^{(2)[i]}$
1	1	1	0.864	0.797	0.870	0.839	0.871	0.831
2	1	1	0.874	0.801	0.867	0.857	0.871	0.863
3	1	1	0.806	0.767	0.747	0.784	0.715	0.804
4	1	1	0.761	0.763	0.690	0.744	0.639	0.747
5	1	1	0.739	0.763	0.594	0.687	0.647	0.652
6	1	1	0.721	0.765	0.703	0.671	0.638	0.741
7	1	1	0.710	0.771	0.585	0.691	0.602	0.694
8	1	1	0.708	0.775	0.631	0.705	0.613	0.689
9	1	1	0.703	0.786	0.597	0.742	0.603	0.723
10	1	1	0.699	0.779	0.625	0.658	0.585	0.733

(b) MaxLogAPP decoding.

Table 4.1: Optimal parameters for the correction functions when LogAPP or MaxLogAPP decoding is applied; Example 4.8. (ORG corresponds to the case where no correction function is applied.)

Chapter 5

Parameter Estimation

A binary-data transmission system that includes a symbol-by-symbol soft-output decoder is typically assessed by two quality parameters: the error probability of the binary symbols and the symbol-wise mutual information between encoder input and decoder output. The first parameter only takes into account the hard outputs of the decoder, i.e., the estimates for the transmitted symbols, whereas the second parameter also takes into account the soft-values, and thus the reliability information. When the soft-outputs of the decoder are reliable according to the definition in Chapter 4, i.e., when they are log-likelihood ratios, the estimation of these two parameters can be simplified and improved, as compared to conventional methods. To be precise, knowledge of the transmitted data is *not* necessary and the estimation variance becomes smaller.

In this chapter, the conventional estimation and the estimation based on reliable soft-outputs is compared by means of their estimation variances. This is done for the symbol error probability, to which special focus has been given, and the symbol-wise mutual information. The investigations are restricted to binary data and to symbol-by-symbol soft-output decoders that compute log-likelihood ratios. The new contributions are as follows:

(a) A general framework for estimating the quality of transmission parameters based on reliable post-decoding soft-values is established.

(b) The estimation based on soft-values is related to the concept of decomposing a channel into subchannels[1].

(c) For the estimation of the symbol error probability and of the symbol-wise mutual information, the conventional method and the new method are compared analytically with respect to their estimation variances.

The presented principle of parameter estimation can easily be generalized; this is discussed at the end of this chapter.

[1] See Chapter 2.

5.1 General Estimation Setup

In this section, the transmission model and the framework for estimation based on soft-values is described. The main idea is to use an expected value conditioned on the soft-value. The precise mathematical description is given below.

The model for estimating symbol-wise transmission quality parameters based on reliable post-decoding soft-values is depicted in Figure 5.1. Binary info symbols $U \in \mathbb{B}$ that are independent and uniformly distributed are transmitted over a binary-input symmetric memoryless channel (BISMC) $U \to L$. The channel outputs $L \in \mathbb{R}$ are assumed to be reliable according to Definition 4.2, i.e.,

$$L(U|L = l) = l.$$

The BISMC $U \to L$ models the transmission chain from encoder input to decoder output. Notice that the magnitude of the soft-value,

$$\Lambda := \mathrm{abs}(L),$$

is a subchannel indicator for the channel $U \to L$, and therefore the subchannels $U \to L|\Lambda = \lambda$ are binary symmetric channels (BSCs) (cf. Chapter 2).

The given model includes coded transmission over memoryless channels, which is the main focus of this thesis, but also transmission over intersymbol-interference channels, etc. These channels are *not* memoryless; but as only symbol-wise transmission parameters, like the symbol error probability or the symbol-wise mutual information, are to be estimated, this channel can be *modeled* as a memoryless channel. The most important presumption is that the decoder (equalizer, detector, etc.) produces reliable post-decoding soft-values for the info symbols, i.e., the soft-values l are assumed to be LLRs.

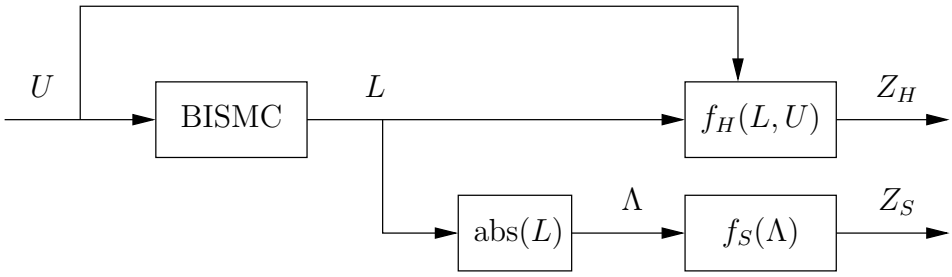


Figure 5.1: Model for estimating symbol-wise parameters.

5.1.1 Description of the Two Methods

The conventional method for parameter estimation is called Method H, and the new method based on reliable soft-values is called[2] Method S. In both methods, one estimation

[2]The samples used in Method S are "softer" than the "hard" samples used in Method H. This is the origin of the names of the two methods.

sample is determined for each transmitted info symbol U. These two samples are random variables, and they are denoted by Z_H for Method H, called the hard sample, and Z_S for Method S, called the soft sample.

Method H A *hard sample* z_H for the parameter to be estimated is determined by a function f_H of the transmitted info symbol u and the soft-value l,

$$z_H := f_H(l, u). \tag{5.1}$$

(The function f_H should be chosen such that the estimation is unbiased.) The mean value of M hard samples is the parameter estimate

$$z_H^{(M)} := \frac{1}{M} \sum_{i=0}^{M-1} z_{H,i}, \tag{5.2}$$

called *hard estimate*.

Method S A *soft sample* z_S for the parameter to be estimated is determined by a function f_S of the reliability value $\lambda := \mathrm{abs}(l)$,

$$z_S := f_S(\lambda), \tag{5.3}$$

where the function f_S is defined as the expectation of function f_H conditioned on the reliability value λ,

$$f_S(\lambda) := \mathrm{E}\big\{ f_H(l, u) \big| \Lambda = \lambda \big\}. \tag{5.4}$$

The mean value of M soft samples is the parameter estimate

$$z_S^{(M)} := \frac{1}{M} \sum_{i=0}^{M-1} z_{S,i}, \tag{5.5}$$

called *soft estimate*.

Due to the definition of function f_S, the soft sample is the conditional expectation of the hard sample,

$$z_S = \mathrm{E}\{ Z_H | \Lambda = \lambda \}.$$

(The definitions of the functions f_H and f_S for the estimation of the symbol error probability and for the symbol-wise mutual information are given in Section 5.2 and Section 5.3, respectively.)

5.1.2 Comparison of the Two Methods

All random processes are assumed to be ergodic. Therefore, we have

$$\begin{aligned} \lim_{M \to \infty} z_H^{(M)} &= \mathrm{E}\{Z_H\}, \\ \lim_{M \to \infty} z_S^{(M)} &= \mathrm{E}\{Z_S\}. \end{aligned}$$

Using the definition of the soft sample from (5.4), we immediately see that the two samples have the same mean value,

$$\mu := \mathrm{E}\{Z_S\} = \mathrm{E}\big\{\mathrm{E}\{Z_H|\Lambda=\lambda\}\big\} = \mathrm{E}\{Z_H\} \tag{5.6}$$

and so the soft estimate $Z_S^{(M)}$ tends to the same value as the hard estimate $Z_H^{(M)}$ for $M \to \infty$. Consequently, any conventional method (called Method H) may be improved using the new method (Method S).

The variance of the hard sample is denoted by $\sigma^2_{Z_H}$, and the variance of the soft sample is denoted by $\sigma^2_{Z_S}$. Since the estimates are mean values of M samples, the variances of the estimates and of the samples are related as

$$\sigma^2_{Z_H^{(M)}} = \frac{1}{M}\sigma^2_{Z_H}, \qquad\qquad \sigma^2_{Z_S^{(M)}} = \frac{1}{M}\sigma^2_{Z_S}.$$

Therefore the variances of the hard and the soft sample are criteria for the quality of the hard and the soft estimate, respectively. In the sequel, the variances of the hard and the soft sample are used to compare Method H and Method S.

The variances of the hard and the soft sample can be written as

$$\begin{aligned} \sigma^2_{Z_H} &= \mathrm{E}\{Z_H^2\} - \mu^2, \\ \sigma^2_{Z_S} &= \mathrm{E}\{Z_S^2\} - \mu^2. \end{aligned} \tag{5.7}$$

Using the definition of the soft sample and Jensen's inequality, we obtain

$$Z_S^2 = (\mathrm{E}\{Z_H|\Lambda=\lambda\})^2 \le \mathrm{E}\{Z_H^2|\Lambda=\lambda\}$$

and thus

$$\mathrm{E}\{Z_S^2\} \le \mathrm{E}\{Z_H^2\}; \tag{5.8}$$

equality holds for constant Z_H. Combining (5.7) and (5.8), we obtain a general relation between the variances of the hard and the soft sample.

Theorem 5.1 (Estimation of Parameter)
Consider the estimation of a parameter, using Method H based on the hard sample Z_H and Method S based on the soft sample Z_S. The sample variances are related as

$$\sigma^2_{Z_H} \ge \sigma^2_{Z_S}.$$

Equality holds if Z_H is constant, i.e., if $\sigma^2_{Z_H} = \sigma^2_{Z_S} = 0$.

Theorem 5.1 proofs that Method S is superior to Method H with respect to the estimation variance. Notice that this relation follows simply from the fact that the soft sample is defined as the conditional expectation of the hard sample.

For comparing the two methods for the estimation of a particular parameter, the hard sample has to be defined in a proper way, and the two estimation variances have to be related. In the following two sections, this is done for the symbol error rate and for the symbol-wise mutual information. Notice the relation between Theorem 2.2 and the estimation of these two parameters using Method S.

5.1.3 Relation to Decomposition into Subchannels

The estimation of a parameter using Method S is closely related to the concept of decomposing a channel into subchannels (cf. Chapter 2). The BISMC $U \rightarrow L$ is decomposed into subchannels by the reliability value Λ, as can easily be seen. Thus, Λ is a subchannel indicator for this channel.

Consider now the soft sample. It is defined as the expectation of the hard sample conditioned on the reliability value λ,

$$z_S = \mathrm{E}\{Z_H | \Lambda = \lambda\}.$$

In other words, the soft sample is the expectation of the hard sample conditioned on the subchannel defined by λ. Following this interpretation, the soft sample z_S represents the parameter, which is to be estimated, for the subchannel $U \rightarrow L|\Lambda = \lambda$.

Notice that the parameter conditioned on the subchannel has to be computed on basis of the subchannel indicator $\Lambda = \lambda$. This is only possible if the soft-value L is reliable according to Definition 4.2, as assumed in the given model.

Further relations between Method S and subchannels are explained in the following two sections.

5.2 Estimation of the Bit Error Rate

The basic method for estimating the error rate of binary symbols, called bit error rate (BER) in the sequel, based on soft-values was first published in [Loe94], and was then independently re-invented and further investigated in [HLS00]. The estimation variances of the conventional method and the new method were analyzed in [LH03]. In the sequel, the results from [LH03] are embedded in the framework given in the previous section.

The soft-value l is assumed to be an LLR, and so the estimate $\hat{u} \in \mathbb{B}$ for the transmitted info symbol $u \in \mathbb{B}$ that minimizes the BER is given by the decision

$$\hat{u} := \begin{cases} +1 & \text{if } \mathrm{sgn}(l) > 0, \\ -1 & \text{if } \mathrm{sgn}(l) < 0, \\ \text{randomly chosen from } \mathbb{B} & \text{if } \mathrm{sgn}(l) = 0. \end{cases}$$

This decision is erroneous if $u \neq \hat{u}$.

The BER is formally defined as

$$P_b := \Pr(U \neq \hat{U}).$$

The conventional method for estimating the BER is to count the number of errors and divide this value by the number of transmitted info symbols. The following formal description of this method, called Method H, enables a straight-forward analysis of the estimation variance, and in particular, the extension and comparison to Method S.

Hard and Soft BER Sample

The *hard BER sample* z_H indicates whether an error occurred or not. It is defined as

$$z_H := \begin{cases} 0 & \text{if } u = \operatorname{sgn}(l), \\ 1 & \text{if } u = -\operatorname{sgn}(l), \\ \text{randomly chosen from } \{0,1\} & \text{if } \operatorname{sgn}(l) = 0. \end{cases} \tag{5.9}$$

Notice that $z_H \in \{0,1\}$. The hard BER sample z_H is a function of both the transmitted info symbol u and the soft-value l, as given in (5.1), where only the sign of the soft-value is used. Since it may also be written as

$$z_H = \Pr(U \neq \hat{U} | U = u, \hat{U} = \hat{u}), \tag{5.10}$$

we obtain

$$\mathrm{E}\{Z_H\} = \mathrm{E}\big\{\Pr(U \neq \hat{U} | U = u, \hat{U} = \hat{u})\big\} = \Pr(U \neq \hat{U}),$$

and the estimation is unbiased[3].

The *soft BER sample* is defined as

$$z_S := \mathrm{E}\{Z_H | \Lambda = \lambda\} = \Pr(U \neq \hat{U} | \Lambda = \lambda), \tag{5.11}$$

according to (5.4); the last expression follows from substituting (5.10). The soft BER sample indicates the probability that the symbol estimate $\hat{u}$ is wrong conditioned on the reliability value λ. Using the conversion from LLRs to probabilities given in Appendix C, the soft BER sample can be computed [HLS00] as

$$z_S = \frac{1}{1 + e^{\lambda}}, \tag{5.12}$$

which agrees with (5.4). Notice that $z_S \in [0, \frac{1}{2}]$, whereas $z_H \in \{0,1\}$. The soft BER sample relies only on the magnitude of the soft-value; knowledge of the transmitted info symbol is not necessary.

Notice that the soft BER sample z_S is identical to the crossover probability of the subchannel $U \to L | \Lambda = \lambda$; so the random variable Z_H is identical to the error probability indicator $\mathcal{E}$, as defined in Chapter 2. Therefore, estimation based on the reliability value λ may be seen as an application of the concept of decomposing a channel into binary symmetric subchannels (cf. Theorem 2.2).

The mean values of the hard and the soft BER sample are equal, as shown in the previous section,

$$\mu := \mathrm{E}\{Z_H\} = \mathrm{E}\{Z_S\} = P_b,$$

and they are equal to the bit error rate P_b.

The comparison of the variances of the hard and the soft BER sample is based on their probability distributions. The hard BER sample is distributed as

$$p_{Z_H}(z_H) = \begin{cases} 1 - \mu & \text{for } z_H = 0, \\ \mu & \text{for } z_H = 1. \end{cases} \tag{5.13}$$

[3]Which is not surprising.

Given a soft BER sample z_S, the conditional distribution of the hard BER sample can be written as

$$p_{Z_H|Z_S}(z_H|z_S) = \begin{cases} 1 - z_S & \text{for } z_H = 0, \\ z_S & \text{for } z_H = 1. \end{cases} \tag{5.14}$$

Regarding the distribution $p_{Z_S}(z_S)$ of the soft BER sample, only the mean value μ of Z_S can assumed to be known; other statements that are independent of the actual distribution cannot be made.

Statistical Dependence

The hard and the soft BER sample are not statistically independent. For measuring this statistical dependence, we compute the mutual information between the two samples.

Let $h(p) = -p \operatorname{ld} p - (1-p) \operatorname{ld}(1-p) \in [0,1]$, $p \in [0,1]$, denote the binary entropy function (cf. Appendix C). Then we have

$$\begin{aligned} I(Z_H; Z_S) &= H(Z_H) - H(Z_H|Z_S) \\ &= H(Z_H) - \mathop{\mathrm{E}}_{z_S}\{H(Z_H|Z_S = z_S)\} \\ &= h(\mu) - \mathrm{E}\{h(Z_S)\}. \end{aligned}$$

In the last line, we have applied (5.14) and

$$H(Z_H|Z_S{=}z_S) = h\big(p_{Z_H|Z_S}(1|z_S)\big) = h(z_S).$$

The binary entropy function can be bounded as $h(z_S) \geq 2z_S$, and so we obtain

$$I(Z_H; Z_S) \leq h(\mu) - 2\mu. \tag{5.15}$$

Therefore, the hard BER sample Z_H and the soft BER sample Z_S become statistically independent when the BER approaches 0 or 1/2, and mutual information between Z_H and Z_S is upper-bounded by $h(\mu) - 2\mu$. Notice that these are general statements depending only on the mean value $\mu = P_b$.

Comparison

The variance of the hard BER sample Z_H can be written as

$$\sigma^2_{Z_H} = \mathrm{E}\{Z_H^2\} - \mu^2 = \mathrm{E}\{Z_H\} - \mu^2 = \mu(1-\mu), \tag{5.16}$$

where the identity $Z_H^2 = Z_H$ is applied. (Notice that $Z_H \in \{0,1\}$.) The variance of the soft BER sample Z_S can be written as

$$\sigma^2_{Z_S} = \mathrm{E}\{Z_S^2\} - \mu^2; \tag{5.17}$$

further simplification is not possible. Two lower bounds on the ratio of these variances are derived in the sequel.

From $Z_S \in [0, \frac{1}{2}]$, it follows that $Z_S^2 \leq \frac{1}{2} Z_S$, and thus

$$\mathrm{E}\{Z_S^2\} \leq \tfrac{1}{2}\,\mathrm{E}\{Z_S\}. \tag{5.18}$$

This inequality is the starting point for the two bounds. Equality holds for $Z_S = \frac{1}{2}$ and for $Z_S = 0$, which corresponds to $P_b = \frac{1}{2}$ and $P_b = 0$, respectively. In each case, Z_S is constant, and thus its variance is zero.

For deriving the first bound, we write the left hand side of (5.18) as

$$\mathrm{E}\{Z_S^2\} = \sigma_{Z_S}^2 + \mu^2$$

and the right hand side as

$$\tfrac{1}{2}\,\mathrm{E}\{Z_S\} = \tfrac{1}{2}\,\mathrm{E}\{Z_H\} = \tfrac{1}{2}\,\mathrm{E}\{Z_H^2\} = \tfrac{1}{2}(\sigma_{Z_H}^2 + \mu^2).$$

Substituting these two equalities into (5.18) yields

$$\begin{aligned} & \sigma_{Z_S}^2 + \mu^2 && \leq && \tfrac{1}{2}(\sigma_{Z_H}^2 + \mu^2) \\ \Leftrightarrow \quad & \sigma_{Z_S}^2 && \leq && \tfrac{1}{2}\sigma_{Z_H}^2 - \tfrac{1}{2}\mu^2 \\ \Rightarrow \quad & \sigma_{Z_S}^2 && \leq && \tfrac{1}{2}\sigma_{Z_H}^2; \end{aligned} \tag{5.19}$$

equality holds in the last line if and only if $\mu = P_b = 0$. Thus, we have the first bound.

Theorem 5.2 (Estimation of Bit Error Rate)
Consider the estimation of the bit error rate (BER), using Method H based on the hard BER samples Z_H defined in (5.9) *and Method S based on the soft BER samples Z_S defined in* (5.11)*. The ratio of the variances is lower-bounded as*

$$\frac{\sigma_{Z_H}^2}{\sigma_{Z_S}^2} \geq 2.$$

For deriving the second bound, we write the left hand side of (5.18) as before, and we substitute $\mathrm{E}\{Z_S\} = \mu$ on the right hand side. Thus, we obtain

$$\begin{aligned} & \sigma_{Z_S}^2 + \mu^2 && \leq && \frac{\mu}{2} \\ \Leftrightarrow \quad & \sigma_{Z_S}^2 && \leq && \frac{\mu(1-2\mu)}{2}. \end{aligned}$$

Using this inequality and (5.16), the ratio of the variances can be written as

$$\frac{\sigma_{Z_H}^2}{\sigma_{Z_S}^2} \geq \frac{2\mu(1-\mu)}{\mu(1-2\mu)} = \frac{2-2\mu}{1-2\mu},$$

and we have the second bound.

Theorem 5.3 (Estimation of Bit Error Rate)
Consider the estimation of the bit error rate (BER), using Method H based on the hard BER samples Z_H defined in (5.9) *and Method S based on the soft BER samples Z_S defined in* (5.11)*. With P_b denoting the BER, the ratio of the variances is lower-bounded as*

$$\frac{\sigma_{Z_H}^2}{\sigma_{Z_S}^2} \geq \frac{2-2P_b}{1-2P_b}. \tag{5.20}$$

This bound is tighter than the first bound for large BERs, and it becomes equal to the first bound when the BER tends to zero.

Example 5.1
Independent and uniformly distributed info symbols are encoded by a convolutional encoder of rate 1/2 and memory length m; the code symbols are transmitted over a binary-input AWGN channel. The ratio of variances $\sigma^2_{Z_H}/\sigma^2_{Z_S}$ found by simulation (solid lines) and the lower bounds according to Theorem 5.3 (dashed lines) are plotted versus the SNR in Figure 5.2. For low SNR, both the actual ratio $\sigma^2_{Z_H}/\sigma^2_{Z_S}$ and the bound are increasing. For higher SNR, the bound tends to 2, which is equal to the bound from Theorem 5.2, and the simulated value tends to about 4. Thus, the actual advantage of Method S is even larger than predicted by Theorem 5.2 and Theorem 5.3. ◇

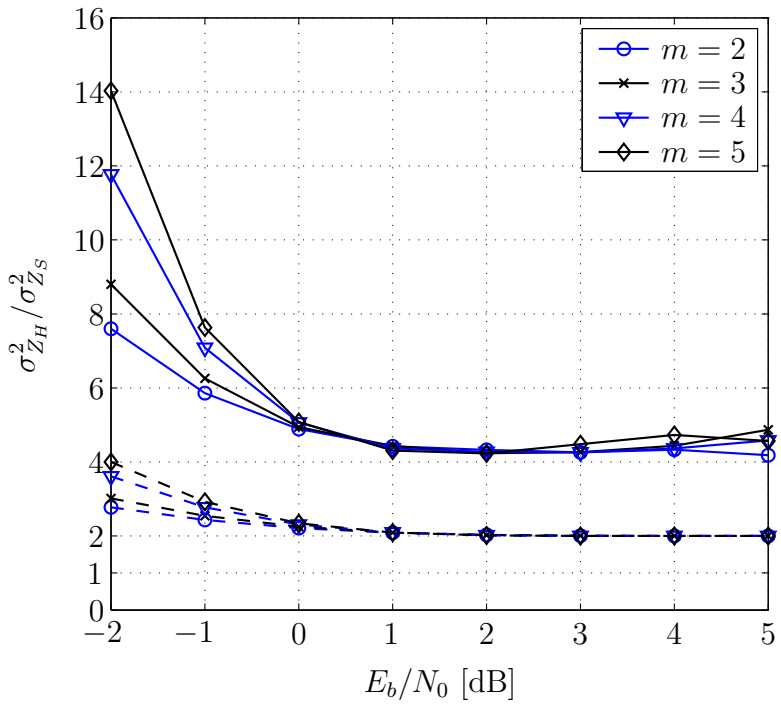


Figure 5.2: Ratio of variances $\sigma^2_{Z_H}/\sigma^2_{Z_S}$ determined by simulation (solid lines) and lower bound according to Theorem 5.3 (dashed lines) versus E_b/N_0; convolutional codes of rate 1/2 and several memory lengths m, Example 5.1.

The variances of the hard and the soft BER sample show the following behavior when the BER tends to zero or $\frac{1}{2}$. For $P_b \to 0$, we have $Z_H \to 0$ and $Z_S \to 0$, and thus $\sigma^2_{Z_H} \to 0$ and $\sigma^2_{Z_S} \to 0$. The ratio of the variances depends on how quickly the two variances tend to zero.

For $P_b \to \frac{1}{2}$, we have for the hard BER sample $p_{Z_H}(0) \to \frac{1}{2}$ and $p_{Z_H}(1) \to \frac{1}{2}$, and thus the maximal variance $\sigma^2_{Z_H} \to \frac{1}{4}$, according to (5.16). The soft BER sample becomes constant, $Z_S \to \frac{1}{2}$, and thus $\sigma^2_{Z_S} \to 0$. Consequently, the ratio $\sigma^2_{Z_H}/\sigma^2_{Z_S}$ tends to infinity.

Theorem 5.2 and Theorem 5.3 prove that the hard BER sample has always a larger variance than the soft BER sample, even for the extreme cases $P_b = 0$ and $P_b = \frac{1}{2}$. Thus, Method S is always better than Method H with respect to the estimation variance. Defining the precision of an estimate Z based on M samples as the relative standard deviation,

$$\frac{\sigma_Z^{(M)}}{\mu} = \frac{\sqrt{\sigma_Z^2/M}}{\mu},$$

the advantage of Method S can be formulated in the following two equivalent ways:

(a) For achieving a required precision of the BER estimate, Method S needs less than half the number of samples.

(b) Given a fixed number of samples, the precision achieved by Method S is larger by a factor of a least $\sqrt{2}$.

Optimal Linear Combination

Method H uses only the sign of the soft-value and Method S uses only the magnitude of the soft-value. This gives rise to the question whether the two methods can be combined to obtain a BER estimate that is even better than the soft BER estimate. In the sequel, the optimal *linear* combination of the hard and the soft BER sample is considered.

The combined BER sample is defined as

$$z := \alpha z_H + (1-\alpha) z_S \tag{5.21}$$

with weighting factor $\alpha \in [0,1]$. Due to its definition, the combined BER Z is unbiased, i.e., $\mathrm{E}\{Z\} = \mu = P_b$. The weighting factor is now determined such that the variance of Z becomes minimal.

The variance of Z can be written as

$$\begin{aligned}
\sigma_Z^2 &= \mathrm{E}\{(Z-\mu)^2\} \\
&= \mathrm{E}\{(\alpha Z_H + (1-\alpha) Z_S - \mu)^2\} \\
&= \mathrm{E}\{(\alpha(Z_H-\mu) + (1-\alpha)(Z_S-\mu))^2\} \\
&= \alpha^2 \sigma_{Z_H}^2 + (1-\alpha)^2 \sigma_{Z_S}^2 + 2\alpha(1-\alpha)\sigma_{Z_H Z_S}^2,
\end{aligned} \tag{5.22}$$

where

$$\sigma_{Z_H Z_S}^2 := \mathrm{E}\big\{(Z_H-\mu)(Z_S-\mu)\big\}$$

denotes the covariance of Z_H and Z_S. Applying (5.14), the conditional expectation of Z_H can be evaluated as

$$\mathrm{E}\{Z_H | Z_S = z_S\} = \sum_{z_H \in \{0,1\}} p_{Z_H|Z_S}(z_H|z_S) \cdot z_h = z_s.$$

Thus, the covariance can be expressed as

$$\begin{aligned}
\sigma_{Z_H Z_S}^2 &= \mathrm{E}\big\{(Z_H-\mu)(Z_S-\mu)\big\} = \mathrm{E}\{Z_H Z_S\} - \mu^2 \\
&= \mathrm{E}\big\{Z_S \cdot \mathrm{E}\{Z_H|Z_S\}\big\} - \mu^2 \\
&= \mathrm{E}\{Z_S^2\} - \mu^2 \\
&= \sigma_{Z_S}^2.
\end{aligned}$$

Substituting this equality into (5.22), the variance of Z can be written as

$$\begin{aligned}\sigma_Z^2 &= \alpha^2\sigma_{Z_H}^2 + (1-\alpha)^2\sigma_{Z_S}^2 + 2\alpha(1-\alpha)\sigma_{Z_S}^2 \\ &= \alpha^2\sigma_{Z_H}^2 + (1-\alpha^2)\sigma_{Z_S}^2. \end{aligned} \tag{5.23}$$

Note that this expression contains only the variances of the hard and the soft BER sample and the weighting factor α.

The extremum is found by evaluating

$$\frac{d}{d\alpha}\sigma_Z^2 = 2\alpha\sigma_{Z_H}^2 - 2\alpha\sigma_{Z_S}^2 \stackrel{!}{=} 0.$$

The single solution is $\alpha = 0$, because $\sigma_{Z_H}^2 > \sigma_{Z_S}^2$ according to Theorem 5.2. For the same reason, the second derivative is strictly positive,

$$\frac{d^2}{d\alpha^2}\sigma_Z^2 = 2\sigma_{Z_H}^2 - 2\sigma_{Z_S}^2 > 0,$$

and $\alpha = 0$ minimizes the variance of Z.

Thus we have the final result: the optimal linear combination of the hard and the soft BER sample (optimal with respect to minimal estimation variance) consists only of the soft BER sample. In other words: Method S cannot be improved by linearly combining it with Method H.

5.3 Estimation of the Mutual Information

The symbol-wise mutual information between info symbols and soft-values may be estimated in a similar way as the bit error probability [LHG04]. This mutual information (MI) is defined as

$$I(U;L) = \mathrm{E}\big\{I(U=u;L=l)\big\} = \mathrm{E}\Big\{\mathrm{ld}\,\frac{p_{U|L}(u|l)}{p_U(u)}\Big\}.$$

The conventional method[4], called Method H, uses both the transmitted info symbol u and the soft-value l provided by the decoder. On the other hand, $I(U;L)$ may also be estimated using only the reliability value λ; this method is called Method S. Notice that the soft-value is assumed to be an LLR.

Hard and Soft MI Sample

The hard sample for estimating the mutual information, for short, the *hard MI sample* z_H, is defined as

$$z_H := I(U=u;L=l) = \mathrm{ld}\,\frac{p_{U|L}(u|l)}{p_U(u)}. \tag{5.24}$$

[4]Alternatively, the probability density functions $p_{L|U}(l|u)$ may be determined by histogram measurements; applying Bayes' rule, the distribution $p_{U|L}(u|l)$ and thus $I(U;L)$ may be computed. However, when the soft-values l are LLRs, the histogram measurement is obviously not necessary.

As the soft-value l is an LLR, it can be used to express the probability density function $p_{U|L}(u|l)$ (cf. Appendix C). Applying this and $p_U(u) = \frac{1}{2}$, the hard MI sample can be computed as

$$z_H = \mathrm{ld}\,\frac{2}{1+e^{-lu}}. \tag{5.25}$$

Notice that $z_H \in (-\infty, 1]$. The hard MI sample is a function of both the transmitted info symbol u and the soft-value l, as given in (5.1). It is obvious that

$$\mathrm{E}\{Z_H\} = I(U;L),$$

and so this estimation is unbiased[5].

The *soft MI sample* is defined as

$$z_S := \mathrm{E}\big\{I(U=u;L=l)\big|\Lambda=\lambda\big\} = I(U;L|\Lambda=\lambda), \tag{5.26}$$

according to (5.4). The condition $\Lambda = \lambda$ defines a subchannel, and the crossover probability of this subchannel is given by

$$\epsilon = \frac{1}{1+e^{\lambda}}.$$

Therefore, the soft MI sample can be computed as

$$\begin{aligned} z_S &= I(U;L|\Lambda=\lambda) = 1 - h(\epsilon) \\ &= \frac{1}{1+e^{\lambda}}\,\mathrm{ld}\,\frac{2}{1+e^{\lambda}} + \frac{1}{1+e^{-\lambda}}\,\mathrm{ld}\,\frac{2}{1+e^{-\lambda}}, \end{aligned} \tag{5.27}$$

where h denotes the binary entropy function (cf. Appendix C) [LHG04]. Notice that z_S is only a function of the reliability value λ, as required by (5.4). As opposed to the hard MI sample, the domain of the soft MI sample is limited: $z_S \in [0,1]$.

Notice that the soft MI sample z_S is identical to the subchannel mutual information of the subchannel $U \to L|\Lambda=\lambda$; so the random variable Z_S is identical to the mutual information indicator J, as defined in Chapter 2. Therefore, estimation based on the reliability value λ may be seen as an application of the concept of decomposing a channel into binary symmetric subchannels (cf. Theorem 2.2).

The mean values of the hard and the soft MI sample are equal, as shown in the previous section,

$$\mu := \mathrm{E}\{Z_H\} = \mathrm{E}\{Z_S\} = I(U;L),$$

and they are equal to the symbol-wise mutual information $I(U;L)$.

Comparison

The variances of the hard and the soft MI sample can be written as

$$\begin{aligned} \sigma^2_{Z_H} &= \mathrm{E}\{Z_H^2\} - \mu^2, \\ \sigma^2_{Z_S} &= \mathrm{E}\{Z_S^2\} - \mu^2. \end{aligned} \tag{5.28}$$

[5]Which is again not surprising.

Using the definition of the soft MI sample and Jensen's inequality, we obtain

$$Z_S^2 = (\mathrm{E}\{Z_H|\Lambda = \lambda\})^2 \leq \mathrm{E}\{Z_H^2|\Lambda = \lambda\}$$

and thus

$$\mathrm{E}\{Z_S^2\} \leq \mathrm{E}\{Z_H^2\}, \tag{5.29}$$

where equality holds for constant Z_H. Notice that this is the case if and only if $I(U;L) = 0$.

Combining (5.28) and (5.29), we find the relation between the variances of the hard and the soft MI sample.

Theorem 5.4 (Estimation of Mutual Information)
Consider the estimation of the symbol-wise mutual information (MI) between info symbols U and reliable soft-values L (soft-values that are LLRs), using Method H based on the hard MI samples Z_H defined in (5.24) *and Method S based on the soft MI samples Z_S defined in* (5.26)*. The variances are related as*

$$\sigma_{Z_H}^2 \geq \sigma_{Z_S}^2.$$

Equality holds if $I(U;L) = 0$.

Theorem 5.4 proves that Method S is superior to Method H with respect to the estimation variance. The ratio of the two variances depends on the probability density function $p_{L|U}(l|u)$. Thus, the advantage of using Method S, i.e., the gain due to the decreased estimation variance, depends on the system under consideration.

A direct application is the computation of information transfer functions, like information processing characteristics and extrinsic information transfer characteristics (cf. Section 3.3). Method S allows to do this in a simple, efficient, and convenient way [LHG04].

5.4 Further Applications

This thesis focuses on the transmission of independent and uniformly distributed binary data. Thus, the error rate of the binary info symbols and the symbol-wise mutual information between the info symbols and the post-decoding soft-values are the most interesting criteria for the transmission quality.

Consider now a more general system: redundant source signals are quantized and the quantization indices are transmitted (coded or uncoded) over a communication channel; a symbol-by-symbol soft-output decoder computes soft-values for each quantization index, and based on these soft-values, the source signals are reconstructed. Relevant quality parameters are then the error probability of the quantization indices and the SNR of the reconstructed source signals. Both parameters can be estimated using Method H and Method S, and the two methods can be compared similarly to the estimation of the BER and the mutual information. Details can be found in [TL04].

This shows that the presented methods are quite general and may be applied to estimate other parameters in a similar way. The only prerequisite are reliable post-decoding soft-values, e.g., a-posteriori LLRs or a-posteriori probabilities.

5.5 Summary

In this chapter, the estimation of transmission quality parameters based on reliable post-decoding soft-values has been discussed. Two basic methods have been distinguished: Method H and Method S. Method H corresponds to the conventional method; "hard" samples are determined using the transmitted info symbol and the corresponding soft-value, and the estimate is the mean of the hard samples. In contrast, Method S is based on "soft" samples that are the expectations of the hard samples conditioned on the reliability values; i.e., each soft sample is only a function of a reliability value. The estimate is again the mean of the soft samples.

For the estimation of the symbol error probability and the estimation of the symbol-wise mutual information, the relationship between the variances of the hard and the soft samples has been determined analytically. In both cases, Method S outperforms Method H with respect to the estimation variance. Finally, possible generalizations of this estimation principle have been discussed.

Chapter 6

Information Combining

Channel decoding is based on combining information. Each noisy observation of a code symbol carries information about this code symbol as well as information about other code symbols due to the code constraints. A channel decoder collects and combines this information, e.g., to compute post-decoding soft-values that carry the overall information available on info or code symbols. A decoder may thus be interpreted as an information processor and may be characterized by information transfer functions (cf. Section 3.3). The information available after decoding can be used completely by subsequent processing stages only if the decoder computes reliable[1] soft-values (cf. Chapter 4). Therefore, information transfer functions characterize the capability of a decoder to process mutual information, assuming that all available reliability information is fully exploited (cf. Section 3.3).

This chapter addresses the combining of information by decoders from an information theory point of view: *information combining* is used in the sense of combining values of mutual information. The term "information combining" was coined in [HHJF01, HHFJ02, Hue04], where this concept was applied to analyze and design concatenated coding schemes.

The combined mutual information depends not only on the values of mutual information that are combined, but also on the channel models. However, when the channels are assumed to have certain properties, bounds on the combined mutual information can be determined. This concept of bounding combined information was first introduced for the case of two binary-input symmetric memoryless channels (BISMCs) with the same input in [LHHH03].

In this chapter, this concept is extended to the case of multiple BISMCs with an equality or a parity-check constraint on their inputs. Using these results, information transfer functions for repetition codes, single parity check codes, and the accumulator are bounded. Furthermore, these bounds are applied to determine decoding thresholds for low-density parity-check codes using the EXIT chart method. A similar method was developed in [SSSZ03, SSSZ05] motivated by [LHHH03]. The *bounds on information combining* provide new insights into the information processing capabilities of decoders and into the convergence behavior of iterative decoders, both from an information theory

[1] See Definition 4.2.

point of view.

The new contributions are as follows:

(a) The concept of bounding combined information is introduced. The derivations of the bounds are based on the decomposition of channels into subchannels[2].

(b) Bounds on the extrinsic information for single parity check codes, on the extrinsic and the complete information for repetition codes, and bounds on the complete information based on intrinsic and extrinsic information are derived.

(c) The concept of information profiles is applied to explain the bounds on information combining.

(d) Bounds on information transfer functions for single parity check codes, repetition codes, and the accumulator are presented.

(e) The bounds on EXIT functions are applied in the EXIT chart method to determine bounds on the decoding threshold for low-density parity-check codes. These bounds hold for all communication channels that are BISMCs; in particular, the Gaussian assumption, commonly used in the EXIT chart method, is not required.

Parts of this work have been published in [LSH04, LHH04b, LHHH05a, LHHH05b]. Motivated by [LHHH03], similar results were independently derived in [SSSZ03, SSSZ05].

The derivations in this chapter rely on the concept of decomposing BISMCs into binary symmetric subchannels, as presented in Chapter 2. Furthermore, the decoding model, the information transfer functions, and the descriptions of concatenated codes and low-density parity-check codes, provided in Chapter 3, are employed.

6.1 Decoding Model and Notation

The processing of mutual information is analyzed based on the notation and the decoding model introduced in Section 3.1 and Section 3.2. (Notice especially the notation for binary symbols: $a \in \mathbb{B} \Leftrightarrow \breve{a} \in \mathbb{F}_2$.) In the following section, this decoding model is expanded to include more general cases.

Decoding Model

Consider a *systematic binary linear code* $\mathcal{C}$ of length N with equiprobable code words $\boldsymbol{x} = [x_0, x_1, \ldots, x_{N-1}] \in \mathbb{B}^N$. The code symbols are transmitted over independent *BISMCs*, denoted by $X_i \rightarrow Y_i$, with the mutual information $I(X_i; Y_i)$, $i = 0, 1, \ldots, N-1$. The word comprising all received values is denoted by $\boldsymbol{y} = [y_0, y_1, \ldots, y_{N-1}] \in \mathbb{R}^N$. A symbol-by-symbol soft-output decoder computes complete post-decoding values v_i and extrinsic post-decoding values w_i for the code symbols:

$$\begin{aligned} v_i &:= L(X_i|\boldsymbol{Y} = \boldsymbol{y}), \\ w_i &:= L(X_i|\boldsymbol{Y}_{\setminus i} = \boldsymbol{y}_{\setminus i}), \end{aligned}$$

[2] See Chapter 2.

$i = 0, 1, \ldots, N-1$.

From an information theory point of view, we have N parallel channels that are coupled by constraints on their inputs. These constraints are given by the code constraints of code $\mathcal{C}$, as the channel inputs are the code symbols. The parallel channels and the input constraints are depicted in Fig. 6.1.

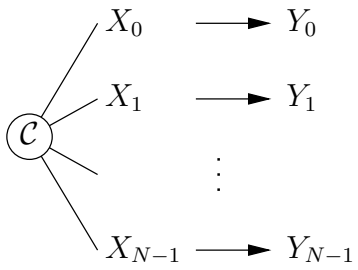


Figure 6.1: Parallel channels with inputs subject to the code constraints of a code $\mathcal{C}$.

All channels are assumed to be BISMCs (cf. Chapter 2). In this chapter, two special BISMCs are of importance, namely the binary symmetric channel (BSC) and the binary erasure channel (BEC). We use the following representations[3]:

BSC Channel $X \to Y$ with inputs $X \in \mathbb{B}$ and outputs $Y \in \mathbb{B}$. The crossover probability is given by $\epsilon = \Pr(Y \neq x | X = x) \leq \frac{1}{2}$ for $x \in \mathbb{B}$. (This channel is depicted in Fig. 2.1.)

BEC Channel $X \to Y$ with inputs $X \in \mathbb{B}$ and outputs $Y \in \{-1, 0, +1\}$, where $Y = 0$ represents the erasure. The erasure probability is given by $\delta = \Pr(Y = 0 | X = x)$ for $x \in \mathbb{B}$. (This channel is depicted in Fig. 2.2.)

The decoding model given here and the decoding model given in Section 3.2 differ in one important aspect: In the decoding model from Section 3.2, all info symbols are transmitted over the same (info-symbol) channel and all code symbols are transmitted over the same (code-symbol) channel. As opposed to that, in the decoding model described above, each code symbol has "its own" channel. Thus, the channels and their mutual information values may be different for each code symbol. Consequently, info and code symbols need not be strictly distinguished, and the use of systematic code symbols is sufficient. The results based on this extended decoding model are thus more general.

Values of Mutual Information

Since v_i are a-posteriori LLRs and w_i are extrinsic a-posteriori LLRs, both soft-values are sufficient statistics of the channel outputs, and we have

$$\begin{aligned} I(X_i; V_i) &= I(X_i; \boldsymbol{Y}), \\ I(X_i; W_i) &= I(X_i; \boldsymbol{Y}_{\setminus i}) \end{aligned}$$

[3]The derivations are valid for any input alphabet and any output alphabet.

for $i = 0, 1, \ldots, N-1$. Thus, the mutual information between a code symbol and its soft-value may equivalently be written as the mutual information between this code symbol and the corresponding channel outputs. Correspondingly, the information processing by the given soft-output decoder can be analyzed without an explicit decoding operation; for this reason, the soft-output decoder is omitted in Fig. 6.1.

Three kinds of mutual information values may be associated with each code symbol X_i [HH02]: the intrinsic information, the extrinsic information and the complete information.

Intrinsic Information The intrinsic information on code symbol X_i,

$$I_{\mathsf{int},i} := I(X_i; Y_i), \tag{6.1}$$

is defined as the mutual information between X_i and its noisy observation Y_i. The intrinsic information is equal to the mutual information of the channel over which the code symbol is transmitted.

Extrinsic Information The extrinsic information on code symbol X_i,

$$I_{\mathsf{ext},i} := I(X_i; \boldsymbol{Y}_{\setminus i}), \tag{6.2}$$

is defined as the mutual information between X_i and the noisy observations of all other code symbols, $\boldsymbol{Y}_{\setminus i}$. This kind of mutual information follows the definition of extrinsic probabilities or extrinsic log-likelihood ratios used in iterative decoding [BG96, HOP96].

Complete Information The complete information on code symbol X_i,

$$I_{\mathsf{cmp},i} := I(X_i; \boldsymbol{Y}), \tag{6.3}$$

is defined as the mutual information between X_i and the observations of all code symbols, $\boldsymbol{Y}$. The complete information may be formed by combining the intrinsic information and the extrinsic information [HH02], where the combining operator depends on the channel models (cf. Section 6.2.2).

Notice that the intrinsic, the extrinsic, and the complete information are values of mutual information. For convenience, we abbreviate "the mutual information between code symbol X_i and observations ..." by "the intrinsic (extrinsic, complete) information on code symbol X_i", whenever this is possible without causing ambiguity.

Average Symbol-Wise Information The average symbol-wise extrinsic and the average symbol-wise complete information, averaged over info or code symbols, are denoted by

$$I_{\mathsf{ext}} := \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} I_{\mathsf{ext},i}, \tag{6.4}$$

$$I_{\mathsf{cmp}} := \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} I_{\mathsf{cmp},i}, \tag{6.5}$$

respectively, where $\mathcal{I}$ denotes the index set of the info or code symbols. This complies with Definition 3.6, where the labeling of the mutual information values with "U" or "X" is omitted for brevity; the meanings become clear from the context.

6.2 Bounds on Mutual Information

Code symbols are subject to code constraints. Therefore, the extrinsic information on a particular code symbol is a combination of the values of intrinsic information on the other code symbols. Similarly, the complete information on a code symbol is a combination of the intrinsic information and the extrinsic information on this code symbol.

The extrinsic information and the complete information can be computed precisely when the channel models are given. This is done to determine information transfer functions (cf. Section 3.3). On the other hand, if only the intrinsic information on each code symbol is known, and not the underlying channel model, bounds on the extrinsic and on the complete information can still be determined.

This section deals with bounds on the extrinsic information for single parity check codes, with bounds on the complete and the extrinsic information for repetition codes, and with bounds on the combining of intrinsic and extrinsic information. The main results are stated in Theorem 6.1, Theorem 6.2, Theorem 6.3, and Theorem 6.4. These theorems generalize the work presented in [LHHH03, LHHH05a, LHH04b] and have been published in part in [LHHH05b]. The proofs are based on the concept of decomposing a BISMC into BSCs and the concept of information profiles of channels (cf. Chapter 2).

Motivated by [HH03,LHHH03], results similar to those presented in this thesis were independently derived in [SSSZ03,SSSZ05]: a more general notion of information combining is introduced (though results are presented only for optimal combining) and the extremes of information combining are determined with respect to each individual channel; the proofs are based on Mrs. Gerber's Lemma [WZ73] and its extension in [CSS89].

6.2.1 Single Parity Check Codes

Consider a single parity check (SPC) code of length N, which is defined by the constraint

$$\breve{X}_0 \oplus \breve{X}_1 \oplus \cdots \oplus \breve{X}_{N-1} = 0 \tag{6.6}$$

on the code symbols $\breve{X}_i \in \mathbb{F}_2$. The code constraint and the transmission channels $X_i \to Y_i$ are shown in Fig. 6.2 for $N = 4$. In the following derivation, we consider only the extrinsic information on code symbol X_0. Due to the symmetric structure of the code, the expressions for the other code symbols are similar.

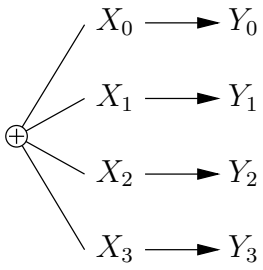


Figure 6.2: Single parity check code of length $N = 4$.

We discuss first the two cases where all channels are BECs and where all channels are BSCs. The BEC case turns out to be a simple combinatorial problem, whereas the BSC case is more involved. These cases are then shown to lead to the minimal and to the maximal extrinsic information.

Binary Erasure Channels

When the channels are all BECs, the value of code symbol X_0 can be recovered with certainty if no erasure has occurred, i.e., if $Y_i \neq 0$ for all $i = 1, 2, \ldots, N-1$. (An erasure corresponds to $Y_i = 0$.) This happens with probability $(1-\delta_1)(1-\delta_2)\cdots(1-\delta_{N-1})$. If we have one or more erasures, no extrinsic information on code symbol X_0 is available. Using (2.7) and the above probability, it can easily be seen that

$$I_{\mathsf{ext},0}^{\mathsf{BEC}} = I_{\mathsf{int},1} I_{\mathsf{int},2} \cdots I_{\mathsf{int},N-1}. \tag{6.7}$$

Binary Symmetric Channels

For the case where the channels are all BSCs, the following function is introduced.

Definition 6.1 (Binary Information Function for Serial Concatenation) Let $I_1, I_2, \ldots, I_n \in [0,1]$, $n \geq 1$. We define the binary information function for serial concatenation for $n = 1$ as

$$f_1^{\mathsf{ser}}(I_1) := I_1,$$

for $n = 2$ as

$$f_2^{\mathsf{ser}}(I_1, I_2) := 1 - h\big((1-\epsilon_1)\epsilon_2 + \epsilon_1(1-\epsilon_2)\big),$$

where $\epsilon_1 := h^{-1}(1-I_1)$ and $\epsilon_2 := h^{-1}(1-I_2)$, and for $n > 2$ as

$$f_n^{\mathsf{ser}}(I_1, I_2, \ldots, I_n) := f_2^{\mathsf{ser}}\big(I_1, f_{n-1}^{\mathsf{ser}}(I_2, I_3, \ldots, I_n)\big).$$

(Including the case $n = 1$ allows to write the following formulas in a more compact form.)

The function $f_n^{\mathsf{ser}}(.)$ describes the mutual information of the channel formed by a serial concatenation of n independent BSCs, where the inputs of the first channel are assumed to be independent and uniformly distributed. With $I_1, I_2, \ldots, I_n$ denoting the mutual information values of the BSCs, the mutual information between the input of the first BSC and the output of the last BSC is given by $f_n^{\mathsf{ser}}(I_1, I_2, \ldots, I_n)$. Further details are provided in Appendix D. The function defined above is now used to express the extrinsic information.

Using the chain rule of mutual information [CT91], the extrinsic information on code symbol X_0 can be written as

$$I(X_0; \boldsymbol{Y}_{[1,N-1]}) = I(X_0; \boldsymbol{Y}_{[1,N-2]}) + I(X_0; Y_{N-1} | \boldsymbol{Y}_{[1,N-2]}). \tag{6.8}$$

The first term is equal to zero, i.e., $I(X_0; \boldsymbol{Y}_{[1,N-2]}) = 0$, as X_0 and $\boldsymbol{Y}_{[1,N-2]}$ are independent if neither X_{N-1} nor Y_{N-1} are known.

To determine the second term, we use the representation of the code symbols and the channel outputs over $\mathbb{F}_2$, written as $\breve{X}_i$ and $\breve{Y}_i$. Changing the symbol alphabet does not change the mutual information, i.e.,

$$I(X_0; Y_{N-1}|\boldsymbol{Y}_{[1,N-2]}) = I(\breve{X}_0; \breve{Y}_{N-1}|\breve{\boldsymbol{Y}}_{[1,N-2]}). \tag{6.9}$$

Let binary random variables $Z_i \in \mathbb{F}_2$, $i = 0, 1, \ldots, N-1$, be defined as

$$\begin{aligned} Z_0 &:= \breve{X}_0, \\ Z_1 &:= Z_0 \oplus \breve{X}_1, \\ Z_2 &:= Z_1 \oplus \breve{X}_2, \\ &\ldots \\ Z_{N-2} &:= Z_{N-3} \oplus \breve{X}_{N-2}, \\ Z_{N-2} &= \breve{X}_{N-1}, \\ Z_{N-1} &:= \breve{Y}_{N-1}. \end{aligned}$$

The penultimate line is not a definition, but an equality which results from the previous definitions and the parity check equation (6.6); it is only included for the sake of completeness. Due to their definitions, all Z_i are uniformly distributed and

$$I(\breve{X}_0; \breve{Y}_{N-1}|\breve{\boldsymbol{Y}}_{[1,N-2]}) = I(Z_0; Z_{N-1}|\breve{\boldsymbol{Y}}_{[1,N-2]}). \tag{6.10}$$

For the time being, assume $\breve{\boldsymbol{Y}}_{[1,N-2]} = \breve{\boldsymbol{y}}_{[1,N-2]}$, where $\breve{\boldsymbol{y}}_{[1,N-2]} \in \mathbb{F}_2^{N-2}$ denotes an arbitrary but fixed realization of $\breve{\boldsymbol{Y}}_{[1,N-2]}$. Then, the random variables Z_i form a Markov chain,

$$Z_0 \to Z_1 \to Z_2 \to \cdots \to Z_{N-3} \to Z_{N-2} \to Z_{N-1},$$

where each pair $Z_i \to Z_{i+1}$, $i = 0, 1, \ldots, N-2$, can be interpreted as a BSC. The mutual information of each BSC is as follows:

- $Z_0 \to Z_1$: The code symbol $\breve{X}_1$ represents the error symbol of this BSC. The crossover probability of the channel $\breve{X}_1 \to \breve{Y}_1$ and that of the channel $\breve{Y}_1 \to \breve{X}_1$ are equal due to the uniform distribution of $\breve{X}_1$. Thus, we have for the crossover probability ϵ_1 of the channel $Z_0 \to Z_1$:

  $$\epsilon_1 \in \left\{ p_{\breve{X}_1|\breve{Y}_1}(1|\breve{y}') : \breve{y}' \in \mathbb{F}_2 \right\} = \left\{ h^{-1}(1 - I_{\mathsf{int},1}), 1 - h^{-1}(1 - I_{\mathsf{int},1}) \right\}.$$

  The mutual information of this channel is given by $I(Z_0; Z_1) = 1 - h(\epsilon_1) = I_{\mathsf{int},1}$, which is independent of $\breve{y}_1$.

- $Z_i \to Z_{i+1}$, $i = 1, 2, \ldots, N-3$: Similar to $Z_0 \to Z_1$, the mutual information is given by $I(Z_i; Z_{i+1}) = I_{\mathsf{int},i+1}$.

- $Z_{N-2} \to Z_{N-1}$: This channel is identical to the channel $\breve{X}_{N-1} \to \breve{Y}_{N-1}$, and thus its mutual information is given by $I(Z_{N-2}; Z_{N-1}) = I_{\mathsf{int},N-1}$.

Notice that the mutual information of each BSC is independent of $\breve{\boldsymbol{y}}_{[1,N-2]}$.

As we have a serial concatenation of BSCs with known values of mutual information, we can apply the binary information function for serial concatenation according to Definition 6.1:

$$I(Z_0; Z_{N-1}|\breve{\boldsymbol{Y}}_{[1,N-2]} = \breve{\boldsymbol{y}}_{[1,N-2]}) = f_{N-1}^{\mathsf{ser}}(I_{\mathsf{int},1}, I_{\mathsf{int},2}, \ldots, I_{\mathsf{int},N-1}).$$

This mutual information is independent of $\breve{\boldsymbol{y}}_{[1,N-2]}$, and so the expected value with respect to $\breve{\boldsymbol{y}}_{[1,N-2]}$ is simply given as

$$\begin{aligned} I(Z_0; Z_{N-1}|\breve{\boldsymbol{Y}}_{[1,N-2]}) &= \mathrm{E}\Big\{ f_{N-1}^{\mathsf{ser}}(I_{\mathsf{int},1}, I_{\mathsf{int},2}, \ldots, I_{\mathsf{int},N-1}) \Big\} \\ &= f_{N-1}^{\mathsf{ser}}(I_{\mathsf{int},1}, I_{\mathsf{int},2}, \ldots, I_{\mathsf{int},N-1}). \end{aligned} \tag{6.11}$$

Using (6.11) in (6.10), we obtain

$$I(\breve{X}_0; \breve{Y}_{N-1}|\breve{\boldsymbol{Y}}_{[1,N-2]}) = f_{N-1}^{\mathsf{ser}}(I_{\mathsf{int},1}, I_{\mathsf{int},2}, \ldots, I_{\mathsf{int},N-1}).$$

After substituting this result into (6.8), we obtain the extrinsic information on code symbol X_0 for the case where all channels are BSCs:

$$I_{\mathsf{ext},0}^{\mathsf{BSC}} = f_{N-1}^{\mathsf{ser}}(I_{\mathsf{int},1}, I_{\mathsf{int},2}, \ldots, I_{\mathsf{int},N-1}). \tag{6.12}$$

General Binary-Input Symmetric Memoryless Channels

The two cases where either the channels are all BECs or all BSCs represent bounds on the extrinsic information for a code symbol, when only the values of intrinsic information for code symbols are known, and not the underlying channel models. This is proved by the following theorem.

Theorem 6.1 (Extrinsic Information for Single Parity Check Codes)
Let $X_0, X_1, \ldots, X_{N-1} \in \mathbb{B}$ denote the code symbols of a single parity check code of length N. Let $X_i \to Y_i$, $i = 1, 2, \ldots, N-1$, denote $N-1$ independent BISMCs having mutual information $I(X_i; Y_i)$. Let the intrinsic information on code symbol X_i be defined by $I_{\mathsf{int},i} := I(X_i; Y_i)$, $i = 1, 2, \ldots, N-1$, and let the extrinsic information on code symbol X_0 by defined by $I_{\mathsf{ext},0} := I(X_0; \boldsymbol{Y}_{\setminus 0})$. Then, the following tight bounds hold:

$$\begin{aligned} I_{\mathsf{ext},0} &\geq I_{\mathsf{int},1} I_{\mathsf{int},2} \cdots I_{\mathsf{int},N-1}, \\ I_{\mathsf{ext},0} &\leq f_{N-1}^{\mathsf{ser}}(I_{\mathsf{int},1}, I_{\mathsf{int},2}, \ldots, I_{\mathsf{int},N-1}). \end{aligned}$$

The lower bound is achieved if the channels are all BECs, and the upper bound is achieved if the channels are all BSCs.

To prove this theorem, we use the following lemma.

Lemma 6.1
The binary information function for serial concatenation (Definition 6.1) has the following two properties:

(a) $f_n^{\mathsf{ser}}(I_1, I_2, \ldots, I_n)$ *is convex-*$\cap$ *in each* I_i, $i = 1, 2, \ldots, n$;

(b) $f_n^{\mathsf{ser}}(I_1, I_2, \ldots, I_n)$ *is lower-bounded by the product of its arguments:*

$$f_n^{\mathsf{ser}}(I_1, I_2, \ldots, I_n) \geq I_1 I_2 \cdots I_n.$$

Proof. The two properties are proved for $n = 2$, i.e., for $f_2^{\mathsf{ser}}(I_1, I_2)$. The general case follows by induction.

First, we define the function

$$\begin{aligned} g(\iota) &:= 1 - f_2^{\mathsf{ser}}\big(1 - \iota, 1 - h(\rho)\big) \\ &= h\big([1 - 2\rho]h^{-1}(\iota) + \rho\big) \end{aligned}$$

for $\iota \in [0, 1]$ with parameter $\rho \in [0, \frac{1}{2}]$. (The range of ρ is chosen such that $h^{-1}(h(\rho)) = \rho$.) As $\iota = 1 - I_1$ and $h(\rho) = 1 - I_2$, ι and $h(\rho)$ correspond to entropies, and ρ corresponds to a probability.

(a): The function $f_2^{\mathsf{ser}}(I_1, I_2)$ is symmetric in I_1 and I_2, and so it is sufficient to consider it as a function of I_1 with constant parameter I_2. Then, $f_2^{\mathsf{ser}}(I_1, I_2)$ is convex-$\cap$ in I_1 for constant I_2 if and only if $g(\iota)$ is convex-$\cup$ in ι for constant ρ. For illustration, the function $g(\iota)$ is plotted versus ι for several values of ρ in Fig. 6.3. The plot indicates that $g(\iota)$ is convex-$\cup$ for all ρ. A formal proof is provided in Appendix E.

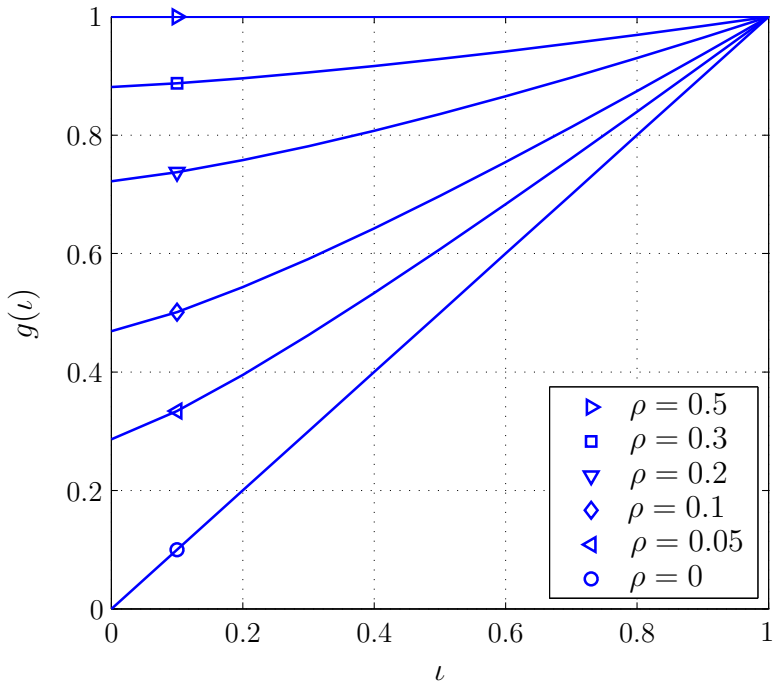


Figure 6.3: Function $g(\iota)$ for several values of parameter ρ (cf. Lemma 6.1).

(b): For the time being, let I_2 be constant. Then, the (one-dimensional) bound on f_2^{ser} holds if and only if

$$h\big([1 - 2\rho]h^{-1}(\iota) + \rho\big) \leq \big(1 - h(\rho)\big)\iota + h(\rho)$$

for $\iota \in [0,1]$ and $\rho \in [0, \frac{1}{2}]$. For $\iota = 0$ and $\iota = 1$, the left hand side is equal to the right hand side. Regarding this and the fact that $g(\iota)$ is convex-$\cup$, the right hand side represents the secant of $g(\iota)$ for $\iota \in [0,1]$, and thus the inequality holds. These considerations are independent of ρ, and thus statement (b) holds for all I_2. QED

Lemma 6.1 is now used to prove Theorem 6.1:

Proof. All channels are assumed to be BISMCs. Therefore we may define a subchannel indicator A_i and a (mutual) information indicator J_i for each channel $X_i \to Y_i$ (cf. Chapter 2). Notice that the expectation of the mutual information indicator is equal to the intrinsic information, $\mathrm{E}\{J_i\} = I(X_i;Y_i) = I_{\mathsf{int},i}$.

The extrinsic information does not change if it is written conditioned on the subchannel indicators $\boldsymbol{A}_{\setminus 0}$,

$$\begin{aligned} I_{\mathsf{ext},0} &= I(X_0;\boldsymbol{Y}_{\setminus 0}) = I(X_0;\boldsymbol{Y}_{\setminus 0},\boldsymbol{A}_{\setminus 0}) \\ &= I(X_0;\boldsymbol{A}_{\setminus 0}) + I(X_0;\boldsymbol{Y}_{\setminus 0}|\boldsymbol{A}_{\setminus 0}) \\ &= I(X_0;\boldsymbol{Y}_{\setminus 0}|\boldsymbol{A}_{\setminus 0}), \end{aligned}$$

since $\boldsymbol{A}_{\setminus 0}$ is independent of X_0 by definition. (Notice that $\boldsymbol{A}_{\setminus 0} = \boldsymbol{A}_{[1,N-1]}$ and $\boldsymbol{Y}_{\setminus 0} = \boldsymbol{Y}_{[1,N-1]}$.) For a given realization $\boldsymbol{a}_{[1,N-1]}$ of subchannel indicators, we have the case where the channels are all BSCs, and we can apply the function $f^{\mathsf{ser}}_{N-1}(\ldots)$ according to Definition 6.1:

$$I(X_0;\boldsymbol{Y}_{[1,N-1]}|\boldsymbol{A}_{[1,N-1]} = \boldsymbol{a}_{[1,N-1]}) = f^{\mathsf{ser}}_{N-1}(j_1, j_2, \ldots, j_{N-1}).$$

Notice that $j_i = I(X_i;Y_i|A_i = a_i)$ is the mutual information corresponding to the subchannel $A_i = a_i$ of channel $X_i \to Y_i$. Taking the expectation, we can write the extrinsic information as

$$\begin{aligned} I_{\mathsf{ext},0} &= I(X_0;\boldsymbol{Y}_{[1,N-1]}|\boldsymbol{A}_{[1,N-1]}) \\ &= \mathrm{E}\Big\{ f^{\mathsf{ser}}_{N-1}(J_1, J_2, \ldots, J_{N-1}) \Big\}. \end{aligned} \tag{6.13}$$

Now, the two properties of the function $f^{\mathsf{ser}}_{N-1}(\ldots)$ given in Lemma 6.1 are exploited. Using Lemma 6.1(b) in (6.13), we obtain

$$\mathrm{E}\Big\{ f^{\mathsf{ser}}_{N-1}(J_1, J_2, \ldots, J_{N-1}) \Big\} \geq \mathrm{E}\Big\{ J_1 J_2 \cdots J_{N-1} \Big\} = I_{\mathsf{int},1} I_{\mathsf{int},2} \cdots I_{\mathsf{int},N-1},$$

where $\mathrm{E}\{J_i\} = I_{\mathsf{int},i}$ is used. On the other hand, due to Lemma 6.1(a), Jensen's inequality [CT91] can be applied in (6.13), and we obtain

$$\begin{aligned} &\mathrm{E}\Big\{ f^{\mathsf{ser}}_{N-1}(J_1, J_2, \ldots, J_{N-1}) \Big\} \leq \\ &\qquad \leq f^{\mathsf{ser}}_{N-1}\Big(\mathrm{E}\{J_1\}, \mathrm{E}\{J_2\}, \ldots, \mathrm{E}\{J_{N-1}\}) \Big) = f^{\mathsf{ser}}_{N-1}(I_{\mathsf{int},1}, I_{\mathsf{int},2}, \ldots, I_{\mathsf{int},N-1}). \end{aligned}$$

Thus, we have proved the two bounds.

According to (6.7) and (6.12), the lower bound and the upper bound are actually achieved when the channels are all BECs or all BSCs, respectively. QED

Remark 6.1
When not only the mutual information but also the information profiles of the individual channels are given, the *precise* extrinsic information can be computed by simply evaluating (6.13).

6.2.2 Repetition Codes

Consider a repetition code of length N, which is defined by the constraint

$$X_0 = X_1 = \cdots = X_{N-1} \tag{6.14}$$

on the code symbols X_i. The code constraints and the transmission channels $X_i \to Y_i$ are shown in Fig. 6.4 for $N = 4$. In the following derivation, we consider only the complete information on code symbol X_0. Due to the symmetric structure of the code, the expressions for the other code symbols are similar.

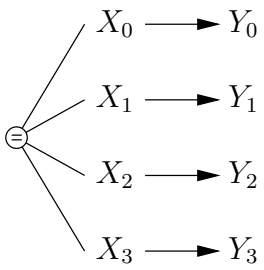


Figure 6.4: Repetition code of length $N = 4$.

Similarly to the case of single parity check codes, we discuss first the two cases where all channels are BECs and where all channels are BSCs. Both the BEC case and the BSC case turn out to be a simple combinatorial problems. These cases are then shown to lead to the maximal and to the minimal complete information.

Binary Erasure Channels

If the channels are all BECs, the value of code symbol X_0 can be recovered with certainty provided that not all channel outputs are erasures. (An erasure corresponds to $Y_i = 0$.) Otherwise, no extrinsic information on code symbol X_0 is available. Using (2.7) and the probabilities of these events, it can easily be seen that

$$I_{\mathsf{cmp},0}^{\mathsf{BEC}} = 1 - (1 - I_{\mathsf{int},0})(1 - I_{\mathsf{int},1}) \cdots (1 - I_{\mathsf{int},N-1}). \tag{6.15}$$

Binary Symmetric Channels

For the case where the channels are all BSCs, we introduce the following function.

Definition 6.2 (Binary Information Function for Parallel Concatenation)
Let $I_1, I_2, \ldots, I_n \in [0,1]$, $n \geq 1$. Let further $\boldsymbol{r} = [r_1, r_2, \ldots, r_n] \in \mathbb{B}^n$. We define the binary information function for parallel concatenation as

$$f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n) := -\sum_{\boldsymbol{r}\in\mathbb{B}^n} \psi(\boldsymbol{r}) \operatorname{ld} \psi(\boldsymbol{r}) - \sum_{i=1}^{n}(1 - I_i)$$

with

$$\psi(\boldsymbol{r}) := \frac{1}{2}\Big(\prod_{i=1}^{n} \varphi_i(r_i) + \prod_{i=1}^{n}\big(1 - \varphi_i(r_i)\big)\Big)$$

and

$$\varphi_i(r_i) := \begin{cases} \epsilon_i & \text{for} \quad r_i = +1, \\ 1 - \epsilon_i & \text{for} \quad r_i = -1, \end{cases}$$

where $\epsilon_i := h^{-1}(1 - I_i)$ for $i = 1, 2, \ldots, n$. —

(Similar to Definition 6.1, we have included the case $n = 1$, so that the following formulas can be written in a more compact form.)

This function describes the mutual information of a channel formed by a parallel concatenation of n independent BSCs, i.e., BSCs having the same inputs. These inputs are assumed to be independent and uniformly distributed (i.u.d.). With $I_1, I_2, \ldots, I_n$ denoting the mutual information values of the BSCs, the mutual information between the (common) input and the vector of all channel outputs is given by $f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n)$. Appendix D provides further details.

The complete information on code symbol X_0 corresponds exactly to the interpretation of function f_N^{par}. Thus, the complete information on code symbol X_0 can be written as

$$I_{\mathsf{cmp},0}^{\mathsf{BSC}} = f_N^{\mathsf{par}}(I_{\mathsf{int},0}, I_{\mathsf{int},1}, \ldots, I_{\mathsf{int},N-1}). \tag{6.16}$$

General Binary-Input Symmetric Memoryless Channels

The two cases considered above represent bounds on the complete information on a code symbol when only the intrinsic information on code symbols is known, and not the underlying channel models. This is stated in the following theorem.

Theorem 6.2 (Complete Information for Repetition Codes)
Let $X_0, X_1, \ldots, X_{N-1} \in \mathbb{B}$ denote the code symbols of a repetition code of length N. Let $X_i \to Y_i$, $i = 1, 2, \ldots, N-1$, denote $N-1$ independent BISMCs having mutual information $I(X_i; Y_i)$. Let the intrinsic information on code symbol X_i be defined by $I_{\mathsf{int},i} := I(X_i; Y_i)$, $i = 1, 2, \ldots, N-1$, and let the complete information on code symbol X_0 by defined by $I_{\mathsf{cmp},0} := I(X_0; \boldsymbol{Y}_{[0,N-1]})$. Then, the following tight bounds hold:

$$\begin{aligned} I_{\mathsf{cmp},0} &\geq f_N^{\mathsf{par}}(I_{\mathsf{int},0}, I_{\mathsf{int},1}, \ldots, I_{\mathsf{int},N-1}), \\ I_{\mathsf{cmp},0} &\leq 1 - (1 - I_{\mathsf{int},0})(1 - I_{\mathsf{int},1}) \cdots (1 - I_{\mathsf{int},N-1}). \end{aligned}$$

The lower bound is achieved if the channels are all BSCs, and the upper bound is achieved if the channels are all BECs.

Note that BSCs achieve the lower bound for repetition codes, but the upper bound for single parity check codes; the reverse holds for BECs.

To prove this theorem, we use the following lemma.

Lemma 6.2
The binary information function for parallel concatenation (Definition 6.2) has the following two properties:

(a) $f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n)$ *is convex-∪ in each* I_i, $i = 1, 2, \ldots, n$;

(b) $f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n)$ *is upper-bounded as*

$$f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n) \leq 1 - (1 - I_1)(1 - I_2) \cdots (1 - I_n).$$

Proof. Consider first the case $n = 2$. The binary information function for parallel concatenation and that for serial concatenation are related as

$$f_2^{\mathsf{par}}(I_1, I_2) = I_1 + I_2 - f_2^{\mathsf{ser}}(I_1, I_2), \tag{6.17}$$

as can easily be shown[4]. Then, Lemma 6.1 immediately gives the proof.

Consider now the case $n > 2$. In the sequel, we exploit the meaning of function f_n^{par} (see above and Appendix D): for n BSCs $X \to Y_i$ with $I_i := I(X; Y_i)$, $i = 1, 2, \ldots, n$, and the same i.u.d. inputs X, the complete information of the parallel concatenated channel $X \to \boldsymbol{Y}_{[1,n]}$ is given by

$$I(X; \boldsymbol{Y}_{[1,n]}) = f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n).$$

Furthermore, we make use of the superchannel $X \to \boldsymbol{Y}_{[2,n]}$. This chanel is a BISMC, because all individual channels are BSCs, and so we may define a mutual information indicator J' for this channel. Notice that $\mathrm{E}\{J'\} = I(X; \boldsymbol{Y}_{[2,n]})$.

(a): Using this superchannel, the mutual information of the parallel concatenated channel and thus function f_n^{par} can be written as

$$I(X; \boldsymbol{Y}_{[1,n]}) = f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n) = \mathrm{E}\Big\{ f_2^{\mathsf{par}}(I_1, J') \Big\}. \tag{6.18}$$

As the functions $f_2^{\mathsf{par}}(I_1, J')$ are convex-∪ in I_1 for any J' (see above) and $f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n)$ is a weighted sum of such functions, $f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n)$ must be convex-∪ in I_1. This holds for $I_2, I_3, \ldots, I_n$ in an analogous way, and we have (a).

(b): We start with (6.18) and use the upper bound of f_2^{par}:

$$I(X; \boldsymbol{Y}_{[1,n]}) = \mathrm{E}\Big\{ f_2^{\mathsf{par}}(I_1, J') \Big\} \leq$$
$$\leq \mathrm{E}\Big\{ 1 - (1 - I_1)(1 - J') \Big\} = 1 - (1 - I_1)(1 - \mathrm{E}\{J'\}).$$

[4]This relations holds only for $n = 2$.

The mutual information of the superchannel, $\mathrm{E}\{J'\} = I(X; \boldsymbol{Y}_{[2,n]})$ may now be upper bounded in the same way, using the new superchannel $X \rightarrow \boldsymbol{Y}_{[3,n]}$ with mutual information indicator J'':

$$I(X; \boldsymbol{Y}_{[2,n]}) = \mathrm{E}\Big\{ f_2^{\mathsf{par}}(I_2, J'') \Big\} \leq$$
$$\leq \mathrm{E}\Big\{ 1 - (1 - I_2)(1 - J'') \Big\} = 1 - (1 - I_2)(1 - \mathrm{E}\{J''\}).$$

Proceeding recursively, we obtain

$$\begin{aligned} f_n^{\mathsf{par}}(I_1, I_2, \ldots, I_n) &\leq 1 - (1 - I_1)(1 - \mathrm{E}\{J'\}) \quad \leq \\ &\leq 1 - (1 - I_1)(1 - I_2)(1 - \mathrm{E}\{J''\}) \quad \leq \cdots \leq \\ &\leq 1 - (1 - I_1)(1 - I_2) \cdots (1 - I_n), \end{aligned}$$

and we have (b). QED

These two properties are now exploited to prove Theorem 6.2. Notice that the same methods are applied as in the proof of Theorem 6.1.

Proof. All channels are assumed to be BISMCs. Therefore we may define a subchannel indicator A_i and a (mutual) information indicator J_i for each channel $X_i \rightarrow Y_i$ (cf. Chapter 2). Notice that the expectation of the mutual information indicator is equal to the intrinsic information, $\mathrm{E}\{J_i\} = I(X_i; Y_i) = I_{\mathsf{int},i}$. We use the abbreviations $\boldsymbol{Y} = \boldsymbol{Y}_{[0,N-1]}$ and $\boldsymbol{A} = \boldsymbol{A}_{[0,N-1]}$.

The complete information does not change if it is written conditioned on the subchannel indicators $\boldsymbol{A}$,

$$\begin{aligned} I_{\mathsf{cmp},0} &= I(X_0; \boldsymbol{Y}) = I(X_0; \boldsymbol{Y}, \boldsymbol{A}) \\ &= I(X_0; \boldsymbol{A}) + I(X_0; \boldsymbol{Y}|\boldsymbol{A}) \\ &= I(X_0; \boldsymbol{Y}|\boldsymbol{A}), \end{aligned}$$

since $\boldsymbol{A}$ is independent of X_0 by definition. For a given realization $\boldsymbol{a}$ of subchannel indicators, we have the case where the channels are all BSCs, and we can apply the function $f_N^{\mathsf{par}}(\ldots)$ according to Definition 6.2:

$$I(X_0; \boldsymbol{Y}_{[0,N-1]}|\boldsymbol{A}_{[0,N-1]} = \boldsymbol{a}_{[0,N-1]}) = f_N^{\mathsf{par}}(j_0, j_1, \ldots, j_{N-1}).$$

Notice that $j_i = I(X_i; Y_i|A_i = a_i)$ is the mutual information corresponding to the subchannel $A_i = a_i$ of channel $X_i \rightarrow Y_i$. Taking the expectation, we can write the complete information as

$$\begin{aligned} I_{\mathsf{cmp},0} &= I(X_0; \boldsymbol{Y}_{[0,N-1]}|\boldsymbol{A}_{[0,N-1]}) \\ &= \mathrm{E}\Big\{ f_N^{\mathsf{par}}(J_0, J_1, \ldots, J_{N-1}) \Big\}. \end{aligned} \tag{6.19}$$

Now, the two properties of the function $f_N^{\mathsf{par}}(\ldots)$ given in Lemma 6.2 are exploited. Using Lemma 6.2(b) in (6.19), we obtain

$$\mathrm{E}\Big\{ f_N^{\mathsf{par}}(J_0, J_1, \ldots, J_{N-1}) \Big\} \leq \mathrm{E}\Big\{ 1 - (1 - J_0)(1 - J_1) \cdots (1 - J_{N-1}) \Big\} =$$
$$= 1 - (1 - I_{\mathsf{int},0})(1 - I_{\mathsf{int},1}) \cdots (1 - I_{\mathsf{int},N-1}),$$

where $\mathrm{E}\{J_i\} = I_{\mathsf{int},i}$ has been used. On the other hand, Lemma 6.2(a) and Jensen's inequality [CT91] can be applied in (6.19), and we obtain

$$\mathrm{E}\Big\{ f_N^{\mathsf{par}}(J_0, J_1, \ldots, J_{N-1}) \Big\} \geq$$
$$\geq f_N^{\mathsf{par}}\Big(\mathrm{E}\{J_0\}, \mathrm{E}\{J_1\}, \ldots, \mathrm{E}\{J_{N-1}\}) \Big) = f_N^{\mathsf{par}}(I_{\mathsf{int},0}, I_{\mathsf{int},1}, \ldots, I_{\mathsf{int},N-1}).$$

Thus, we have the two bounds.

According to (6.15) and (6.16), the upper bound and the lower bound are actually achieved when the channels are all BECs or all BSCs, respectively. QED

In the case of a repetition code, the extrinsic information on a code symbol can easily be determined via the complete information. Assume a repetition code of length N, where the code symbols are transmitted over channels $X_i \to Y_i$, $i = 0, 2, \ldots, N-1$. The extrinsic information on code symbol X_0, $I_{\mathsf{ext},0} = I(X_0, \boldsymbol{Y}_{[1,N-1]})$, is identical to the complete information on code symbol X_1, $I_{\mathsf{cmp},1} = I(X_1, \boldsymbol{Y}_{[1,N-1]})$ for the case where the channel $X_0 \to Y_0$ is discarded. Therefore, Theorem 6.2 can directly be applied to bound the extrinsic information.

Theorem 6.3 (Extrinsic Information for Repetition Codes)
Let $X_0, X_1, \ldots, X_{N-1} \in \mathbb{B}$ denote the code symbols of a repetition code of length N. Let $X_i \to Y_i$, $i = 1, 2, \ldots, N-1$, denote $N-1$ independent BISMCs having mutual information $I(X_i; Y_i)$. Let the intrinsic information on code symbol X_i be defined by $I_{\mathsf{int},i} := I(X_i; Y_i)$, $i = 1, 2, \ldots, N-1$, and let the extrinsic information on code symbol X_0 by defined by $I_{\mathsf{ext},0} := I(X_0; \boldsymbol{Y}_{\setminus 0})$. Then, the following tight bounds hold:

$$\begin{aligned} I_{\mathsf{ext},0} &\geq f_{N-1}^{\mathsf{par}}(I_{\mathsf{int},1}, I_{\mathsf{int},2}, \ldots, I_{\mathsf{int},N-1}), \\ I_{\mathsf{ext},0} &\leq 1 - (1 - I_{\mathsf{int},1})(1 - I_{\mathsf{int},2}) \cdots (1 - I_{\mathsf{int},N-1}). \end{aligned}$$

The lower bound is achieved if the channels are all BSCs, and the upper bound is achieved if the channels are all BECs.

Remark 6.2
When not only the mutual information but also the information profiles of the individual channels are given, the *precise* complete information can be computed by simply evaluating (6.19). Similarly, the *precise* extrinsic information can be computed.

6.2.3 Complete Information

The complete information on a code symbol comprises the intrinsic and the extrinsic information [HH02]. Bounds on combining these two values of mutual information are addressed in this section.

The formation of complete information on a code symbol based on the intrinsic and the extrinsic information corresponds to the case where this code symbol is transmitted over two parallel independent channels. This is the *most basic scenario for information combining*. In [HH02,Hub02,HH03], it was considered for the first time, and the complete

information was computed for specific channel models. The bounds on information combining were presented in [LHHH03]. Since this scenario corresponds to a repetition code of length 2, the bounds on the complete information follow immediately from Theorem 6.2.

Theorem 6.4 (Complete Information)
Let $X_0, X_1, \ldots, X_{N-1} \in \mathbb{B}$ denote the code symbols of a linear code of length N. Let $X_i \to Y_i$, $i = 1, 2, \ldots, N-1$, denote $N-1$ independent BISMCs. Let the intrinsic information on code symbol X_0 be defined by $I_{\mathsf{int},0} := I(X_0; Y_0)$, let the extrinsic information on code symbol X_0 be defined by $I_{\mathsf{ext},0} := I(X_0; \boldsymbol{Y}_{\setminus 0})$, and let the complete information on code symbol X_0 be defined by $I_{\mathsf{cmp},0} := I(X_0; \boldsymbol{Y})$. Then, the following bounds hold:

$$\begin{aligned} I_{\mathsf{cmp},0} &\geq f_2^{\mathsf{par}}(I_{\mathsf{int},0}, I_{\mathsf{ext},0}), \\ I_{\mathsf{cmp},0} &\leq 1 - (1 - I_{\mathsf{int},0})(1 - I_{\mathsf{ext},0}). \end{aligned}$$

The lower bound is achieved if the intrinsic channel $X_0 \to Y_0$ and the extrinsic channel $X_0 \to \boldsymbol{Y}_{\setminus 0}$ are BSCs. The upper bound is achieved if the intrinsic and the extrinsic channel are BECs.

Remark 6.3
When not only the mutual information but also the information profiles of the intrinsic and the extrinsic channel are given, the *precise* complete information can be computed by simply evaluating (6.19) for the case $N = 2$, where the two channels correspond to the intrinsic and the extrinsic channel.

6.2.4 Impact of Information Profiles

In the previous sections, the BSC and the BEC were shown to lead to the extremes of the combined information. This behavior can be explained using the concept of information profiles, introduced in Chapter 2.

The information profile of a BISMC $X \to Y$ is the distribution of the mutual information of its binary symmetric subchannels. Let A denote the subchannel indicator of this channel, and let J denote the mutual information indicator of this channel. Then the information profile is the probability density function $p_J(j)$.

For a given mutual information of the channel,

$$I := \mathrm{E}\{J\} = I(X; Y),$$

the variance of J,

$$\sigma_J^2 := \mathrm{E}\{(J - I)^2\},$$

is minimal if the channel is a BSC, and it is maximal if the channel is a BEC. To be precise, we have

$$\sigma_{J,\mathsf{BSC}}^2 = 0$$

in the case of the BSC, and we have

$$\begin{aligned} \sigma_{J,\mathsf{BEC}}^2 &= \delta \cdot (0 - I)^2 + (1 - \delta) \cdot (1 - I)^2 \\ &= (1 - I)I^2 + I(1 - I)^2 \\ &= I(1 - I) \end{aligned}$$

in the case of the BEC, where $\delta = 1 - I$ denotes the erasure probability.

The combined information values for single parity check codes and for repetition codes can be written as the expectations of functions f_N^{ser} and f_N^{par}, according to (6.13) and (6.19). These expectations are evaluated with respect to the information profiles of the involved channels. The two functions are convex and monotonically increasing in each argument; the first property is proved in Lemma 6.1 and Lemma 6.2, and the second property can easily be shown. Determining the information profiles that lead to the maximum and the minimum of the expected values, as done in Theorem 6.1, Theorem 6.2, Theorem 6.3, and Theorem 6.4, corresponds to the following problem:

Given a random variable J having mean value I and a convex and monotonically increasing function $f(j)$, determine the distributions (corresponding to the information profiles) that lead to the minimum and to the maximum of $\mathrm{E}\{f(J)\}$, respectively. These optima are obviously attained by the distributions having the minimum and the maximum variance, as the function f is convex and monotonically increasing.

Since the BSC and the BEC correspond to the information profiles with the minimal and the maximal variance, they result in the extremes of the combined information for repetition codes and for single parity check codes [Hub04]. (Whether the BEC or the BSC leads to the minimum or maximum depends on whether the function is convex-$\cup$ or convex-$\cap$.) Thus, the concept of information profiles provides an elegant method to explain the bounds on information combining.

6.3 Bounds on Information Transfer Functions

Information transfer functions are defined in Section 3.3; they include extrinsic information transfer (EXIT) functions and complete information transfer (CIT) functions. Information transfer functions depend on the models applied for the info-symbol channel and the code-symbol channel. Using the bounds on information combining from the previous section, we derive now bounds on information transfer functions that are valid for all BISMCs. We discuss single parity check codes, repetition codes, and the accumulator, which is the recursive rate-1 memory-1 convolutional encoder.

We first specialize the decoding model from Section 6.1, used for deriving the bounds on information combining, such that it corresponds to the decoding model from Section 3.2, on which the definitions of the information transfer functions are based:

The channels over which the code symbols are transmitted are assumed to be either the communication channel or the a-priori channel. The communication channel models the physical transmission channel; its mutual information is called the *channel information*, and it is denoted by I_{ch}. The a-priori channel is the virtual channel between a code symbol and a soft-value provided by another constituent decoder; its mutual information is called the *a-priori information*, and it is denoted by I_{apri}. The observation Y_i of a code symbol X_i is thus either a channel soft-value or an a-priori soft-value. Correspondingly, the intrinsic information on X_i is either the channel information I_{ch} or the a-priori information I_{apri}. Notice that both the channel information and the a-priori information are values of mutual information.

The following kinds of information transfer functions are considered in the sequel:

CIT functions (only channel information) All code symbols are transmitted over the communication channel. The corresponding CIT functions $\mathrm{itf}_{\mathsf{cmp}}$ and $\mathrm{itf}_{\mathsf{wcmp}}$ are given in Definition 3.8. (The code-symbol pre-decoding information is equal to I_{ch}; the info word length is equal to K.)

EXIT functions for code symbols (only a-priori information) All code symbols are transmitted over the a-priori channel. This corresponds to the outer decoder of a serially concatenated coding scheme and to the check-node decoder of an LDPC code. The code-symbol EXIT functions $\mathrm{itf}_{\mathsf{ext},X}$ are given in Definition 3.7. (The info-symbol pre-decoding information is equal to zero, and the code-symbol pre-decoding information is equal to I_{apri}; the code word length is equal to N.)

EXIT functions for code symbols (a-priori and channel information) The systematic code symbols are treated as info symbols, and they are transmitted over the communication channel; the nonsystematic code symbols are treated as actual code symbols, and they are transmitted over the a-priori channel. This corresponds to the variable-node decoder of an LDPC code. The code-symbol EXIT functions $\mathrm{itf}_{\mathsf{ext},X}$ are given in Definition 3.7. (The info-symbol pre-decoding information is equal to I_{ch}, the code-symbol pre-decoding information is equal to I_{apri}, and the code word length is equal to $N - K$.)

EXIT functions for info symbols (a-priori and channel information) The systematic code symbols are treated as info symbols, and they are transmitted over the a-priori channel; the nonsystematic code symbols are treated as actual code symbols, and they are transmitted over the communication channel. This corresponds to the inner decoder of a serially concatenated coding scheme. The info-symbol EXIT functions $\mathrm{itf}_{\mathsf{ext},U}$ are given in Definition 3.7. (The info-symbol pre-decoding information is equal to I_{apri}, the code-symbol pre-decoding information is equal to I_{ch}, and the info word length is equal to K.)

6.3.1 Single Parity Check Codes

Consider a single parity check code of length N as defined in Section 6.2.1. In the sequel, we determine bounds on the EXIT functions and bounds on the CIT functions.

Bounds on EXIT Functions

Assume first that a-priori information is available for all code symbols and there is no channel information for any code symbol, i.e., $I_{\mathsf{int},i} = I_{\mathsf{apri}}$ for $i = 0, 1, \ldots, N-1$. This corresponds to the decoding operation for a check node in the iterative decoder for a low-density parity-check code (cf. Section 3.6). It also applies when a single parity check code is used as an outer code in a serially concatenated coding scheme (cf. Section 3.5.2).

The extrinsic information on code symbol X_0 can be bounded according to Theorem 6.1. Since the average extrinsic information is equal to the extrinsic information on

code symbol X_0, we have the bounds

$$\begin{aligned} I_{\mathsf{ext}} &\geq (I_{\mathsf{apri}})^{N-1}, \\ I_{\mathsf{ext}} &\leq f_{N-1}^{\mathsf{ser}}(I_{\mathsf{apri}}, I_{\mathsf{apri}}, \ldots, I_{\mathsf{apri}}). \end{aligned} \tag{6.20}$$

These bounds are depicted in Fig. 6.5.

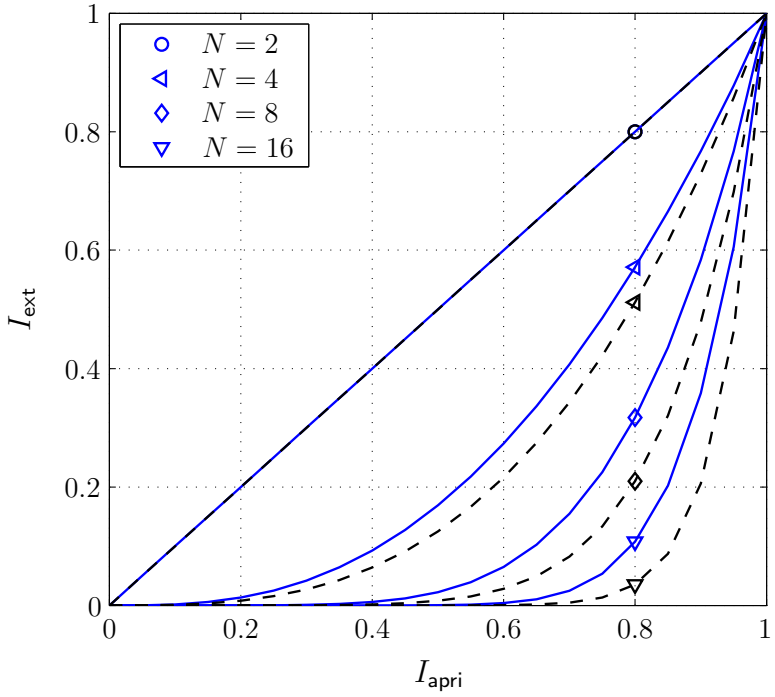


Figure 6.5: Bounds on EXIT functions for single parity check codes of several code lengths N. Upper bounds (solid lines) correspond to BSCs and lower bounds (dashed lines) correspond to BECs. (Mutual information is given in bit/use.)

Assume now that channel information on code symbol X_{N-1} and a-priori information on all other code symbols is available, i.e., $I_{\mathsf{int},i} = I_{\mathsf{apri}}$ for $i = 0, 1, \ldots, N-2$ and $I_{\mathsf{int},N-1} = I_{\mathsf{ch}}$. This is the case when single parity check codes are used as inner codes in serially concatenated coding schemes, and only the parity symbol X_{N-1} is transmitted over the communication channel; but also when single parity check codes are used in repeat-accumulate codes as proposed in [tBK03].

Using Theorem 6.1, we obtain the bounds

$$\begin{aligned} I_{\mathsf{ext}} &\geq I_{\mathsf{ch}} \cdot (I_{\mathsf{apri}})^{N-2}, \\ I_{\mathsf{ext}} &\leq f_{N-1}^{\mathsf{ser}}(I_{\mathsf{ch}}, I_{\mathsf{apri}}, \ldots, I_{\mathsf{apri}}). \end{aligned} \tag{6.21}$$

These bounds are depicted in Fig. 6.6. Obviously, the extrinsic information cannot become larger than the channel information, even if the a-priori information is equal to 1. Therefore, these codes are unattractive as inner codes of a serially concatenated coding scheme, because iterative decoding can never achieve mutual information of 1, and thus the symbol estimates can never become error-free.

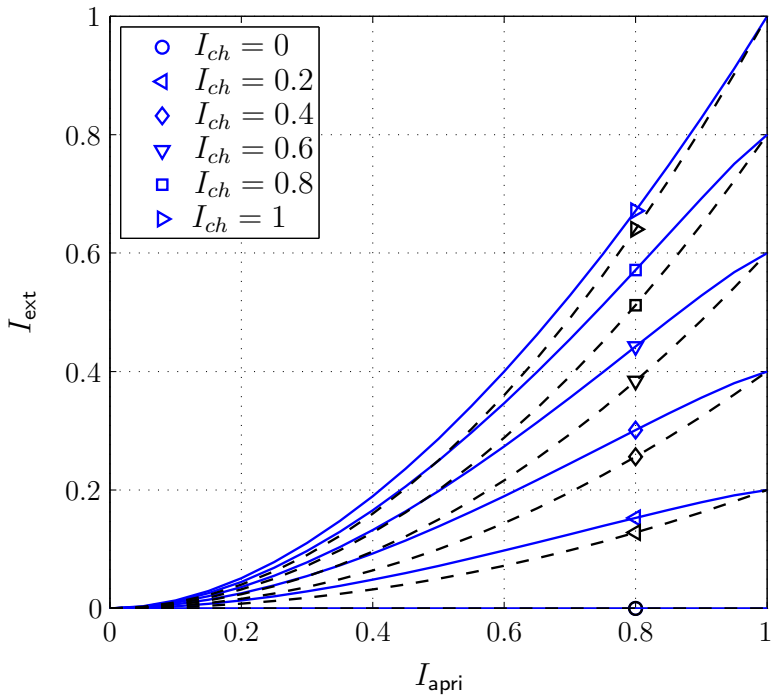


Figure 6.6: Bounds on EXIT functions for single parity check codes of length $N = 4$ for several values of channel information I_{ch}. Upper bounds (solid lines) correspond to BSCs and lower bounds (dashed lines) correspond to BECs. (Mutual information is given in bit/use.)

Bounds on CIT Functions

For the CIT function, we assume that channel information is available for all code symbols, i.e., $I_{\mathsf{int},i} = I_{\mathsf{ch}}$ for $i = 0, 1, \ldots, N-1$. The complete information on code symbol X_0 is a combination of the intrinsic information $I_{\mathsf{int},0} = I_{\mathsf{ch}}$ and the extrinsic information $I_{\mathsf{ext},0}$, and it can be bounded according to Theorem 6.4. The extrinsic information on code symbol X_0 is bounded as

$$I_{\mathsf{ch}}^{N-1} \leq I_{\mathsf{ext},0} \leq f_{N-1}^{\mathsf{ser}}(I_{\mathsf{ch}}, \ldots, I_{\mathsf{ch}}),$$

according to Theorem 6.1. Using the lower bound on the extrinsic information in the lower bound on the complete information, we obtain

$$I_{\mathsf{cmp},0} \geq f_2^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{ch}}^{N-1}). \tag{6.22}$$

Similarly, using the upper bound on the extrinsic information in the upper bound on the complete information, we obtain

$$I_{\mathsf{cmp},0} \leq 1 - (1 - I_{\mathsf{ch}})\big(1 - f_{N-1}^{\mathsf{ser}}(I_{\mathsf{ch}}, I_{\mathsf{ch}}, \ldots, I_{\mathsf{ch}})\big). \tag{6.23}$$

The complete information is the same for each systematic symbol. Therefore, bounds

on the (symbol-wise) complete information are given by

$$\begin{aligned} I_{\mathsf{cmp}} &\geq f_2^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{ch}}^{N-1}), \\ I_{\mathsf{cmp}} &\leq 1-(1-I_{\mathsf{ch}})(1-f_{N-1}^{\mathsf{ser}}(I_{\mathsf{ch}}, I_{\mathsf{ch}}, \ldots, I_{\mathsf{ch}})), \end{aligned} \tag{6.24}$$

and we have bounds on the CIT functions. These bounds are plotted in Fig. 6.7. For each code length N, a large gap between the upper bound and the bound given by the ideal coding scheme (cf. Section 3.3) can be observed.

In contrast to the bounds on the extrinsic information, these bounds on the complete information are not tight, because contradictory assumptions on the model for the communication channel are used in the derivation. Consider first the lower bound. The extrinsic information is minimal if the channels are BECs, i.e., if the communication channel is a BEC. On the other hand, the lower bound on the combination of the extrinsic and the intrinsic information holds with equality if both channels are BSCs, i.e., if the communication channel is a BSC. Due to this contradiction, the lower bound on the complete information cannot be tight.

Consider now the upper bound on the complete information. The extrinsic information is maximal if the channels are BSC, i.e., if the communication channel is a BSC. On the other hand, the upper bound on the combination of the extrinsic and the intrinsic information holds with equality if both channels are BECs, i.e., if the communication channel is a BEC. Again, we have a contradiction and the upper bound on the complete information cannot be tight.

The word-wise complete information may be written as follows by applying the chain rule for mutual information [CT91]:

$$\begin{aligned} I_{\mathsf{wcmp}} &= I(\boldsymbol{X};\boldsymbol{Y}) \\ &= I(X_0;\boldsymbol{Y}) + I(X_1;\boldsymbol{Y}|X_0) + I(X_2;\boldsymbol{Y}|\boldsymbol{X}_{[0,1]}) + \\ &\quad + I(X_3;\boldsymbol{Y}|\boldsymbol{X}_{[0,2]}) + \cdots + I(X_{N-2};\boldsymbol{Y}|\boldsymbol{X}_{[0,N-3]}) \\ &= I(X_0;\boldsymbol{Y}) + I(X_1;\boldsymbol{Y}_{[1,N-1]}|X_0) + I(X_2;\boldsymbol{Y}_{[2,N-1]}|\boldsymbol{X}_{[0,1]}) + \\ &\quad + I(X_3;\boldsymbol{Y}_{[3,N-1]}|\boldsymbol{X}_{[0,2]}) + \cdots + I(X_{N-2};\boldsymbol{Y}_{[N-2,N-1]}|\boldsymbol{X}_{[0,N-3]}). \end{aligned} \tag{6.25}$$

In the latter expression, observations for given code symbols are omitted, because they do not contribute to the mutual information. Notice that

$$I(X_{N-1};Y_{N-1}|\boldsymbol{X}_{[0,N-2]}) = 0,$$

as X_{N-1} can be computed using $\boldsymbol{X}_{[0,N-2]}$ due to the parity-check constraint.

The first term in this sum is identical to the complete information on code symbol X_0, $I_{\mathsf{cmp},0}$. The second term corresponds to the complete information on a code symbol for a single parity check code of length $N-1$, because the code symbol X_0 is known. Proceeding in a similar way, it can be seen that the term $I(X_i;\boldsymbol{Y}_{[i,N-1]}|\boldsymbol{X}_{[0,i-1]})$ corresponds to the complete information on a code symbol for a single parity check code of length $N-i$, $i=0,1,\ldots,N-2$. Thus, we can use the bounds given in (6.22) and (6.23).

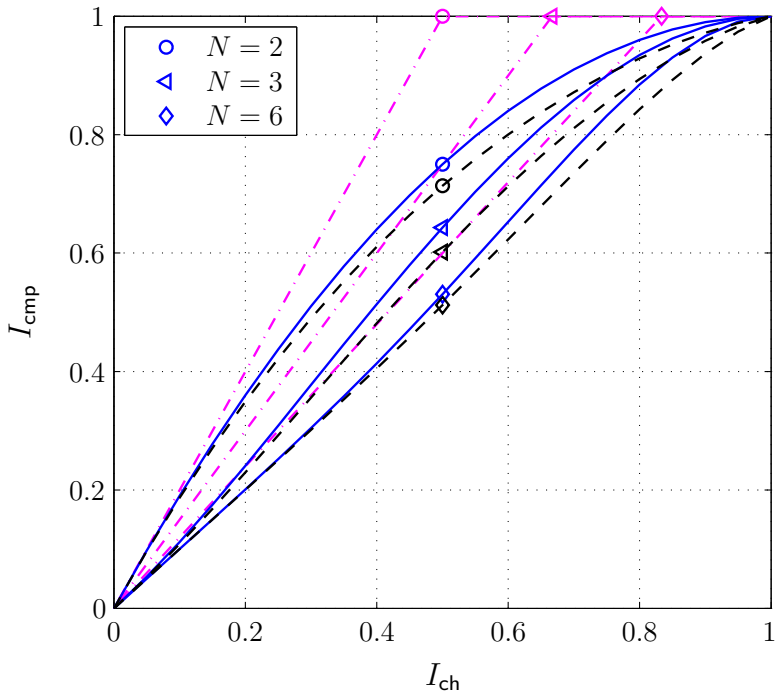


Figure 6.7: Bounds on CIT functions for single parity check codes of several code lengths N. Upper bounds: solid lines; lower bounds: dashed lines; ideal coding scheme: dash-dotted lines. (Mutual information is given in bit/use.)

When applying these bounds in (6.25), we obtain bounds on the word-wise CIT function:

$$\begin{aligned} I_{\mathsf{wcmp}} &\geq \frac{1}{K}\sum_{i=0}^{N-2} f_2^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{ch}}^{N-1-i}), \\ I_{\mathsf{wcmp}} &\leq \frac{1}{K}\sum_{i=0}^{N-2}\Big(1-(1-I_{\mathsf{ch}})(1-f_{N-1-i}^{\mathsf{ser}}(I_{\mathsf{ch}},\ldots,I_{\mathsf{ch}})\Big) = \\ &= 1-(1-I_{\mathsf{ch}})\Big(1-\frac{1}{K}\sum_{i=0}^{N-2} f_{N-1-i}^{\mathsf{ser}}(I_{\mathsf{ch}},\ldots,I_{\mathsf{ch}})\Big). \end{aligned} \tag{6.26}$$

Note that $K = N - 1$ for single parity check codes.

These bounds are plotted in Fig. 6.8. When comparing these results to those in Fig. 6.7, we see that the gap to the bound given by the ideal coding scheme is now relatively small as long as the channel information is smaller than about half the code rate. On the other hand, the upper bound (solid line) is slightly above the IPC of the ideal coding scheme. This means that this bound is not tight, as predicted above by theory.

The complete information for blocks of info symbols may be bounded using the same method as above. Basically, only the sum in (6.26) has to be truncated in an appropriate way.

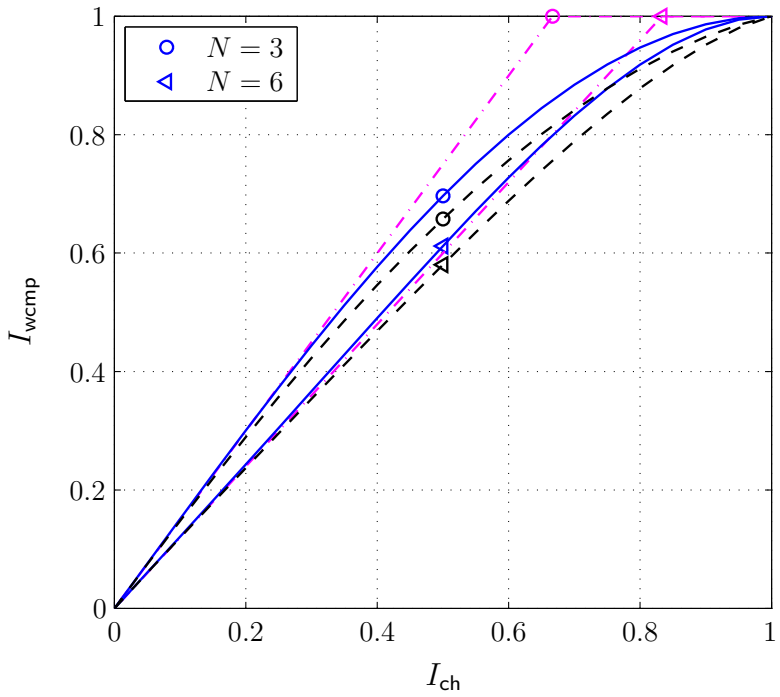


Figure 6.8: Bounds on word-wise CIT functions for single parity check codes of several code lengths N. Upper bounds: solid lines; lower bounds: dashed lines; ideal coding scheme: dash-dotted lines. (Mutual information is given in bit/use.)

6.3.2 Repetition Codes

Consider a repetition code of length N as defined in Section 6.2.2. In the sequel, we determine bounds on the EXIT functions and bounds on the CIT functions.

Bounds on EXIT Functions

Assume first that a-priori information is available for all code symbols and there is no channel information for any code symbol, i.e., $I_{\mathsf{int},i} = I_{\mathsf{apri}}$ for $i = 0, 1, \ldots, N-1$. This is the case when repetition codes are used as outer codes in serially concatenated coding schemes, like in repeat-accumulate codes or DRS codes (cf. Section 3.5.2).

The extrinsic information on code symbol X_0 can be bounded according to Theorem 6.3. Since the average extrinsic information is equal to the extrinsic information on code symbol X_0, we have the bounds

$$\begin{aligned} I_{\mathsf{ext}} &\geq f_{N-1}^{\mathsf{par}}(I_{\mathsf{apri}}, I_{\mathsf{apri}}, \ldots, I_{\mathsf{apri}}), \\ I_{\mathsf{ext}} &\leq 1 - (1 - I_{\mathsf{apri}})^{N-1}. \end{aligned} \tag{6.27}$$

These bounds are depicted in Fig. 6.9 for several code lengths.

Assume now that the channel information on code symbol X_{N-1} and a-priori information for all other code symbols are available, i.e., $I_{\mathsf{int},i} = I_{\mathsf{apri}}$ for $i = 0, 1, \ldots, N-2$ and $I_{\mathsf{int},N-1} = I_{\mathsf{ch}}$. This corresponds to the decoding operation for a variable node in the

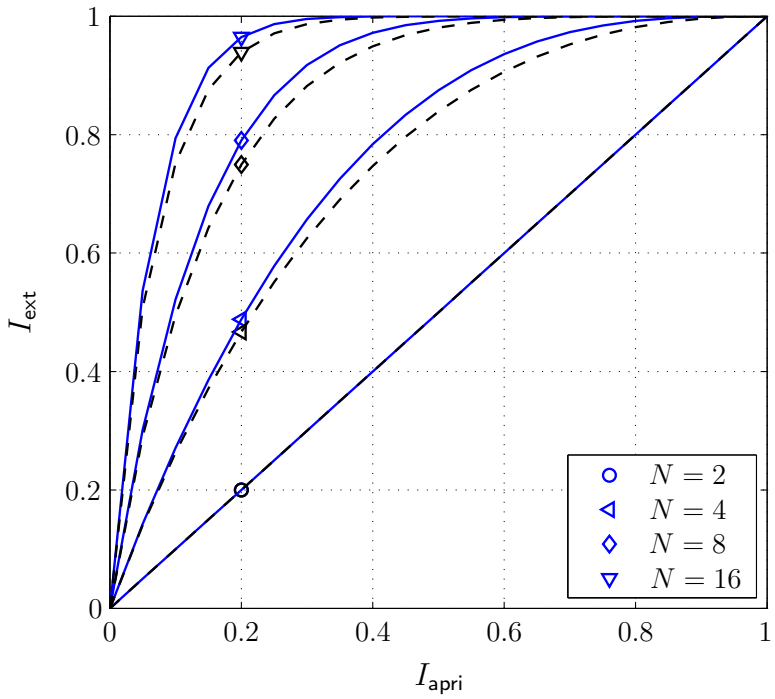


Figure 6.9: Bounds on EXIT functions for repetition codes of several code lengths N. Upper bounds (solid lines) correspond to BECs and lower bounds (dashed lines) correspond to BSCs. (Mutual information is given in bit/use.)

iterative decoder for a low-density parity-check code (cf. Section 3.6). It is also the case for repetition codes used in systematic repeat accumulate codes (cf. Section 3.5.2).

Using Theorem 6.3, we obtain the bounds

$$\begin{aligned} I_{\mathsf{ext}} &\geq f_{N-1}^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{apri}}, \ldots, I_{\mathsf{apri}}), \\ I_{\mathsf{ext}} &\leq 1 - (1 - I_{\mathsf{ch}})(1 - I_{\mathsf{apri}})^{N-2}. \end{aligned} \tag{6.28}$$

These bounds are depicted in Fig. 6.10. In contrast to the curves for the single parity check codes in Fig. 6.6, these curves start with $I_{\mathsf{ext}} = I_{\mathsf{ch}}$ (at $I_{\mathsf{apri}} = 0$) and end with $I_{\mathsf{ext}} = 1$ (at $I_{\mathsf{apri}} = 1$) for increasing a-priori information. The latter property makes these codes particularly suitable for iterative decoding.

Bounds on CIT Functions

For the CIT function, we assume that channel information is available for all code symbols, i.e., $I_{\mathsf{int},i} = I_{\mathsf{ch}}$ for $i = 0, 1, \ldots, N-1$. It is sufficient to discuss the complete information on code symbol X_0, as it is identical to both the symbol-wise complete information I_{cmp} and the word-wise complete information I_{wcmp}.

Applying Theorem 6.3, we obtain the bounds on the complete information:

$$\begin{aligned} I_{\mathsf{cmp}} &\geq f_{N}^{\mathsf{par}}(I_{\mathsf{ch}}, \ldots, I_{\mathsf{ch}}), \\ I_{\mathsf{cmp}} &\leq 1 - (1 - I_{\mathsf{ch}})^{N}. \end{aligned}$$

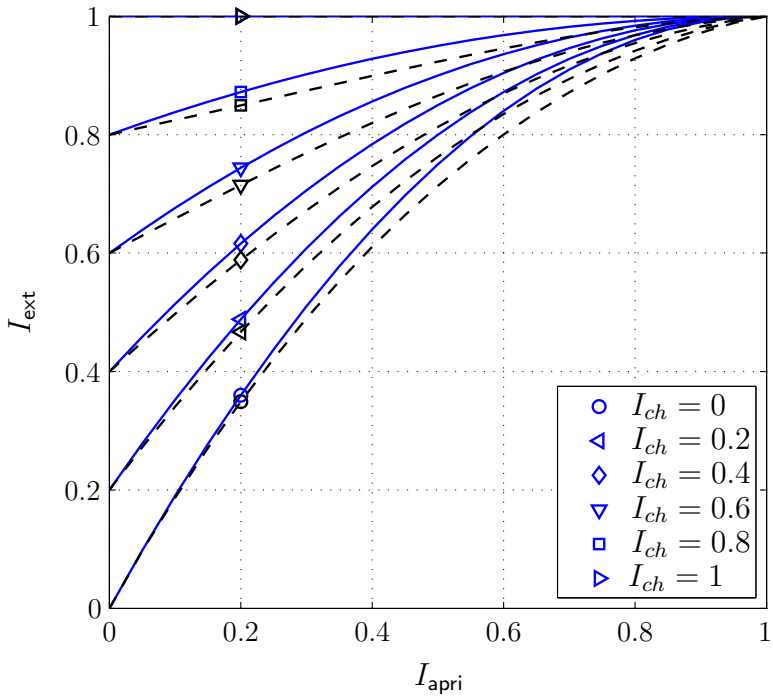


Figure 6.10: Bounds on EXIT functions for repetition codes of length $N = 4$ for several values of channel information I_{ch}. Upper bounds (solid lines) correspond to BECs and lower bounds (dashed lines) correspond to BSCs. (Mutual information is given in bit/use.)

Due to $I_{\mathsf{wcmp}} = I_{\mathsf{cmp}}$, the same bounds hold for the word-wise complete information.

These bounds are plotted in Fig. 6.11. Similarly to the bounds for the single parity check code in Fig. 6.7, we observe a large gap between the upper bound and the IPC of the ideal coding scheme, unless the channel information is very small or very large.

6.3.3 Accumulator

The accumulator is the recursive convolutional encoder with rate $R = 1$ and memory length $m = 1$, defined by the generator function

$$g(D) = \frac{1}{1+D}.$$

It is used, e.g., as the inner code of repeat-accumulate codes (cf. Section 3.5.2).

In the sequel, bounds on its EXIT function are determined. First, the decoding model is adapted to the given problem and the factor graph of the accumulator is introduced. Then, the bounds of information combining are applied on the factor graph in a recursive way. This method was partially presented in [LSH04].

The derivation of the bounds for the accumulator is more involved than for the single parity check codes and the repetition codes. However, the technique applied may have the potential to be extendable to other convolutional codes. Furthermore, an extension

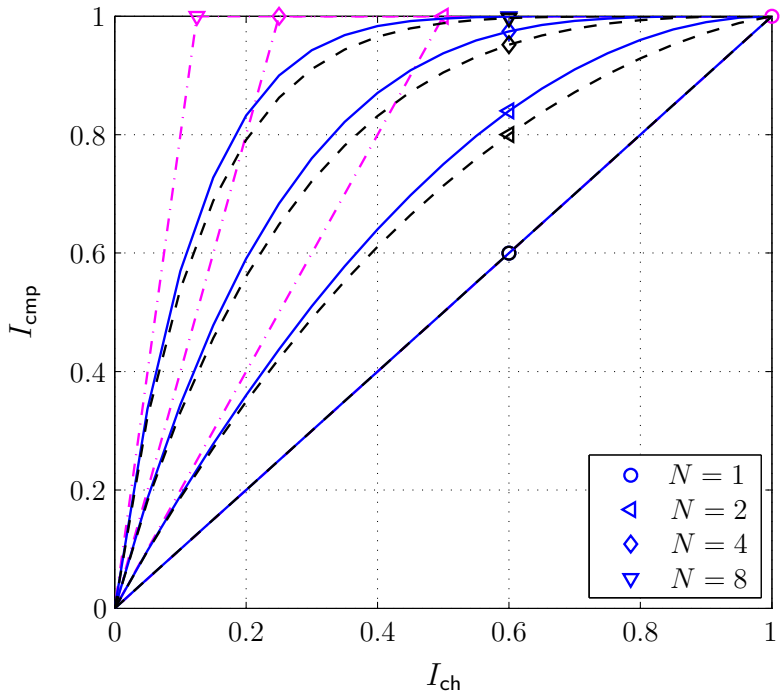


Figure 6.11: Bounds on CIT functions for repetition codes of several code word lengths N. Upper bounds (solid lines) correspond to BECs and lower bounds (dashed lines) correspond to BSCs; ideal coding schemes (dash-dotted lines). (Mutual information is given in bit/use.)

to other block codes may be possible, as all block codes can be represented in a trellis and can thus be interpreted as irregular convolutional codes.

Decoding Model

Following the decoding model from Section 3.2, the info word and the code word are denoted by $\boldsymbol{u} = [u_0, u_1, \ldots, u_{K-1}]$ and $\boldsymbol{x} = [x_0, x_1, \ldots, x_{K-1}]$. Their length K is assumed to be very large, i.e., $K \to \infty$. The info word $\boldsymbol{u}$ is transmitted over the a-priori channel; the received word is denoted by $\boldsymbol{y}_u = [y_{u,0}, y_{u,1}, \ldots, y_{u,K-1}]$. Similarly, the code word $\boldsymbol{x}$ is transmitted over the communication channel; the received word is denoted by $\boldsymbol{y}_x = [y_{x,0}, y_{x,1}, \ldots, y_{x,K-1}]$.

Therefore, a-priori information is available for all info symbols and channel information is available for all code symbols:

$$I(U_k; Y_{u,k}) = I_{\mathsf{apri}},$$
$$I(X_k; Y_{x,k}) = I_{\mathsf{ch}}$$

for $k = 0, 1, \ldots, K-1$. The extrinsic information on info symbol U_k is defined as

$$I_{\mathsf{ext},k} := I(U_k; \boldsymbol{Y}_x \boldsymbol{Y}_{u,\backslash k}).$$

Due to the regular code structure and the assumption of infinite length, the average extrinsic information I_{ext} tends to the extrinsic information on info symbols in the middle of the info word when the info word length approaches infinity, i.e.,

$$I_{\mathsf{ext}} \to I_{\mathsf{ext},K/2} \tag{6.29}$$

for $K \to \infty$. Therefore, the average extrinsic information can be bounded by bounding the extrinsic information on an info symbol U_k in the middle of the info word, based on the a-priori information and the channel information. If not stated otherwise, we assume $0 \ll k \ll K$ and $K \to \infty$, or for simplicity, $k \approx K/2$ and $K \to \infty$.

Factor Graph

The analysis of the accumulator is based on its factor graph (cf. Section 3.6). According to the generator function $g(D) = 1/(1+D)$, the info symbols and the code symbols are coupled by the parity check equation

$$\breve{u}_k \oplus \breve{x}_{k-1} = \breve{x}_k, \tag{6.30}$$

$k = 0, 1, \ldots, K-1$, where $\breve{x}_{-1} := 0$. This relation gives the factor graph shown in Fig. 6.12. Notice the similarity to Fig. 6.1, Fig. 6.2, and Fig. 6.4.

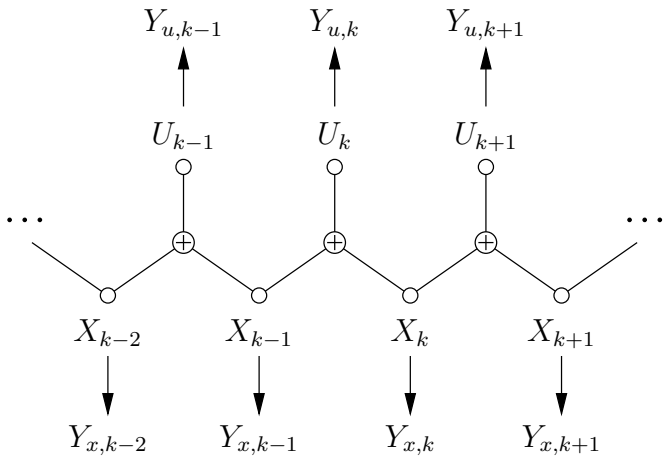


Figure 6.12: Factor graph of the accumulator.

Derivation of Bounds

Bounds on the extrinsic information for the accumulator that are only based on the channel information and on the a-priori information are given in the following theorem.

Theorem 6.5 (Bounds on Extrinsic Information for Accumulator)
Consider an accumulator, where the channel information I_{ch} is available for all code symbols and a-priori information I_{apri} is available for all info symbols. The extrinsic

information I_{ext} on info symbols is bounded as

$$I_{\min} \cdot I_{\min} \;\leq\; I_{\mathsf{ext}} \;\leq\; f_2^{\mathsf{ser}}(I_{\max}, I_{\max}),$$

where $I_{\min}$ is the minimum value and $I_{\max}$ is the maximum value $I \in [0,1]$ fulfilling simultaneously

$$I \geq f_2^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{apri}} \cdot I)$$

and

$$I \leq 1 - (1 - I_{\mathsf{ch}})\big(1 - f_2^{\mathsf{ser}}(I_{\mathsf{apri}}, I)\big).$$

For proving these bounds, Theorem 6.1 and Theorem 6.3 are applied in a nested way, and a stationarity condition is employed.

To start with, we write (6.30) as

$$\breve{u}_k = \breve{x}_{k-1} \oplus \breve{x}_k.$$

Accordingly, the extrinsic information on info symbol U_k,

$$I_{\mathsf{ext},k} := I(U_k; \boldsymbol{Y}_x \boldsymbol{Y}_{u,\setminus k}),$$

can be expressed using

$$I_{\alpha,k-1} \;:=\; I(X_{k-1}; \boldsymbol{Y}_{x,[0,k-1]} \boldsymbol{Y}_{u,[0,k-1]}), \tag{6.31}$$

$$I_{\beta,k} \;:=\; I(X_k; \boldsymbol{Y}_{x,[k,K-1]} \boldsymbol{Y}_{u,[k+1,K-1]}), \tag{6.32}$$

which can easily be seen from Fig. 6.12. As the first term corresponds to the forward recursion and the second term to the backward recursion in the BCJR algorithm [BCJR74], we label them with "α" and "β", respectively. Using Theorem 6.1, the extrinsic information can then be bounded as

$$I_{\alpha,k-1} \cdot I_{\beta,k} \;\leq\; I_{\mathsf{ext},k} \;\leq\; f_2^{\mathsf{ser}}(I_{\alpha,k-1}, I_{\beta,k}). \tag{6.33}$$

Consider first the mutual information corresponding to the *forward recursion*, $I_{\alpha,k-1}$. This information on code symbol X_{k-1} can be separated into the intrinsic information based on a direct observation, $I(X_{k-1}; Y_{x,k-1})$, and the (left-)extrinsic information based on indirect observations,

$$I_{\alpha\mathsf{ext},k-1} := I(X_{k-1}; \boldsymbol{Y}_{x,[0,k-2]} \boldsymbol{Y}_{u,[0,k-1]}).$$

This term is called left-extrinsic, because it denotes extrinsic information based only on observations left to X_{k-1} in the factor graph. Regarding that $I(X_{k-1}; Y_{x,k-1}) = I_{\mathsf{ch}}$ and applying Theorem 6.3, we obtain

$$I_{\alpha,k-1} \;\geq\; f_2^{\mathsf{par}}(I_{\mathsf{ch}}, , I_{\alpha\mathsf{ext},k-1}) \tag{6.34}$$

$$I_{\alpha,k-1} \;\leq\; 1 - (1 - I_{\mathsf{ch}})(1 - I_{\alpha\mathsf{ext},k-1}). \tag{6.35}$$

Due to the parity check equation

$$\breve{x}_{k-1} = \breve{u}_{k-1} \oplus \breve{x}_{k-2}$$

following from (6.30), the term $I_{\alpha\mathsf{ext},k-1}$ can be expressed using the intrinsic information on info symbol U_{k-1}, $I(U_{k-1};Y_{u,k-1}) = I_{\mathsf{apri}}$, and the information on code symbol X_{k-2},

$$I(X_{k-2};\boldsymbol{Y}_{x,[0,k-2]}\boldsymbol{Y}_{u,[0,k-2]}) = I_{\alpha,k-2};$$

the last identity can easily be deducted from (6.31). When applying Theorem 6.1, we obtain

$$I_{\mathsf{apri}} \cdot I_{\alpha,k-2} \;\leq\; I_{\alpha\mathsf{ext},k-1} \;\leq\; f_2^{\mathsf{ser}}(I_{\mathsf{apri}}, I_{\alpha,k-2}).$$

Substituting the lower bound into (6.34) and the upper bound into (6.35), we obtain bounds for the forward recursion:

$$I_{\alpha,k-1} \;\geq\; f_2^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{apri}} \cdot I_{\alpha,k-2}) \tag{6.36}$$

$$I_{\alpha,k-1} \;\leq\; 1-(1-I_{\mathsf{ch}})\big(1-f_2^{\mathsf{ser}}(I_{\mathsf{apri}}, I_{\alpha,k-2})\big). \tag{6.37}$$

Since the factor graph has a regular structure, we have $I_{\alpha,k-2} \to I_{\alpha,k-1}$ for $k \to K/2$ and $K \to \infty$. (We are interested in the extrinsic information "in the middle" of the factor graph.) Let

$$I_\alpha := \lim_{K\to\infty} I_{\alpha,K/2} \tag{6.38}$$

denote the stationary value of $I_{\alpha,k}$. When assuming *stationarity* in (6.36) and (6.37), we obtain

$$I_\alpha \;\geq\; f_2^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{apri}} \cdot I_\alpha), \tag{6.39}$$

$$I_\alpha \;\leq\; 1-(1-I_{\mathsf{ch}})\big(1-f_2^{\mathsf{ser}}(I_{\mathsf{apri}}, I_\alpha)\big). \tag{6.40}$$

These two relations are necessary conditions for possible stationary values I_α. Thus, we have the following lemma.

Lemma 6.3 (Forward Recursion)
Let $I_{\alpha,k}$ be defined according to (6.31)*, and let I_α be defined as the stationary value of $I_{\alpha,k}$ according to* (6.38)*. Bounds on I_α are given by the minimum and the maximum value of $I_\alpha \in [0,1]$, fulfilling simultaneously* (6.39) *and* (6.40)*.*

Consider now the mutual information corresponding to the *backward recursion*, $I_{\beta,k}$. Since the factor graph is symmetric with respect to k, it can easily be seen that the analysis for the backward recursion is identical to the one for the forward recursion. Let

$$I_\beta := \lim_{K\to\infty} I_{\beta,K/2} \tag{6.41}$$

denote the stationary value. The necessary conditions for possible stationary values I_β are the same as for I_α, i.e., they are given in (6.39) and (6.40). Thus, we have the following lemma.

Lemma 6.4 (Backward Recursion)
Let $I_{\beta,k}$ be defined according to (6.32)*, and let I_β be defined as the stationary value of $I_{\beta,k}$ according to* (6.41)*. Bounds on I_β are given by the minimum and the maximum value of $I_\alpha \in [0,1]$, fulfilling simultaneously* (6.39) *and* (6.40)*.*

The bounds on I_α and on I_β according to Lemma 6.3 and Lemma 6.4 can now be used to bound the extrinsic information $I_{\mathsf{ext},K/2}$. Using the minimum values for I_α and I_β in the lower bound in (6.33), we obtain a lower bound on the extrinsic information. On the other hand, using the maximum values of I_α and I_β in the upper bound in (6.33), we obtain an upper bound on the extrinsic information. Considering that the two minimum values are equal and that the two maximum values are equal, as discussed above, and taking into account (6.29), we have the proof of Theorem 6.5.

Illustration

The bounds on the extrinsic information according to Theorem 6.5 depend only on the channel information and on the a-priori information. Accordingly, they represent bounds on the EXIT functions of the accumulator. The corresponding EXIT chart is depicted in Fig. 6.13.

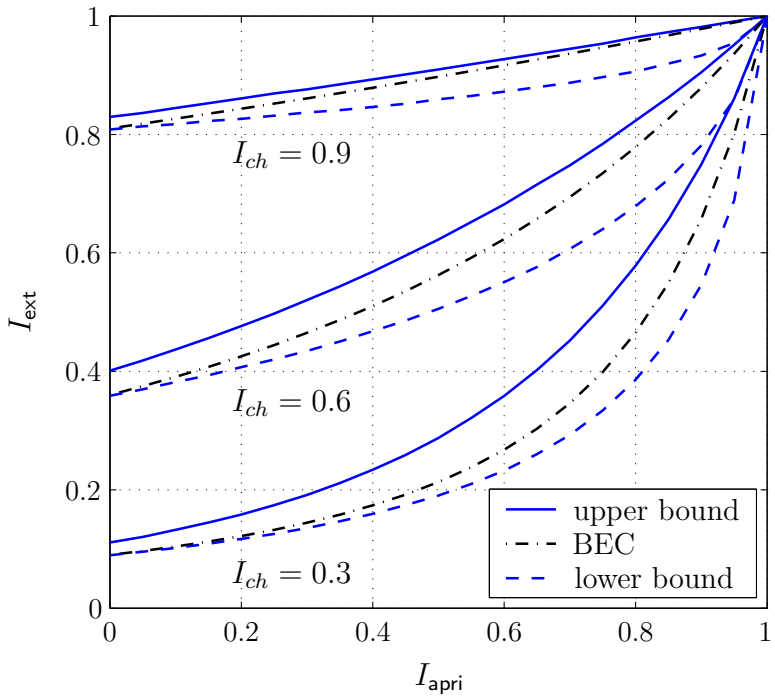


Figure 6.13: Bounds on the EXIT functions of the accumulator and exact EXIT functions for BECs; several values of channel information I_{ch}. (Mutual information is given in bit/use.

It can be seen that the bounds are close to each other only for small and for large a-priori information. This can be explained as follows. For the derivation of the bounds, we applied Theorem 6.1 and Theorem 6.3 in a nested way. The lower bound for repetition codes is achieved if both channels are BSCs, and the lower bound for single parity check codes is achieved if both channels are BECs. Thus, if the two lower bounds are applied in a nested way, the assumptions contradict each other and the resulting lower bound may

be too pessimistic. Similarly, this holds for a nested upper bound, and thus the upper bound may be too optimistic. Therefore, the bounds given in Theorem 6.5 are not tight and may have a potential for improvement.

The extrinsic information can be computed exactly when both the communication channel and the a-priori channel are BECs. We simply have to use the expression corresponding to the case of BECs each time Theorem 6.1 and Theorem 6.3 are applied in the derivations. Doing so, we obtain the condition for the stationary value,

$$I_{\alpha}^{\mathsf{BEC}} = I_{\mathsf{ch}} + I_{\mathsf{apri}} \cdot I_{\alpha}^{\mathsf{BEC}} - I_{\mathsf{ch}} \cdot I_{\mathsf{apri}} \cdot I_{\alpha}^{\mathsf{BEC}}, \tag{6.42}$$

corresponding to (6.39) and (6.40). After solving for I_{α} and applying Theorem 6.1, we obtain

$$I_{\mathsf{ext}}^{\mathsf{BEC}} = \left(I_{\alpha}^{\mathsf{BEC}}\right)^2 = \left(\frac{I_{\mathsf{ch}}}{1-(1-I_{\mathsf{ch}})I_{\mathsf{apri}}}\right)^2. \tag{6.43}$$

This result is also reported in [tBK03] (see also the references therein).

These EXIT functions are plotted in Fig. 6.13. When the channel information I_{ch} is small, the BEC curves are close to the lower bounds for small I_{apri}, and they are close to the upper bounds for large I_{apri}. Therefore, the case where channels are all BECs may not be an extreme case for the accumulator, as opposed to single parity check codes and repetition codes.

6.4 Application to LDPC Codes

The bounds on information combining may be used to analyze the iterative decoder of low-density parity-check codes (LDPCCs). We consider first the EXIT charts comprising the bounds on the EXIT functions for variable nodes and check nodes. Then, analytical bounds on decoding thresholds are derived.

6.4.1 EXIT Charts

EXIT functions were originally introduced to analyze the behavior of an iterative decoder comprising two or more constituent decoders by means of the EXIT chart method[5] [tB01c, tB01a]. This method was shown to predict quite accurately the decoding threshold of iterative decoders using only the EXIT functions of the constituent decoders.

EXIT functions depend on the model for the a-priori channel and the communication channel. This has two implications:

(a) The predicted decoding threshold is accurate only if the model for the a-priori channel is correct. Although the often used AWGN channel provides a good approximation of the actual a-priori channel, it is still an approximation. Thus, the predicted decoding thresholds cannot be proved to be accurate.

(b) The decoding threshold depends on the model for the communication channel.

[5]The EXIT chart method is explained for parallel concatenated codes, serially concatenated codes, and LDPCCs in Chapter 3.

The EXIT functions for specific channel models may be replaced in the EXIT chart method by bounds on EXIT functions that are valid for all BISMCs. This way, bounds on the true decoding threshold can be obtained, and these bounds are valid for all models of communication channels that are BISMCs. In the sequel, this method is applied to regular LDPCCs[6]. The bounds on the corresponding EXIT functions have been derived in Section 6.3.1 and Section 6.3.2. This work was partially published in [LHH04a]. Similar results were independently found in [SSSZ03, SSSZ05], where irregular LDPCCs were addressed as well.

Consider a regular LDPCC with variable node degree d_{v} and check node degree d_{c}. The code symbols are transmitted over a communication channel that is a BISMC and has mutual information I_{ch}, called channel information. The iterative decoder operates on the factor graph of the parity check matrix of the code, and LogAPP decoding is applied in the constituent decoders.

The *decoding operation in a variable node* is equivalent to extrinsic soft-output decoding of a repetition code of length $d_{\mathsf{v}}+1$, where the decoder has channel information I_{ch} about one code symbol and a-priori information $I_{\mathsf{apri}}^{(\mathsf{v})}$ about the other code symbols; the extrinsic information is denoted by $I_{\mathsf{ext}}^{(\mathsf{v})}$. (The a-priori information and the extrinsic information for the variable node are labeled with "v".) Accordingly, we can give bounds on the EXIT functions using (6.28):

$$I_{\mathsf{ext}}^{(\mathsf{v})} \geq f_{d_{\mathsf{v}}}^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{apri}}^{(\mathsf{v})}, \ldots, I_{\mathsf{apri}}^{(\mathsf{v})}), \tag{6.44}$$

$$I_{\mathsf{ext}}^{(\mathsf{v})} \leq 1-(1-I_{\mathsf{ch}})(1-I_{\mathsf{apri}}^{(\mathsf{v})})^{d_{\mathsf{v}}-1}. \tag{6.45}$$

The lower bound corresponds to BSCs, and the upper bound corresponds to BECs.

Similarly, the *decoding operation in a check node* is equivalent to extrinsic soft-output decoding of a single parity check code of length d_{c}, where the decoder has a-priori information $I_{\mathsf{apri}}^{(\mathsf{c})}$ about all code symbols; the extrinsic information is denoted by $I_{\mathsf{ext}}^{(\mathsf{c})}$. (The a-priori information and the extrinsic information for the check node are labeled with "c".) Thus, we can give bounds on the EXIT functions using (6.20):

$$I_{\mathsf{ext}}^{(\mathsf{c})} \geq (I_{\mathsf{apri}}^{(\mathsf{c})})^{d_{\mathsf{c}}-1}, \tag{6.46}$$

$$I_{\mathsf{ext}}^{(\mathsf{c})} \leq f_{d_{\mathsf{c}}-1}^{\mathsf{ser}}(I_{\mathsf{apri}}^{(\mathsf{c})}, I_{\mathsf{apri}}^{(\mathsf{c})}, \ldots, I_{\mathsf{apri}}^{(\mathsf{c})}). \tag{6.47}$$

The lower bound corresponds to BECs, and the upper bound corresponds to BSCs.

Due to the iterative decoder structure, the extrinsic information of the check nodes is equal to the a-priori information of the variable nodes in the following decoding step, and vice versa:

$$I_{\mathsf{ext}}^{(\mathsf{v})} = I_{\mathsf{apri}}^{(\mathsf{c})}, \qquad I_{\mathsf{ext}}^{(\mathsf{c})} = I_{\mathsf{apri}}^{(\mathsf{v})}. \tag{6.48}$$

Thus, iterative decoding proceeds in between the two EXIT functions (cf. Section 3.6).

Example 6.1
Consider a regular LDPC code with variable node degree $d_v = 3$ and check node degree $d_c = 4$, having design rate $R_d = 1/4$. The EXIT chart of this code is depicted

[6]LDPCCs, factor graphs, and the iterative decoding algorithm are explained in Section 3.6.

in Fig. 6.14 for two values of channel information I_{ch}. The EXIT function for the check node is flipped; therefore the upper curve corresponds to the lower bound on the EXIT function, and vice versa. ◇

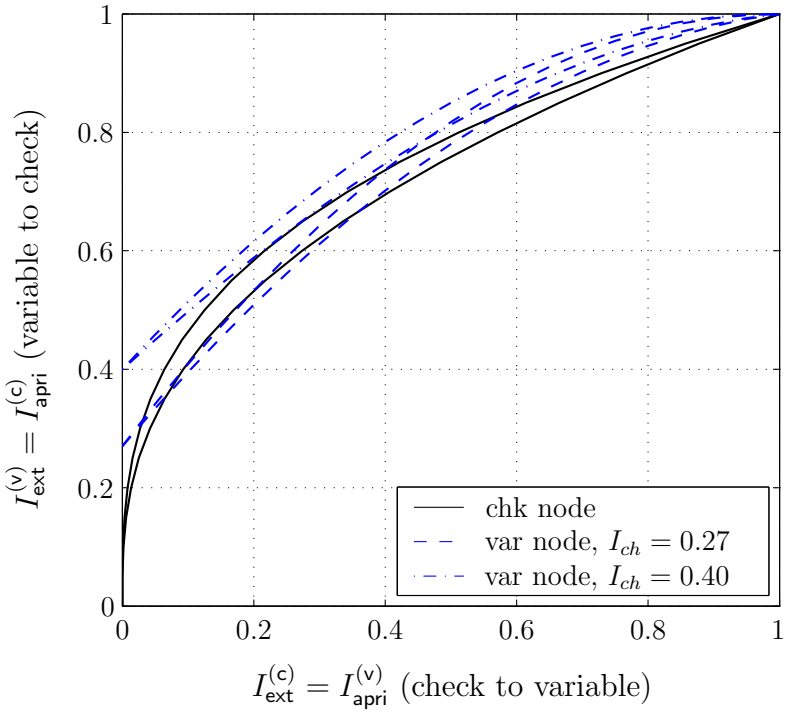


Figure 6.14: EXIT chart for an LDPC code with variable node degree $d_{\mathsf{v}} = 3$ and check node degree $d_{\mathsf{c}} = 4$ for channel information $I_{\mathsf{ch}} = 0.27$ and $I_{\mathsf{ch}} = 0.40$; bounds on the EXIT functions for variable nodes and check nodes are depicted; Example 6.1. (Mutual information is given in bit/use.)

6.4.2 Bounds on Decoding Thresholds

The bounds on the EXIT functions for variable nodes and check nodes can be used to determine the smallest channel information that is necessary for convergence, and the smallest channel information that is sufficient for convergence. Thus, a necessary and a sufficient condition for convergence are obtained, and these conditions are given in terms of the mutual information (or capacity) of the communication channel. These bounds on the convergence threshold are valid for all a-priori channels and all communication channels that are BISMCs.

Example 6.2
We continue Example 6.1. For $I_{\mathsf{ch}} = 0.27$, the upper bound on the check node EXIT function and the upper bound on the variable node EXIT function intersect. Therefore, the decoder cannot converge for any model of the a-priori and the communication channel if $I_{\mathsf{ch}} < 0.27$, and we have a lower bound on the convergence threshold.

For $I_{\mathsf{ch}} = 0.40$, the lower bound on the check node EXIT function and the lower bound on the variable node EXIT function do not intersect. Accordingly, the decoder will converge for any model of the a-priori and the communication channel if $I_{\mathsf{ch}} > 0.40$, and we have an upper bound on the convergence threshold. $\diamond$

The following theorem is based on this concept and gives a lower and an upper bound on the decoding threshold, and thus a necessary and a sufficient condition for convergence, respectively. We consider LDPCCs of infinite code length, $N \to \infty$, as usually assumed in the EXIT chart method; the codes are iteratively decoded on their factor graphs using LogAPP decoding in the variable and the check node decoders.

Theorem 6.6 (Bounds on Convergence Threshold for LDPC Codes)
Consider a binary LDPC code with variable node degree d_{v} and check node degree d_{c}. The communication channel is assumed to be an arbitrary BISMC with mutual information (capacity) I_{ch}. Let $I_{\mathsf{ch},low}$ be defined as the maximum value $I_{\mathsf{ch}} \in [0,1]$ such that there is an $I \in [0,1]$ fulfilling

$$I = 1 - (1 - I_{\mathsf{ch}})\big(1 - f_{d_{\mathsf{c}}-1}^{\mathsf{ser}}(I, I, \ldots, I)\big)^{d_{\mathsf{v}}-1}.$$

Let further $I_{\mathsf{ch},upp}$ be defined as the maximum value $I_{\mathsf{ch}} \in [0,1]$ such that there is an $I \in [0,1]$ fulfilling

$$I = f_{d_{\mathsf{v}}-1}^{\mathsf{par}}(I_{\mathsf{ch}}, I^{d_{\mathsf{c}}-1}, \ldots, I^{d_{\mathsf{c}}-1}).$$

The iterative decoder can converge only if $I_{\mathsf{ch}} > I_{\mathsf{ch},low}$ (necessary condition), and it converges for sure if $I_{\mathsf{ch}} > I_{\mathsf{ch},upp}$ (sufficient condition).

The first equality corresponds to an intersection of the upper bounds on the EXIT functions, and the second equality corresponds to an intersection of the lower bounds on the EXIT functions. These intersections of EXIT functions are the basis for proving Theorem 6.6:

Proof. Consider first the case that the upper bounds on the EXIT functions have an intersection. The upper bounds are given by (6.45) and (6.47); regarding (6.48), they may be written as

$$\begin{aligned} I_{\mathsf{ext}}^{(\mathsf{v})} &= 1 - (1 - I_{\mathsf{ch}})(1 - I_{\mathsf{ext}}^{(\mathsf{c})})^{d_{\mathsf{v}}-1}, \\ I_{\mathsf{ext}}^{(\mathsf{c})} &= f_{d_{\mathsf{c}}-1}^{\mathsf{ser}}(I_{\mathsf{ext}}^{(\mathsf{v})}, I_{\mathsf{ext}}^{(\mathsf{v})}, \ldots, I_{\mathsf{ext}}^{(\mathsf{v})}). \end{aligned}$$

An intersection of the EXIT functions corresponds to a fixed point of these two equations. Substituting the second equation into the first one and using $I := I_{\mathsf{ext}}^{(\mathsf{v})}$, we obtain the fixed point condition

$$I = 1 - (1 - I_{\mathsf{ch}})\big(1 - f_{d_{\mathsf{c}}-1}^{\mathsf{ser}}(I, I, \ldots, I)\big)^{d_{\mathsf{v}}-1},$$

where the fixed point is defined by $I \in [0,1]$. When a fixed point exists for a given channel information I_{ch}, the upper bounds on the EXIT functions intersect and convergence of the iterative decoder is impossible. Therefore, the channel information has to be larger than the maximum value I_{ch} for which a fixed point exists, such that convergence is possible. This proves the first part of the theorem.

Consider now the case that the lower bounds on the EXIT functions have an intersection. The lower bounds are given by (6.44) and (6.46); regarding (6.48), they may be written as

$$\begin{aligned} I_{\mathsf{ext}}^{(\mathsf{v})} &= f_{d_{\mathsf{v}}-1}^{\mathsf{par}}(I_{\mathsf{ch}}, I_{\mathsf{ext}}^{(\mathsf{c})}, \ldots, I_{\mathsf{ext}}^{(\mathsf{c})}), \\ I_{\mathsf{ext}}^{(\mathsf{c})} &= (I_{\mathsf{ext}}^{(\mathsf{v})})^{d_{\mathsf{c}}-1}. \end{aligned}$$

An intersection of the EXIT functions corresponds to a fixed point of these two equations. Substituting the second equation into the first one and using $I := I_{\mathsf{ext}}^{(\mathsf{v})}$, we obtain the fixed point condition

$$I = f_{d_{\mathsf{v}}-1}^{\mathsf{par}}(I_{\mathsf{ch}}, I^{d_{\mathsf{c}}-1}, \ldots, I^{d_{\mathsf{c}}-1}),$$

where the fixed point is defined by $I \in [0, 1]$. When a fixed point exists for a given channel information I_{ch}, the lower bounds on the EXIT functions intersect and convergence of the iterative decoder cannot be guaranteed (but may be possible). Therefore, the channel information has to be larger than the maximum value I_{ch} for which a fixed point exists, such that convergence can be guaranteed. This proves the second part of the theorem. QED

Theorem 6.6 may be used to compute the upper and the lower bound on the decoding threshold numerically. The results are usually more precise than those determined graphically using the conventional EXIT chart method. Notice that the lower bound corresponds to a communication channel that is a BSC, and that the upper bound corresponds to a communication channel that is a BEC.

Example 6.3
We continue Example 6.2. The numerical solutions for the decoding threshold are $I_{\mathsf{ch,low}} = 0.278$ and $I_{\mathsf{ch,upp}} = 0.398$. ◇

The necessary condition of convergence may be used to state whether a code can achieve capacity. If the design rate is smaller than the lower bound on the channel information that is necessary for convergence, the code cannot achieve the capacity of any communication channel that is a BISMC.

Example 6.4
We continue Example 6.3. The design rate of the code is $R_d = 1/4$, and the lower bound on the decoding threshold is $I_{\mathsf{ch,low}} = 0.278$. Since $R_d < I_{\mathsf{ch,low}}$, the given code is not capacity achieving for any communication channel that is a BISMC. ◇

In a similar way as for LDPCCs, bounds on the decoding thresholds of repeat-accumulate codes may be determined using the bounds on the EXIT functions of repetition codes and of the accumulator.

6.5 Summary

When channels are coupled by code constraints on their inputs, the mutual information between one channel input and a vector of channel outputs may be represented by a combination of the mutual information values of the individual channels. For binary-input

symmetric memoryless channels (BISMCs), the combined information has been bounded for equality constraints and for parity-check constraints on their inputs. To prove these bounds, the concept of decomposing BISMCs into binary symmetric subchannels has been applied. Several applications of these bounds on information combining have been presented. EXIT functions for repetition codes, single parity check codes, and the accumulator have been bounded. The bounds for single parity check codes and for repetition codes are tight, whereas the bounds derived for the accumulator may be improved. Furthermore, bounds on the decoding thresholds for low-density parity-check codes have been determined. These bounds are given in terms of the mutual information of the communication channel, and they represent necessary conditions and sufficient conditions for convergence of the iterative decoder, respectively. In particular, these bounds are valid for all BISMCs, and they do not rely on the assumption of Gaussian distributed extrinsic soft-values, as in the original EXIT chart method.

The presented concepts to bound combined information are rather general. Extensions may lead to similar bounds for other block or convolutional codes, giving further insights into the convergence behavior of iterative decoders.

Chapter 7

Conclusions

Various aspects of reliability information in channel decoding have been studied in this thesis, including practical aspects and information theoretical bounds. The transmission systems under consideration comprise linear binary channel encoders, symmetric memoryless communication channels, and non-iterative or iterative symbol-by-symbol soft-output decoders.

The investigations began with properties of symmetric memoryless channels, a general model for symbol-by-symbol soft-output decoding, and a unified view of coding schemes with iterative decoders. Based on these fundamentals, the quality of reliability information has been addressed, and true reliability information has been exploited for estimating transmission quality parameters. Furthermore, soft-output decoders have been interpreted as processors for mutual information, assuming true reliability information at the input and at the output of the decoders. Bounds on the decoding behavior have been derived by means of methods from information theory, using bounds on information combining.

The following sections summarize the key concepts and results presented in this thesis, and discuss further applications and possible extensions.

Summary

Every *binary-input symmetric memoryless channel* (BISMC) can be decomposed into subchannels that are binary symmetric channels (BSCs). Correspondingly, a BISMC can be characterized by the distribution of the subchannel error probabilities, the subchannel reliability values, or the subchannel mutual information values, where independent and uniformly distributed channel inputs are assumed. These distributions have been called the error probability profile, the reliability value profile, and the mutual information profile of the particular BISMC. The decomposition into BSCs and the three profiles have been applied throughout this thesis to prove properties and to interpret results.

A *general model for symbol-by-symbol soft-output decoding* has been introduced. This model assumes that both the info symbols and the code symbols are transmitted over BISMCs and that the noisy observations are converted to log-likelihood ratios (LLRs) before being passed to the decoder. Based on these assumptions, the decoder computes soft-values for info symbols or code symbols. The superchannel between the encoder input

and the decoder output may be modeled as a BISMC, when the memory introduced by the code and the decoder is not of interest. This decoding model has been specified to LogAPP decoding and to MaxLogAPP decoding, and it has been applied to non-iterative decoders and to constituent decoders of iterative decoders.

The encoders and the decoders for *parallel concatenated codes, serially concatenated codes and low-density parity-check codes* have been reviewed in a unified way. In particular, the structures of the parity check matrices have been related to the structures of the iterative decoders. For parallel and serially concatenated codes, parity check matrices have been derived using only the generator matrices, the inverse generator matrices, and parity check matrices of the constituent codes.

Each soft-value for an info symbol or a code symbol has been associated with the conditional log-likelihood ratio (LLR) for this symbol given the soft-value. The magnitude of this LLR has been called the *reliability value* of the soft-value. Following this approach, the *quality of reliability information* has been associated with the difference between soft-value magnitudes and the corresponding reliability values. This difference has been denoted as *reliability mismatch*, since the closer the magnitude of a soft-value is to the reliability value, the more reliable is this soft-value. Motivated by the fact that an LLR parameterizes the probability distribution of a binary symbol, a new measure for the reliability mismatch based on the Kullback-Leibler distance has been introduced.

A general framework has been presented for reducing the reliability mismatch by memoryless post-processing of the soft-values, and thus for *improving the reliability information*. This post-processing is based on a parameterized function, where the parameters are determined such that the reliability mismatch is minimized. This concept has been applied to MaxLogAPP decoding and to iterative decoding of parallel concatenated codes.

Reliability information can be exploited for *estimating transmission quality parameters*, like the bit error probability or the symbol-wise mutual information, provided that the soft-values are a-posteriori LLRs, i.e., that there is no reliability mismatch. This estimation method is unbiased. Furthermore, it is only based on the magnitudes of the a-posteriori LLRs and therefore requires no knowledge of the transmitted info or code symbols. A general framework for this kind of estimation has been introduced, and it has been specifically applied to the estimation of the bit error probability and to the estimation of the symbol-wise mutual information between info or code symbols and the soft-values at the decoder output. The estimation variance of the proposed method has been shown to be smaller than the estimation variance of the conventional method. This method may be applied in a practical receiver, since it requires no knowledge of the transmitted data. Due to the smaller estimation variance, this method may also advantageously be applied in simulations to improve or speed up the estimation of such a parameter.

Channel decoding has been interpreted as *combining and processing of mutual information*. Following this interpretation, a decoder may be characterized by an information transfer function, which is the mapping from a mutual information value associated with the decoder inputs to a mutual information value associated with the decoder outputs. *Information transfer functions* describe the information processing capability of a decoder for the case that there is no reliability mismatch at the decoder input and at the decoder output, i.e., for the case that all soft-values are LLRs. Two kinds of information transfer

functions have been addressed, namely information processing characteristics (IPCs) and extrinsic information transfer (EXIT) functions.

The concept of bounding combined information has been introduced, and *bounds on information combining* have been derived for single parity check codes and for repetition codes. Based on this concept, information transfer functions for single parity check codes, for repetition codes, and for the accumulator have been bounded. These bounds are valid for all BISMCs, as opposed to conventional information transfer functions, which depend on the channel model. Finally, decoding thresholds of regular low-density parity-check codes have been bounded, using an approach based on the conventional EXIT chart method and bounds on information transfer functions; the communication channel has been assumed to be any arbitrary BISMC. As opposed to the conventional EXIT chart method, these bounds on the decoding thresholds do not rely on a particular model for the communication channel, and the virtual a-priori channel need not be modeled by an AWGN channel.

Further Applications and Extensions

The considerations in this thesis have been restricted to binary channel codes and memoryless channels. The application and extension to soft-output equalizers, soft-output demappers, and nonbinary codes is addressed in the following.

The concepts regarding the measurement and the improvement of soft-values may directly be applied to soft-output equalizers or soft-output demappers. This holds similarly for the estimation of transmission quality parameters, provided that the soft-values at the output of the equalizer or the demapper are log-likelihood ratios.

An extension to nonbinary symbols would certainly be of great interest, regarding channel coding, equalization and demapping. A reliability mismatch could also be defined based on the Kullback-Leibler distance, whereas the correction of soft-values by memoryless post-processing seems to be more involved. The estimation of transmission quality parameters, however, may immediately be extended to the nonbinary case.

The bounds on information combining could be extended in two ways. On the one hand, binary codes that are less simple than single parity check codes and repetition codes could be considered. On the other hand, nonbinary single parity check codes and repetition codes could be investigated. Other interesting issues are tighter bounds on information transfer functions for the accumulator and extensions to other convolutional codes.

This thesis has focused on binary channel codes and memoryless channels, but the presented principles and concepts show the potential to be extended in many directions, as outlined above. Such extensions may give further insights into the use of reliability information in channel decoding.

Appendix A

Acronyms

APP	a-posteriori probability
AWGN	additive white Gaussian noise
BEC	binary erasure channel
BER	bit error rate (error rate of binary symbols)
BI-AWGNC	binary-input AWGN channel
BISMC	binary-input symmetric memoryless channel
BPSK	binary phase-shift keying
BSC	binary symmetric channel
BSEC	binary symmetric erasure channel
CIT	complete (mutual) information transfer (... function)
DIMC	discrete-input memoryless channel
DISMC	discrete-input symmetric memoryless channel
DMC	discrete memoryless channel
EXIT	extrinsic (mutual) information transfer (... function)
IPC	(mutual) information processing characteristic
KLD	Kullback-Leibler distance
LDPC	low-density parity-check
LDPCC	low-density parity-check code
LLR	log-likelihood ratio
LogAPP	logarithmic a-posteriori probability (... decoder)
MAP	maximum a-posteriori
MaxLogAPP	maximum logarithmic a-posteriori probability (... decoder)
MI	mutual information
ML	maximum likelihood
PCC	parallel concatenated code
SCC	serially concatenated code
SER	symbol error rate
SNR	signal-to-noise ratio
i.i.d.	independent and identically distributed
i.u.d.	independent and uniformly distributed

Appendix B

Notation

Constants and Number Sets

e	Euler number
$\mathbf{0}$	all-zero vector or all-zero matrix[1]
$\mathbf{1}$	all-one vector or all-one matrix
$\mathbb{B}$	set of binary elements -1 and $+1$
$\mathbb{F}_2$	set of binary elements 0 and 1
$\mathbb{N}$	set of natural numbers including 0
$\mathbb{N}^+$	set of natural numbers without 0
$\mathbb{Q}$	set of rational numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^+$	set of positive real numbers without 0
$\mathbb{R}_0^+$	set of positive real numbers including 0

Functions and Operators

$(.)^{\mathsf{T}}$	transpose of a matrix
abs(.)	absolute value (magnitude) of real value
$f^{\mathsf{ser}}(.)$	binary information function for serial concatenation of BSCs
$f^{\mathsf{par}}(.)$	binary information function for parallel concatenation of BSCs
dec(.)	(soft-output) decoding function
logapp(.)	LogAPP decoding function
maxlogapp(.)	MaxLogAPP decoding function
$D(.\|\|.)$	Kullback-Leibler distance
enc(.)	encoding function
E{.}	expected value
$L(.)$	log-likelihood ratio
$h(.)$	binary entropy function

[1] The meaning becomes clear from the context.

$h^{-1}(.)$	inverse of binary entropy function
$H(.)$	entropy
itf(.)	information transfer function
$I(.;.)$	mutual information
ld(.)	logarithm dualis (base 2)
ln(.)	natural logarithm (base e)
$p(.)$	probability mass function for discrete random variables and probability density function for continuous random variables
Pr(.)	probability of event
perm_π	permutation operator (see below)
sgn(.)	sign of real value

Permutation Function and Permutation Operator

A permutation function π on an index set $\mathcal{I} = \{0, 1, \ldots, L-1\}$ is defined as a bijective map

$$\pi : \mathcal{I} \to \mathcal{I}.$$

Let $\boldsymbol{a}$ denote a (row) vector with indices from $\mathcal{I}$,

$$\boldsymbol{a} = [a_i]_{i \in \mathcal{I}} = [a_0, \ldots, a_{L-1}].$$

The permutation operator for a given permutation function π is defined as

$$\text{perm}_\pi [a_0, \ldots, a_{L-1}] = [a_{\pi(0)}, \ldots, a_{\pi(L-1)}].$$

Using this notation, the permuted vector (also called interleaved vector)

$$\boldsymbol{a}_\pi := [a_{\pi(0)}, \ldots, a_{\pi(L-1)}],$$

may shortly be written as

$$\boldsymbol{a}_\pi = \text{perm}_\pi \, \boldsymbol{a}.$$

Alternatively, the permutation of vector $\boldsymbol{a}$ may be expressed using a permutation matrix

$$\boldsymbol{P} \in \mathbb{R}^{L \times L}.$$

Such a permutation matrix has exactly one 1 in each row and each column, and all other entries are 0. The permutation matrix corresponding to a permutation function π is defined such that

$$\boldsymbol{a}\boldsymbol{P} = \text{perm}_\pi \, \boldsymbol{a}.$$

Example B.1
Consider the two vectors

$$\begin{aligned} \boldsymbol{a} &= [a_0, a_1, a_2] := [10, 20, 30], \\ \boldsymbol{b} &= [b_0, b_1, b_2] := [20, 30, 10] \end{aligned}$$

with the index set $\mathcal{I} = \{0, 1, 2\}$. The permutation function π defined as

$$\begin{aligned} 0 &\mapsto \pi(0) = 1, \\ 1 &\mapsto \pi(1) = 2, \\ 2 &\mapsto \pi(2) = 0 \end{aligned}$$

leads to

$$\boldsymbol{b} = \mathrm{perm}_{\pi}\, \boldsymbol{a}.$$

Similarly, the permutation matrix

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

leads to

$$\boldsymbol{b} = \boldsymbol{a}\boldsymbol{P}.$$

Thus, both the permutation function and the permutation matrix may be used to describe the permutation from $\boldsymbol{a}$ to $\boldsymbol{b}$. ◇

Conventions

Random Variables

Random variables are denoted by uppercase letters and realizations by the corresponding lowercase letters. The sets/alphabets are denoted by the corresponding set letters. For example, a random variable X with alphabet $\mathbb{X}$ may have a realization x.

Vectors

A (row) vector $\boldsymbol{a}$ with element indices from an index set $\mathcal{I} = \{0, 1, \ldots, L-1\}$ is written as

$$\boldsymbol{a} = [a_i]_{i \in \mathcal{I}} := [a_0, \ldots, a_{L-1}].$$

The subvector of $\boldsymbol{a}$ with elements a_i, $i \in \{k, k+1, \ldots, l-1, l\}$, is written as

$$\boldsymbol{a}_{[k,l]} := [a_k, a_{k+1}, \ldots, a_{l-1}, a_l].$$

The vector with all elements of $\boldsymbol{a}$ excluding element a_k is written as

$$\boldsymbol{a}_{\setminus k} := [a_0, \ldots, a_{k-1}, a_{k+1}, a_{L-1}].$$

The vector with all elements of $\boldsymbol{a}$, but element a_k replaced by 0 is written as

$$\boldsymbol{a}_{0 \to k} := [a_0, \ldots, a_{k-1}, 0, a_{k+1}, a_{L-1}].$$

The vector $\boldsymbol{c}$ formed by concatenation of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is written in either way

$$\boldsymbol{c} = [\boldsymbol{a}\, \boldsymbol{b}] = [\boldsymbol{a}\boldsymbol{b}] = [\boldsymbol{a}|\boldsymbol{b}].$$

As opposed to this, $[\boldsymbol{a}, \boldsymbol{b}]$ denotes the vector of the two elements $\boldsymbol{a}$ and $\boldsymbol{b}$, where each element is a vector itself.

Binary Symbols

Binary symbols are represented in $\mathbb{F}_2 := \{0, 1\}$ or $\mathbb{B} := \{-1, +1\}$. Symbols over $\mathbb{B}$ are written as a, and symbols over $\mathbb{F}_2$ are written as $\breve{a}$. For conversion, we define

$$\begin{aligned} a = +1 &\quad\Leftrightarrow\quad \breve{a} = 0, \\ a = -1 &\quad\Leftrightarrow\quad \breve{a} = 1. \end{aligned}$$

Use of Fonts

X	(scalar) random variable
$\boldsymbol{X}$	vector-valued random variable
$\mathbb{X}$	alphabet of random variable X
x	scalar or realization of random variable X
$\boldsymbol{x}$	vector or realization of random variable $\boldsymbol{X}$
$\hat{x}$	estimate of x
$\tilde{x}$	hypothesis for x
$\boldsymbol{A}$	matrix

Convexity

A function $f : \mathbb{R} \to \mathbb{R}$ is called convex-$\cap$ over $\mathbb{D} \subseteq \mathbb{R}$ if the function satisfies

$$\theta \cdot f(\alpha) + (1 - \theta) \cdot f(\beta) \leq f\big(\alpha + (1 - \theta)\beta\big)$$

for all $\alpha, \beta \in \mathbb{D}$ and $\theta \in [0, 1]$. If the inequality is reversed for all such α, β, θ, the function f is called convex-$\cup$ [Gal68].

The use of "convex-$\cap$" and "convex-$\cup$" is supported by the following quotation:

> *In the mathematical literature, a convex-$\cap$ function is usually called concave and a convex-$\cup$ function convex. That notation is avoided here since most people find the distinction very difficult to keep straight. In a recent poll among ten people who thought they knew the distinction, eight got convex and concave confused. [Gal68, p. 84, footnote]*

Lists of Variables

Variables and Sets Related to Channels

$X, x \in \mathbb{X}$	channel input
$Y, y \in \mathbb{Y}$	channel output
$A, a \in \mathbb{A}$	subchannel indicator
$\mathcal{E}, \epsilon \in \mathbb{E}$	error probability indicator

$J, j \in \mathbb{J}$	mutual information indicator
$\Lambda, \lambda \in \mathbb{L}$	reliability value indicator
ϵ	crossover probability of a BSC
δ	erasure probability of a BEC

Variables Related to Coding

u	info symbol (data symbol, information symbol[2])
x	code symbol
z	pre-decoding soft-value
v	post-decoding soft-value
w	extrinsic post-decoding soft-value
$v^\bullet$	improved (or corrected) post-decoding soft-value
$w^\bullet$	improved (or corrected) extrinsic post-decoding soft-value
l	(a-posteriori or extrinsic) LLR
$\square_{u,i}$	soft-value for info symbol U_i
$\square_{x,i}$	soft-value for code symbol X_i
a	magnitude of soft-value
$a^\bullet$	magnitude of improved (or corrected) soft-value
λ	reliability value (= magnitude of LLR)
K	info word length
N	code word length
R	code rate (K/N)
$\mathcal{C}$	code
$\mathcal{C}_{\mathsf{syxt}}$	systematically extended code of code $\mathcal{C}$
$\mathcal{R}_N$	binary repetition code of length N
$\mathcal{S}_N$	binary single parity check code of length N
$\boldsymbol{G}$	generator matrix
$\boldsymbol{A}$	transposed inverse of generator matrix ($GA^\mathsf{T} = I$)
$\boldsymbol{H}$	parity check matrix ($GH^\mathsf{T} = 0$)
$\boldsymbol{P}$	permutation matrix
$\boldsymbol{I}$	identity matrix

Variables Related to Mutual Information

$I_{\mathsf{int},i}$	intrinsic information on an info or code symbol
$I_{\mathsf{ext},i}$	extrinsic information on an info or code symbol
$I_{\mathsf{cmp},i}$	complete information on an info or code symbol

[2]The term "info symbol" is prefered in this thesis to avoid confusion with "mutual information".

I_{ext}	average symbol-wise extrinsic information
I_{cmp}	average symbol-wise complete information
I_{wcmp}	word-wise complete information per info symbol
I_{ch}	channel information (mutual information of communication channel)
I_{apri}	a-priori information (mutual information of (virtual) a-priori channel)

Other Variables

E_b	signal energy per information bit
N_0	single-sided noise power density
μ	mean value
σ	standard deviation
σ^2	variance

Appendix C

Log-Likelihood Ratios

Log-likelihood ratios (LLRs) are often used instead of probabilities when binary random variables (symbols) are considered. Some definitions, relations, properties, and operators are summarized in the sequel. Further information may be found in [HOP96]; the relation to probabilities is described in detail in [Hub02].

C.1 Definitions and Properties

Consider a binary random variable $X \in \mathbb{B}$ and a real-valued random variable Y, which depend on each other. These random variables may be interpreted as a code symbol and its noisy observation.

Definitions

Three kinds of probability distributions may be associated with X for a given realization y of Y: the a-priori probabilities $p_X(x)$, the a-posteriori probabilities $p_{X|Y}(x|y)$, and the likelihoods $p_{Y|X}(y|x)$, $x \in \mathbb{B}$. The corresponding logarithmic ratios are the a-priori LLR

$$L(X) := \ln \frac{p_X(+1)}{p_X(-1)},$$

the a-posteriori LLR

$$L(X|Y = y) := \ln \frac{p_{X|Y}(+1|y)}{p_{X|Y}(-1|y)},$$

and the LLR

$$L(Y = y|X) := \ln \frac{p_{Y|X}(y| + 1)}{p_{Y|X}(y| - 1)}.$$

Although only the last value is actually a logarithmic ratio of likelihoods, also the first and the second value are usually called log-likelihood ratios.

Conversions

The definitions of LLRs may be interpreted as conversions of probability distributions of X into real values. For a-priori LLRs and for a-posteriori LLRs, these mappings can be inverted. For the a-priori LLR $l := L(X)$, we obtain the a-priori probabilities

$$p_X(+1) = \frac{1}{1+e^{-l}}, \qquad p_X(-1) = \frac{1}{1+e^{+l}}.$$

Similarly, for the a-posteriori LLR $l := L(X|Y=y)$, we obtain the a-posteriori probabilities

$$p_{X|Y}(+1|y) = \frac{1}{1+e^{-l}}, \qquad p_{X|Y}(-1|y) = \frac{1}{1+e^{+l}}.$$

As opposed to that, the LLR $l := L(Y=y|X)$ provides not sufficient information to compute the likelihoods $p_{Y|X}(y|+1)$ and $p_{Y|X}(y|-1)$. Only the ratio between these two likelihoods can be determined.

Relations and Exponential Symmetry

Bayes's rule for probabilities,

$$p_{X,Y}(x,y) = p_X(x) \cdot p_{X|Y}(x|y),$$

transforms into the logarithmic domain as

$$\begin{aligned} L(X) + L(Y=y|X) &= \ln \frac{p_X(+1)\cdot p_{Y|X}(y|+1)}{p_X(-1)\cdot p_{Y|X}(y|-1)} \\ &= \ln \frac{p_{X,Y}(+1,y)}{p_{X,Y}(-1,y)} \\ &= \ln \frac{p_{X|Y}(+1|y)\cdot p_Y(y)}{p_{X|Y}(-1|y)\cdot p_Y(y)} \\ &= L(X|Y=y). \end{aligned}$$

This equation is sometimes called *chain rule for LLRs*. Notice that $L(Y=y|X) = L(X|Y=y)$ if $L(X)=0$, i.e., if X is uniformly distributed.

The probability distribution of the LLR $L(Y=y|X)$ shows an exponential symmetry [HLS00], also reported in [BD76, BES82, RU01a]. Let this LLR be regarded as a random variable L with realizations

$$l := L(Y=y|X).$$

Let further $p_{L|X}(l|x)$ denote its conditional probability density function. Since L is a sufficient statistic of Y, we have

$$L(L=l|U) = L(Y=y|U),$$

and thus

$$l = \ln \frac{p_{L|X}(l|+1)}{p_{L|X}(l|-1)}.$$

It follows immediately

$$p_{L|X}(l|-1) = e^{-l} \cdot p_{L|X}(l|+1). \tag{C.1}$$

Using the rules for converting LLRs to probabilities, we obtain the following symmetry:

$$p_{L|X}(l|x) = \frac{1}{1+e^{-lu}} = p_{L|X}(-l|-x); \tag{C.2}$$

the second equality follows because the expression in the middle depends only on the sign of lu. Combining (C.1) and (C.2), we obtain the *exponential symmetry* of the conditional probability density function of L:

$$p_{L|X}(-l|+1) = e^{-l} \cdot p_{L|X}(l|+1). \tag{C.3}$$

Combining this exponential symmetry with the symmetry from (C.2), we obtain

$$p_{L|X}(-l|+1) = e^{-l} \cdot p_{L|X}(-l|-1). \tag{C.4}$$

Notice that these exponential symmetry properties hold for the a-posteriori LLR if and only if the a-priori LLR is equal to zero.

C.2 Operators

Two operators are particularly useful for calculations with LLRs: the (ordinary) plus and the boxplus [BES82, HOP96]. Furthermore, the $\max^\star$ (maxstar) operator and the $\min^\star$ (minstar) operator for logarithmic probabilities are often used for writing the LogAPP decoding principle or the LogAPP algorithm [BCJR74, RHV97]. These operators are shortly revised in the sequel. Furthermore, each operator is written as the combination of an approximate operator and a corrections term. These combinations may be used for low-complexity implementations.

Plus

Consider first a binary symbol $X \in \mathbb{B}$ that is transmitted N times over a BISMC; the channel outputs are denoted by $Y_0, Y_1, \ldots, Y_{N-1}$. Then the overall LLR is given by

$$L(y_0, y_1, \ldots, y_{N-1}|X) = L(y_0|X) + L(y_1|X) + \cdots + L(y_{N-1}|X).$$

If X is uniformly distributed, we also have

$$L(X|y_0, y_1, \ldots, y_{N-1}) = L(X|y_0) + L(X|y_1) + \cdots + L(X|y_{N-1}).$$

Both equations result immediately from the definition of the LLRs. Thus, independent LLRs for the same symbol may be combined by ordinary addition.

This operation may be employed for LogAPP (and equivalently MaxLogAPP) decoding of repetition codes, cf. Section 3.4.

Boxplus

Consider now three binary symbols $X_0, X_1, X_2 \in \mathbb{B}$ that fulfill the parity check equation

$$\breve{X}_0 \oplus \breve{X}_1 \oplus \breve{X}_2 = 0.$$

(Notice the one-to-one correspondence between $X_i \in \mathbb{B}$ and $\breve{X}_i \in \mathbb{F}_2$ defined in Section 3.1.) Assume that X_1 and X_2 are transmitted over two independent BISMCs, yielding the channel outputs Y_1 and Y_2, respectively.

The likelihood for X_0 based on the two channel outputs and the two individual likelihoods for X_1 and X_2 are related as

$$\begin{aligned} &p_{Y_1,Y_2|X_0}(y_1, y_2| + 1) = \\ &\quad = c(y_1, y_2) \cdot \Big(p_{Y_1|X_1}(y_1| + 1) p_{Y_2|X_2}(y_2| + 1) \; + \; p_{Y_1|X_1}(y_1| - 1) p_{Y_2|X_2}(y_2| - 1) \Big), \end{aligned}$$

where $c(y_1, y_2)$ denotes a normalization factor that depends only on y_1 and y_2 but not on the values of X_1 and X_2. Using LLRs, this expression may be written as

$$L(y_1, y_2|X_0) = L(y_1|X_1) \boxplus L(y_2|X_2),$$

using the binary boxplus operator defined as

$$l_1 \boxplus l_2 := \ln \frac{1 + e^{l_1} e^{l_2}}{e^{l_1} + e^{l_2}} = 2 \tanh^{-1} \Big(\tanh \frac{l_1}{2} \cdot \tanh \frac{l_2}{2} \Big)$$

for $l_1, l_2 \in \mathbb{R}$. Notice that the normalization factor $c(y_1, y_2)$ cancels out when LLRs are used.

To reduce the computational complexity, the boxplus operator may be approximated:

$$l_1 \boxplus l_2 \approx l_1 \boxplus l_2 := \operatorname{sgn}(l_1) \cdot \operatorname{sgn}(l_2) \cdot \min\big\{ |l_1|, |l_2| \big\}.$$

The precise and the approximate boxplus operator are related as[1]

$$\begin{aligned} l_1 \boxplus l_2 \;&=\; \operatorname{sgn}(l_1) \cdot \operatorname{sgn}(l_2) \cdot \Bigg(\min\big\{ |l_1|, |l_2| \big\} - \underbrace{\ln \frac{1 + e^{-\left| |l_1| - |l_2| \right|}}{1 + e^{-\left| |l_1| + |l_2| \right|}}}_{\text{additive correction term}} \Bigg) \\ &=\; (l_1 \boxplus l_2) \cdot \underbrace{\left(1 - \frac{1}{\min\big\{ |l_1|, |l_2| \big\}} \ln \frac{1 + e^{-\left| |l_1| - |l_2| \right|}}{1 + e^{-\left| |l_1| + |l_2| \right|}} \right)}_{\text{multiplicative correction term}}. \end{aligned}$$

The "additive correction term" may be seen as an additive correction of the absolute value, whereas the "multiplicative correction" represents a multiplicative correction of

[1] Whereas this description via "approximation plus correction term" is well-known for the $\max^\star$ operator, no reference could be found for the boxplus operator.

the overall value. Both correction terms depend only on the absolute value of the sum and the absolute value of the difference of $|l_1|$ and $|l_2|$. The above relation allows for a low-complexity implementation of the precise operator, as the additive correction term may be stored in a (two-dimensional) look-up table.

The additive correction term ranges within the interval $[0, \ln 2]$. For a given l_1, the term is minimal if $l_2 = 0$, and it is maximal if $|l_1| = |l_2|$; the overall maximum $\ln 2$ is achieved if $|l_1| = |l_2| \to \infty$. Notice that the absolute value of the approximate result is always greater than or equal to the absolute value of the correct result.

The boxplus operator inherits associativity from the binary addition. Consider N binary symbols $X_0, X_1, \ldots, X_{N-1} \in \mathbb{B}$ fulfilling a parity check equation, and let $Y_1, Y_2, \ldots, Y_{N-1}$ denote the noisy observations of $X_1, X_2, \ldots, X_{N-1}$, respectively. Then,

$$L(y_1, y_2, \ldots, y_{N-1}|X_0) = L(y_1|X_1) \boxplus L(y_2|X_2) \boxplus \cdots \boxplus L(y_{N-1}|X_{N-1}),$$

where the expression may be evaluated recursively in arbitrary order. For example, for three LLRs $l_1, l_2, l_3 \in \mathbb{R}$, we have

$$l_1 \boxplus l_2 \boxplus l_3 = \big(l_1 \boxplus l_2\big) \boxplus l_3 = l_1 \boxplus \big(l_2 \boxplus l_3\big).$$

This holds for the approximated boxplus operator in the same way. The above expressions also hold for a-posteriori LLRs if the a-priori LLRs are equal to zero.

The correct and the approximate boxplus operator are employed for LogAPP and MaxLogAPP decoding of single parity check codes, respectively (cf. Section 3.4).

Maxstar

For implementation issues, logarithmic probabilities are sometimes preferred to (plain) probabilities. (Notice that LLRs may be interpreted as unnormalized logarithmic probabilities.) The two basic operations on probabilities are multiplication and addition. Their counterparts in the log-domain are discussed in the sequel.

Let $p, p_1, p_2 \in [0, 1]$ denote probabilities, and let $l, l_1, l_2 \in \mathbb{R}$ denote their counterparts in the log-domain:

$$l := \ln p, \qquad l_1 := \ln p_1, \qquad l_2 := \ln p_2.$$

Multiplication of probabilities transforms in convenient addition of log-probabilities, whereas addition of probabilities transforms in a more involved expression for log-probabilities, denoted by $\max^\star$:

$$\begin{aligned} p = p_1 \cdot p_2 \quad &\longrightarrow \quad l = l_1 + l_2, \\ p = p_1 + p_2 \quad &\longrightarrow \quad l = \max^\star\{l_1, l_2\} := \ln\big(e^l_1 + e^l_2\big). \end{aligned}$$

The name "maxstar" derives from the fact that this operation may be split into a maximization and a correction term:

$$\ln\big(e^l_1 + e^l_2\big) = \max\{l_1, l_2\} + \underbrace{\ln\big(1 + e^{-|l_1 - l_2|}\big)}_{\text{correction term}}.$$

The correction term depends only on the absolute value of the difference between l_1 and l_2 and ranges within the interval $[0, \ln 2]$. It is maximum if $l_1 = l_2$ and it tends to zero if the difference between l_1 and l_2 approaches infinity. Notice that the approximate result is always less than or equal to the correct result, as opposed to the boxplus operation.

For an efficient implementation, the correction term may be stored in a look-up table. When omitting the correction term, we obtain the approximation

$$\max^\star\{l_1, l_2\} \approx \max\{l_1, l_2\},$$

which is often applied in practice due to its smaller computational complexity.

The operations $\max^\star$ and max are associative, i.e., they may be evaluated recursively in arbitrary order. For example, for three values $l_1, l_2, l_3 \in \mathbb{R}_0^-$, we have

$$\max^\star\{l_1, l_2, l_3\} = \max^\star\Big\{ \max^\star\{l_1, l_2\}, l_3\Big\} = \max^\star\Big\{l_1, \max^\star\{l_2, l_3\}\Big\}.$$

This holds for max in the same way, as can easily be seen.

The $\max^\star$ operator and its approximation by max are employed in the LogAPP and the MaxLogAPP decoding principle, respectively (cf. Section 3.4).

Minstar

Similarly to the $\max^\star$ operator, the $\min^\star$ operator is defined as

$$\min^\star\{l_1, l_2\} := -\ln\big(e^{-l_1} + e^{-l_2}\big)$$

for $l_1, l_2 \in \mathbb{R}$, [EPG94]. The two operators are related as

$$\min^\star\{l_1, l_2\} = -\max^\star\{-l_1, -l_2\}.$$

Therefore, the properties of the $\min^\star$ operator can easily be determined from the corresponding properties of the $\max^\star$ operator.

Appendix D

Information Theory

Mutual information, entropy, and Kullback-Leibler distance are basic notions in information theory. These are shortly summarized in this chapter. Further details may be found in [Gal68, CT91, Joh92, Mac03].

D.1 Entropy

Consider a discrete random variable X with probability mass function $p_X(x)$. The entropy of X is defined as the expectation

$$H(X) := \mathrm{E}\big\{-\operatorname{ld} p_X(X)\big\},$$

with $\operatorname{ld} z := \log_2 z$. Notice that $p_X(x)$ is a function of x. The uppercase X in the argument indicates that the expectation is evaluated with respect to X. For a continuous random variable X' with probability density function $p_{X'}(x')$, the expected value

$$H_{\text{diff}}(X') := \mathrm{E}\big\{-\operatorname{ld} p_{X'}(X')\big\}$$

is called the differential entropy[1].

The *binary entropy function* is defined as

$$h(p) := -p \operatorname{ld} p - (1-p)\operatorname{ld}(1-p) \quad \in [0,1],$$

$p \in [0,1]$. It is invertible for $p \in [0, \frac{1}{2}]$, and this inverse function is denoted by $h^{-1}(\eta) \in [0, \frac{1}{2}]$, $\eta \in [0,1]$. The entropy of a binary random variable X with probability distribution $[p, 1-p]$ is thus given by

$$H(X) = h(p) = h(1-p).$$

Conversely, for $\eta := H(X)$, we obtain

$$h^{-1}(\eta) = \min\{p, 1-p\}.$$

[1] Although it should be called integral entropy [Hub04].

Consider an additional random variable Y. Let $p_{X|Y}(x|y)$ denote the conditional probability mass function if X is discrete, and the conditional probability density function if X is continuous. The conditional entropy of X given Y is defined as the expectation

$$H(X|Y) := \mathrm{E}\big\{-\operatorname{ld} p_{X|Y}(X|Y)\big\}.$$

As above, the uppercase letters X and Y in the argument indicate that the expectation is evaluated with respect to both X and Y.

D.2 Mutual Information

In this section, we address first the mutual information of single channels. The channels formed by serial or parallel concatenation of multiple binary symmetric channels (BSCs) are of particular interest in Chapter 6. The mutual information of such channels are addressed in the second and the third part of this section.

D.2.1 Single Channels

Consider two random variables X and Y. Let $p_{X,Y}(x,y)$ denote the joint distribution, i.e., the joint probability mass function or the joint probability density function, depending on whether X and Y are discrete or continuous. The marginal distributions are given by

$$p_X(x) = \mathrm{E}\big\{p_{X,Y}(x,Y)\big\}, \qquad p_Y(y) = \mathrm{E}\big\{p_{X,Y}(X,y)\big\}.$$

The conditional distributions may be computed using Bayes' rule:

$$p_{X|Y}(X|Y) = \frac{p_{X,Y}(x,y)}{p_Y(y)}, \qquad p_{Y|X}(y|x) = \frac{p_{X,Y}(x,y)}{p_X(x)}.$$

The mutual information between X and Y is defined as

$$I(X;Y) := \mathrm{E}\left\{\operatorname{ld}\frac{p_{X,Y}(X,Y)}{p_X(X)p_Y(Y)}\right\}.$$

As can be seen, the mutual information is symmetric in X and Y. It can also be computed using either expression:

$$I(X;Y) = \mathrm{E}\left\{\operatorname{ld}\frac{p_{X|Y}(X|Y)}{p_X(X)}\right\} = \mathrm{E}\left\{\operatorname{ld}\frac{p_{Y|X}(Y|X)}{p_Y(Y)}\right\}.$$

Mutual information and entropy are related as

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

The capacity C of a channel $X \to Y$ is defined as the maximal mutual information subject to the input distribution, i.e.,

$$C := \max_{p_X} I(X;Y).$$

The capacity of symmetric channels is achieved for equiprobable input symbols.

Consider an additional random variable A. The conditional mutual information between X and Y given A is defined as

$$I(X;Y|A) := \mathrm{E}\left\{ \mathrm{ld}\, \frac{p_{X|Y,A}(X|Y,A)}{p_{X|A}(X|A)} \right\},$$

where $p_{X|Y,A}(x|y,a)$ and $p_{X|A}(x|a)$ denote probability mass functions for discrete X and probability density functions for continuous X. Using the definitions of unconditional and conditional mutual information, we obtain the chain rule of mutual information

$$I(X,A;Y) = I(A;Y) + I(X;Y|A).$$

Notice that $I(X,A;Y)$ denotes the mutual information between the vector-valued random variable $[X,A]$ and the random variable Y. In fact, this is a short-hand notation of $I([X,A];Y)$.

D.2.2 Serially Concatenated BSCs

Consider N binary symmetric channels (BSCs) $X_i \to Y_i$, $X_i, Y_i \in \mathbb{B}$, $i = 0, 1, \ldots, N-1$, which are serially concatenated such that $Y_i = X_{i+1}$ for $i = 0, 1, \ldots, N-2$:

$$X_0 \to Y_0 = X_1 \to Y_1 = X_2 \to \cdots \to Y_{N-2} = X_{N-1} \to Y_{N-1}.$$

The input of the first channel is assumed to be uniformly distributed. The mutual information of each individual channel is denoted by $I_i := I(X_i;Y_i)$, $i = 0, 1, \ldots, N-1$. The end-to-end mutual information between the input of the first and the output of the last channel is denoted by $I := I(X_0;Y_{N-1})$.

It is now shown that the end-to-end mutual information is given by the binary information function for serial concatenation according to Definition 6.1, i.e.,

$$I = f_N^{\mathsf{ser}}(I_0, I_1, \ldots, I_{N-1}). \tag{D.1}$$

As the serially concatenated channel $X_0 \to Y_{N-1}$ is symmetric, the mutual information I is equal to the channel capacity.

We start with the case $N = 2$. It can easily be seen that the serially concatenated channel $X_0 \to Y_1$ is also a BSC. Let ϵ_{01} denote its crossover probability. We have an error on this channel if an error occurs either on the first or on the second channel. With $\epsilon_0 = h^{-1}(1-I_0)$ and $\epsilon_1 = h^{-1}(1-I_1)$ denoting the crossover probabilities of the two individual channels, we can compute the crossover probability of the serially concatenated channel as[2]

$$\epsilon_{01} = (1-\epsilon_0)\epsilon_1 + \epsilon_0(1-\epsilon_1).$$

Thus, its mutual information is given by

$$\begin{aligned} I(X_0;Y_1) &= 1 - h(\epsilon_{01}) \\ &= 1 - h\Big((1-\epsilon_0)\epsilon_1 + \epsilon_0(1-\epsilon_1)\Big), \end{aligned}$$

and we have the proof of (D.1) for $N = 2$.

The general case follows by induction.

[2]This operation is called convolution of the two probabilities ϵ_0 and ϵ_1 in [WZ73, CSS89].

D.2.3 Parallel Concatenated BSCs

Consider N binary symmetric channels (BSCs) $X \to Y_i$, $X, Y_i \in \mathbb{B}$, $i = 0, 1, \ldots, N-1$, that have the same input X. Following the accepted practice for parallel concatenated codes, see [BM96b], we call these channels parallel concatenated. The input X is assumed to be uniformly distributed. The mutual information of each channel is denoted by $I_i := I(X;Y_i)$, $i = 0, 1, \ldots, N-1$. The vector of channel outputs is written as $\boldsymbol{Y} := [Y_0, Y_1, \ldots, Y_{N-1}]$. The overall mutual information between the input and the vector of channel outputs is denoted by $I := I(X;\boldsymbol{Y})$.

It is now shown that the overall mutual information is given by the binary information function for parallel concatenation according to Definition 6.2, i.e.,

$$I = f_N^{\mathsf{par}}(I_0, I_1, \ldots, I_{N-1}). \tag{D.2}$$

As the parallel concatenated channel $X \to \boldsymbol{Y}$ is symmetric, the mutual information I is equal to the channel capacity.

To start with, we write the overall mutual information as

$$I = I(X;\boldsymbol{Y}) = H(\boldsymbol{Y}) - H(\boldsymbol{Y}|X). \tag{D.3}$$

The first term can be computed using the joint probabilities of the channel outputs,

$$\begin{aligned} H(\boldsymbol{Y}) &= \mathrm{E}\big\{-\operatorname{ld} p_{\boldsymbol{Y}}(\boldsymbol{y})\big\} \\ &= -\sum_{\boldsymbol{y}\in\mathbb{B}^n} p_{\boldsymbol{Y}}(\boldsymbol{y}) \cdot \operatorname{ld} p_{\boldsymbol{Y}}(\boldsymbol{y}) \end{aligned}$$

with

$$\begin{aligned} p_{\boldsymbol{Y}}(\boldsymbol{y}) &= \sum_{x\in\mathbb{B}} p_{X,\boldsymbol{Y}}(x, \boldsymbol{y}) \\ &= \sum_{x\in\mathbb{B}} p_X(x) \cdot p_{\boldsymbol{Y}|X}(\boldsymbol{y}|x) \\ &= \sum_{x\in\mathbb{B}} p_X(x) \cdot \prod_{i=0}^{N-1} p_{Y_i|X}(y_i|x). \end{aligned}$$

In the last line, we have used the conditional independence of the channel outputs for a given channel input. Notice that $p_X(x) = \frac{1}{2}$ due to the uniform input distribution. The transition probabilities of each channel can be expressed using its mutual information:

$$p_{Y_i|X}(y_i|x) \in \{\epsilon_i, 1-\epsilon_i\}$$

with

$$\epsilon_i := h^{-1}(1 - I_i),$$

$i = 0, 1, \ldots, N-1$. Thus, the joint probability of a vector of channel outputs y can be obtained according to

$$p_{\boldsymbol{Y}}(\boldsymbol{y}) = \frac{1}{2}\Big(\prod_{i=0}^{N-1} \varphi_i(y_i) + \prod_{i=0}^{N-1} \big(1 - \varphi_i(y_i)\big)\Big),$$

with

$$\varphi_i(y_i) := \begin{cases} \epsilon_i & \text{for } y_i = +1, \\ 1 - \epsilon_i & \text{for } y_i = -1. \end{cases}$$

The second term in (D.3) can be written as

$$H(\boldsymbol{Y}|X) = \sum_{i=0}^{N-1} H(Y_i|X) = \sum_{i=0}^{N-1} (1 - I_i),$$

where the conditional independence of the channel outputs for a given channel input has again been used.

By substituting the above equations into (D.3), we obtain the proof of (D.2).

D.3 Kullback-Leibler Distance

The Kullback-Leibler distance[3] may be used to measure the difference between two probability distributions for the same random variable. It is also called relative entropy or cross entropy[4]. Notice that it is not a true distance in the mathematical sense, because it is not symmetric.

The Kullback-Leibler distance has successfully been applied in probabilistic optimization problems, e.g., in the well-known expectation-maximization algorithm [Moo96,FH94]. Furthermore, the LogAPP decoding principle can be shown to minimize the Kullback-Leibler distance between the probability distribution before decoding and that after decoding (after taking into account the code constraints). This was observed for word-decoding in [Moh93, MG98] and for symbol-by-symbol decoding in [Sor02].

Consider a discrete random variable X with alphabet $\mathbb{X}$ and two probability distributions $p_X(x)$ and $q_X(x)$. The Kullback-Leibler distance between p_X and q_X is defined as

$$D(p_X \| q_X) := \sum_{x \in \mathbb{X}} p_X(x) \cdot \mathrm{ld} \frac{p_X(x)}{q_X(x)}.$$

If X is continuous, the sum has to be replaced by an integral. The Kullback-Leibler distance may be interpreted as an expected value. Then, the distribution p_X is the actual distribution, because it is used to evaluate the expectation. The distribution q_X may be seen as a model distribution that is compared to the actual distribution.

The Kullback-Leibler distance is nonnegative, and it is equal to zero precisely if the two distributions are identical:

$$\begin{aligned} D(p_X \| q_X) &\geq 0 \quad \text{for all } p_X, q_X, \\ D(p_X \| q_X) &= 0 \quad \text{precisely if } p_X \equiv q_X. \end{aligned}$$

These properties are very important, and they are often exploited for solving optimization problems, e.g., in the expectation-maximization algorithm [Moo96].

[3] The Kullback-Leibler distance is *not* a mathematical distance.

[4] The name "cross entropy" is rather misleading, as in fact not an entropy but a difference of entropies is addressed.

The Kullback-Leibler distance is used in Chapter 4 for measuring the quality of soft-values.

Appendix E

Convexity Lemma

The proof of Lemma 6.1 is based on the convexity of function $g(x)$. The proof of convexity is given in the following lemma. Alternatively, Mrs. Gerber's Lemma [WZ73] may be used [SS03].

Lemma E.1
The function

$$g(x) = h\big([1-2a]h^{-1}(x) + a\big),$$

$x \in [0,1]$, $a \in [0, \frac{1}{2}]$, *is convex-*$\cup$.

Proof. The function is convex-$\cup$ if the second derivative of $g(x)$ with respect to x is non-negative, i.e., if

$$\frac{d^2 g(x)}{dx^2} \geq 0 \tag{E.1}$$

for $x \in [0,1]$ and $a \in [0, \frac{1}{2}]$.

First, this function is parameterized. Let $x = h(t)$, $t \in [0, \frac{1}{2}]$, and let $y = g(x)$. Then we have

$$y = g\big((h(t)\big) = h([1-2a]t + a). \tag{E.2}$$

The derivatives of x with respect to t are given as

$$\frac{dx}{dt} = h'(t) = \operatorname{ld} \frac{1-t}{t} \geq 0, \tag{E.3}$$

$$\frac{d^2 x}{dt^2} = h''(t) = -\frac{\operatorname{ld} e}{t(1-t)} \leq 0, \tag{E.4}$$

$$\frac{d^3 x}{dt^3} = h'''(t) = \frac{1-2t}{t^2(1-t)^2} \cdot \operatorname{ld} e. \tag{E.5}$$

For the first and the second derivative, also the co-domains are stated. Similarly, the derivatives of y with respect to t are given as

$$\frac{dy}{dt} = [1-2a] \cdot h'([1-2a]t + a) \geq 0, \tag{E.6}$$

$$\frac{d^2 x}{dt^2} = [1-2a]^2 \cdot h''([1-2a]t + a) \leq 0. \tag{E.7}$$

Again, also the co-domains are stated.

Consider now the relations between the derivatives with respect to x and the derivatives with respect to t. The first derivative can be written as

$$\frac{dy}{dt} = \frac{dy}{dx} \cdot \frac{dx}{dt},$$

and thus it follows that

$$\frac{dy}{dx} = \frac{dy/dt}{dx/dt}. \tag{E.8}$$

The second derivative can be written as

$$\begin{aligned} \frac{d^2y}{dt^2} &= \frac{d}{dt}\Big(\frac{dy}{dx}\Big) \cdot \frac{dx}{dt} + \frac{dy}{dx} \cdot \frac{d}{dt}\Big(\frac{dx}{dt}\Big) \\ &= \frac{d^2y}{dx^2} \cdot \frac{dx}{dt} \cdot \frac{dx}{dt} + \frac{dy}{dx} \cdot \frac{d^2x}{dt^2}, \end{aligned}$$

and thus it follows that

$$\frac{d^2y}{dx^2} = \frac{\dfrac{d^2y}{dt^2} - \dfrac{dy}{dx} \cdot \dfrac{d^2x}{dt^2}}{\Big(\dfrac{dx}{dt}\Big)^2}.$$

Since $(dx/dt)^2 \geq 0$, we have

$$\begin{aligned} \frac{d^2y}{dx^2} \geq 0 \quad &\Leftrightarrow \quad \frac{d^2y}{dt^2} - \frac{dy}{dx} \cdot \frac{d^2x}{dt^2} \geq 0 \\ &\Leftrightarrow \quad \frac{d^2y}{dt^2} \geq \frac{dy}{dx} \cdot \frac{d^2x}{dt^2} = \frac{dy/dt}{dx/dt} \cdot \frac{d^2x}{dt^2} \\ &\Leftrightarrow \quad \frac{d^2y/dt^2}{dy/dt} \geq \frac{d^2x/dt^2}{dx/dt}, \end{aligned} \tag{E.9}$$

where (E.8) was applied in the second line, and $dy/dt \geq 0$ was used in the third line.

Using the expressions for the derivatives, (E.3), (E.4), (E.6), (E.7), in (E.9) yields

$$\frac{[1-2a]^2 \cdot h''([1-2a]t + a)}{[1-2a] \cdot h'([1-2a]t + a)} \geq \frac{h''(t)}{h'(t)}. \tag{E.10}$$

Let

$$b(t) := \frac{h''(t)}{h'(t)} \tag{E.11}$$

and

$$c(a) := [1-2a] \cdot b([1-2a]t + a).$$

Then (E.10) can be written as

$$c(a) \geq c(0)$$

for $a \in [0, \frac{1}{2}]$. This relation holds if the first derivative of $c(a)$ with respect to a is non-negative.

Thus, we can give the following sufficient condition for (E.1):

$$\frac{dc(a)}{da} \geq 0 \tag{E.12}$$

for $a \in [0, \frac{1}{2}]$ and $t \in [0, \frac{1}{2}]$. This derivative can be computed as

$$\begin{aligned}
\frac{dc(a)}{da} &= \frac{d}{da}\Big([1-2a]\cdot b([1-2a]t+a)\Big) \\
&= -2\cdot b([1-2a]t+a) + [1-2a]\cdot b'([1-2a]t+a)\cdot(1-2t) \\
&= -2\cdot b([1-2a]t+a) + \Big(1-2([1-2a]t+a)\Big)\cdot b'([1-2a]t+a).
\end{aligned}$$

After substituting $s := [1-2a]t + a$, (E.12) holds if and only if

$$-2\cdot b(s) + (1-2s)\cdot b'(s) \geq 0 \tag{E.13}$$

for $s \in [t, \frac{1}{2}]$ and $t \in [0, \frac{1}{2}]$, and thus for $s \in [0, \frac{1}{2}]$. In (E.13), we firstly apply (E.11) and

$$b'(t) = \frac{db(t)}{dt} = \frac{h'''(t)\cdot h'(t) - \big(h''(t)\big)^2}{\big(h'(t)\big)^2}$$

and then we apply the expressions for $h(x)$ and its derivatives, (E.3), (E.4), (E.5). This gives the following equivalent relations:

$$\begin{aligned}
(1-2s)\cdot b'(s) &\geq 2\cdot b(s) \\
(1-2s)\cdot \frac{h'''(s)\cdot h'(t) - \big(h''(s)\big)^2}{\big(h'(s)\big)^2} &\geq 2\cdot\frac{h''(s)}{h'(s)} \\
(1-2s)\cdot\Big[h'''(s)\cdot h'(t) - \big(h''(s)\big)^2\Big] &\geq 2\cdot h''(s)\cdot h'(s) \\
(1-2s)\cdot\Big[\frac{1-2s}{s^2(1-s)^2}\cdot \operatorname{ld}\frac{1-s}{s} - \frac{\operatorname{ld} e}{s^2(1-s)^2}\Big] &\geq 2\cdot\frac{-1}{s(1-s)}\cdot\operatorname{ld}\frac{1-s}{s} \\
(1-2s)^2\cdot\operatorname{ld}\frac{1-s}{s} - (1-2s)\cdot\operatorname{ld} e &\geq -2s(1-s)\cdot\operatorname{ld}\frac{1-s}{s} \\
\Big[(1-2s)^2 + 2s(1-s)\Big]\operatorname{ld}\frac{1-s}{s} &\geq (1-2s)\cdot\operatorname{ld} e \\
\ln\frac{1-s}{s} &\geq \frac{1-2s}{1-2s+2s^2}.
\end{aligned} \tag{E.14}$$

We substitute now $u := (1-s)/s$. From $s \in [0, \frac{1}{2}]$, it follows that $u \in [1, \infty)$. Using $s = 1/(1+u)$ in (E.14), the left hand side results as $\ln u$, and the right hand side results as

$$\frac{1-2\frac{1}{1+u}}{1-2\frac{1}{1+u}+2(\frac{1}{1+u})^2} = \frac{(1+u)^2-2(1+u)}{(1+u)^2-2(1+u)+2} = \frac{u^2-1}{u^2+1}.$$

Thus, (E.1) holds if

$$\ln u \geq \frac{u^2-1}{u^2+1}$$

for $u \in [1, \infty)$. Since equality holds for $u = 1$, it is sufficient to show that

$$\frac{d}{du} \ln u \geq \frac{d}{du}\Big(\frac{u^2 - 1}{u^2 + 1}\Big). \tag{E.15}$$

The left hand side results as $1/u$, and the right hand side results as

$$\frac{d}{du}\Big(\frac{u^2 - 1}{u^2 + 1}\Big) = \frac{2u(u^2 + 1) - (u^2 - 1)2u}{(u^2 + 1)^2} = \frac{4u}{(u^2 + 1)^2}.$$

Therefore (E.15) can equivalently be written as

$$\begin{array}{rrcl} & \frac{1}{u} & \geq & \frac{4u}{(u^2+1)^2} \\ \Leftrightarrow & (u^2+1)^2 & \geq & 4u^2 \\ \Leftrightarrow & (u^2-1)^2 & \geq & 0. \end{array}$$

To sum up, a sufficient condition for (E.1) is

$$(u^2 - 1)^2 \geq 0$$

for $u \in [1, \infty)$. Since this is the case, we have the proof. QED

Bibliography

[3GP00] "3rd Generation Partnership Project, Technical Specification Group Radio Access Network, Multiplexing and Channel Coding, 3GPP TS 25.212, V3.4.0," Sept. 2000.

[AK02] A. Abedi and A. K. Khandani, "Some properties of bit decoding algorithms over a generalized channel model," in *Proc. Conf. Inform. Sciences and Systems (CISS)*, Princeton University, Princeton, NJ, USA, Mar. 2002.

[AKtB02] A. Ashikhmin, G. Kramer, and S. ten Brink, "Code rate and the area under extrinsic information transfer curves," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Lausanne, Switzerland, June 2002, p. 115.

[AKtB03] ——, "Extrinsic information transfer functions, information functions, support weights, and duality," in *Proc. Int. Symp. on Turbo Codes & Rel. Topics*, Brest, France, Sept. 2003, pp. 223–226.

[AKtB04] ——, "Extrinsic information transfer functions: model and erasure channel properties," *IEEE Trans. Inform. Theory*, 2004.

[And73] I. N. Andersen, "Sample-whitened matched filters," *IEEE Trans. Inform. Theory*, vol. IT-19, no. 5, pp. 653–660, Sept. 1973.

[Bat87] G. Battail, "Pondération des symboles décodés par l'algorithme de Viterbi," *Ann. Télécommun.*, vol. 42, pp. 31–38, Jan. 1987.

[BCJR74] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, pp. 284–287, Mar. 1974.

[BD76] G. Battail and M. C. Decouvelaere, "Décodage par répliques," *Ann. Télécommun.*, vol. 31, no. 11-12, pp. 387–404, 1976.

[BDG79] G. Battail, M. C. Decouvelaere, and P. Godlewski, "Replication decoding," *IEEE Trans. Inform. Theory*, vol. 25, no. 3, pp. 332–345, May 1979.

[BDMP98a] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Analysis, design, and iterative decoding of double serially concatenated codes with interleavers," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 231–244, Feb. 1998.

[BDMP98b] ——, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 909–926, May 1998.

[BES82] G. Battail and A. H. M. El-Sherbini, "Coding for radio channels," *Ann. Télécommun.*, vol. 37, pp. 75–96, 1982.

[BG96] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1064–1070, May 1993.

[BM96a] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 591–600, May 1996.

[BM96b] ——, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 409–428, Mar. 1996.

[BMD03] S. Benedetto, G. Montorsi, and D. Divsalar, "Concatenated convolutional codes with interleavers," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 102 – 109, Aug. 2003.

[Bos98] M. Bossert, *Kanalcodierung*. B. G. Teubner, 1998.

[Brä04] F. Brännström, "Convergence analysis and design of multiple concatenated codes," Ph.D. dissertation, Chalmers University of Technology, Göteborg, Sweden, 2004.

[Bre02] M. Breiling, "Analysis and design of turbo code interleavers," Ph.D. dissertation, University Erlangen-Nürnberg, Germany, 2002.

[CF02] J. Chen and M. P. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 406–414, Mar. 2002.

[CRU01] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[CSS89] N. Chayat and S. Shamai (Shitz), "Expansion of an entropy property for binary input memoryless symmetric channels," *IEEE Trans. Inform. Theory*, vol. 35, no. 5, pp. 1077–1079, Sept. 1989.

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

[DJM98] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for 'turbo-like' codes," in *Proc. Allerton Conf. on Communications, Control, and Computing*, Allerton House, Monticello, Illinois, USA, Sept. 1998, pp. 201–210.

[DP97] D. Divsalar and F. Pollara, "Hybrid concatenated codes and iterative decoding," TDA Progress Report, Tech. Rep. 42-130, 1997.

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

[Eli54] P. Elias, "Error-free coding," *IRE Trans. Inform. Theory*, vol. 4, no. 4, pp. 29 –37, Sept. 1954.

[EPG94] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structures for ISI channels," *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 1661–1671, Feb./Mar./Apr. 1994.

[FB03] M. Ferrari and S. Bellini, "Existence and uniqueness of the solution for turbo decoding of parallel concatenated single parity check codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 3, pp. 722–726, Mar. 2003.

[FBLH98] M. P. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and Max-Log-MAP decodings," *Electron. Letters*, vol. 2, no. 5, pp. 137–139, May 1998.

[FH94] J. Fessler and A. Hero, "Space-alternating generalized expectation-maximization algorithm," *IEEE Trans. Signal Processing*, vol. 42, no. 10, pp. 2664–2677, Oct. 1994.

[For66] G. D. Forney, Jr., "Concatenated codes," Ph.D. dissertation, Cambridge, MA, USA, 1966.

[For73] ——, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[For01] ——, "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.

[For03] ——, "Codes on graphs: Constraint complexity of cycle-free realizations of linear codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 7, pp. 1597–1610, July 2003.

[Fri94] B. Friedrichs, *Kanalcodierung.* Springer-Verlag, 1994.

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

[Gal63] R. G. Gallager, "Low density parity check codes," Ph.D. dissertation, Cambridge, MA, USA, 1963.

[Gal68] ——, *Information Theory and Reliable Communication.* John Wiley & Sons, 1968.

[Hám98] J. Hámorský, "Parallel und seriell verkettete Codes für iterative Decodierung und Entzerrung," Ph.D. dissertation, University Erlangen-Nürnberg, Germany, 1998.

[Has87] T. Hashimoto, "A list-type reduced-constraint generalization of the Viterbi algorithm," *IEEE Trans. Inform. Theory*, vol. IT-33, no. 6, pp. 866–876, Nov. 1987.

[HH89a] J. Hagenauer and P. Hoeher, "Concatenated Viterbi-decoding," in *Proc. 4th Joint Swedish-Soviet Intern. Workshop on Inform. Theory*, Gotland, Sweden, Aug./Sept. 1989, pp. 29–33.

[HH89b] ——, "A Viterbi alorithm with soft-decision outputs and its applications," in *Proc. IEEE Globecom Conf.*, Texas, USA, Nov. 1989, pp. 1680–1686.

[HH02] S. Huettinger and J. Huber, "Extrinsic and intrinsic information in systematic coding," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Lausanne, Switzerland, June 2002, p. 116.

[HH03] ——, "Performance estimation for concatenated coding schemes," in *Proc. IEEE Inform. Theory Workshop*, Paris, France, Mar./Apr. 2003, pp. 123–126.

[HHFJ02] S. Huettinger, J. Huber, R. Fischer, and R. Johannesson, "Soft-output-decoding: Some aspects from information theory," in *Proc. Int. ITG Conf. on Source and Channel Coding*, Berlin, Germany, Jan. 2002, pp. 81–90.

[HHJF01] S. Huettinger, J. Huber, R. Johannesson, and R. Fischer, "Information processing in soft-output decoding," in *Proc. Allerton Conf. on Communications, Control, and Computing*, Monticello, Illinois, USA, Oct. 2001.

[HLS00] P. Hoeher, I. Land, and U. Sorger, "Log-likelihood values and Monte Carlo simulation - some fundamental results," in *Proc. Int. Symp. on Turbo Codes & Rel. Topics*, Brest, France, Sept. 2000, pp. 43–46.

[Hoe95] P. Hoeher, "Optimal subblock-by-subblock detection," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 714–717, Feb./Mar./Apr. 1995.

[Hoe97] ——, "New iterative ("turbo") decoding algorithms," in *Proc. Int. Symp. on Turbo Codes & Rel. Topics*, Brest, France, Sept. 1997, pp. 63–70.

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

[HR70] M. Hellman and J. Raviv, "Probability of error, equivocation, and the Chernoff bound," *IEEE Trans. Inform. Theory*, vol. 16, no. 4, pp. 368–372, July 1970.

[HR76] C. R. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inform. Theory*, vol. 22, no. 5, pp. 514–517, Sept. 1976.

[HR90] J. B. Huber and A. Rueppel, "Zuverlässigkeitsschätzung für die Ausgangssymbole von Trellis-Decoder," *Archiv für Elektronik und Übertragungstechnik (AEÜ)*, vol. 44, pp. 8–21, Jan. 1990.

[HtBH01] S. Huettinger, S. ten Brink, and J. Huber, "Turbo-code representation of RA-codes and DRS-codes for reduced decoding complexity," in *Proc. Conf. Inform. Sciences and Systems (CISS)*, The Johns Hopkins University, Baltimore, MD, USA, Mar. 2001, pp. 118–123.

[Hub02] J. Huber, "Grundlagen der Wahrscheinlichkeitsrechnung für iterative Decodierverfahren," *e&i: Elektrotechnik und Informationstechnik*, vol. 119, no. 11, pp. 386–394, Nov. 2002.

[Hub04] ——, personal communication, July 2004.

[Hue04] S. Huettinger, "Analysis and design of power-efficient coding schemes," Ph.D. dissertation, University Erlangen-Nürnberg, Germany, 2004.

[HW99] C. Heegard and S. B. Wicker, *Turbo Coding*. Kluwer Academic Publishers, 1999.

[IEE01] *IEEE Trans. Inform. Theory*, vol. 47, Feb. 2001.

[Joh92] R. Johannesson, *Informationstheorie - Grundlage der (Tele-) Kommunikation*. Addison-Wesley, 1992.

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

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

[LC83] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, 1983.

[LC02] I. Land and S. Chaoui, "Comparison of convolutional coupled codes and partially systematic turbo codes for medium code lengths," in *Proc. Int. ITG Conf. on Source and Channel Coding*, Berlin, Germany, Jan. 2002, pp. 99–104.

[LH00a] I. Land and P. Hoeher, "Partially systematic rate 1/2 turbo codes," in *Proc. Int. Symp. on Turbo Codes & Rel. Topics*, Brest, France, Sept. 2000, pp. 287–290.

[LH00b] I. Land and P. A. Hoeher, "Improving the reliability by iterative decoding," in *Proc. Winter School on Coding and Information Theory*, Reisensburg Castle, Günzburg, Germany, Feb. 2000.

[LH01] I. Land and P. Hoeher, "Using the mean reliability as a design and stopping criterion for turbo codes," in *Proc. IEEE Inform. Theory Workshop*, Cairns, Australia, Sept. 2001, pp. 27–29.

[LH03] I. Land and P. A. Hoeher, "New results on Monte Carlo bit error simulation based on the a posteriori log-likelihood ratio," in *Proc. Int. Symp. on Turbo Codes & Rel. Topics*, Brest, France, Sept. 2003, pp. 531–534.

[LHG04] I. Land, P. A. Hoeher, and S. Gligorević, "Computation of symbol-wise mutual information in transmission systems with LogAPP decoders and application to EXIT charts," in *Proc. Int. ITG Conf. on Source and Channel Coding*, Erlangen, Germany, Jan. 2004, pp. 195–202.

[LHH92] J. H. Lodge, P. Hoeher, and J. Hagenauer, "The decoding of multidimensional codes using separable MAP "filters"," in *Proc. Queens University Biennial Symp. on Communications*, Queens University, Kingston, Ontario, Canada, May 1992, pp. 343–346.

[LHH04a] I. Land, P. A. Hoeher, and J. Huber, "Analytical derivation of EXIT charts for simple block codes and for LDPC codes using information combining," in *Proc. European Signal Processing Conference (EUSIPCO)*, Vienna, Austria, Sept. 2004.

[LHH04b] ——, "Bounds on information combining for parity-check equations," in *Proc. Int. Zurich Seminar on Communications (IZS)*, Zurich, Switzerland, Feb. 2004, pp. 68–71.

[LHHH03] I. Land, S. Huettinger, P. A. Hoeher, and J. Huber, "Bounds on information combining," in *Proc. Int. Symp. on Turbo Codes & Rel. Topics*, Brest, France, Sept. 2003, pp. 39–42.

[LHHH05a] ——, "Bounds on information combining," *IEEE Trans. Inform. Theory*, vol. 51, no. 2, pp. 612–619, Feb. 2005.

[LHHH05b] ——, "Bounds on mutual information for simple codes using information combining," *Ann. Télécommun.*, vol. 60, no. 1/2, pp. 184–214, Jan./Feb. 2005.

[LHS00] I. Land, P. Hoeher, and U. Sorger, "On the interpretation of the APP algorithm as an LLR filter," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Sorrento, Italy, June 2000, p. 415.

[LKFF98] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes*. Kluwer Academic Publishers, 1998.

[Loe94] H. A. Loeliger, "A posteriori probabilities and performance evaluation of trellis codes," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Trondheim, Norway, June 1994, p. 335.

[Loe04] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[LS04] G. Lechner and J. Sayir, "Improved sum-min decoding of LDPC codes," in *Proc. IEEE Int. Symp. Inform. Theory and Its Applications (ISITA)*, Parma, Italy, Oct. 2004.

[LSH04] I. Land, J. Sayir, and P. A. Hoeher, "Bounds on information combining for the accumulator of repeat-accumulate codes without Gaussian assumption," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Chicago, USA, June/July 2004, p. 443.

[LV95] A. Lafourcade and A. Vardy, "Lower bounds on trellis complexity of block codes," *IEEE Trans. Inform. Theory*, vol. 41, no. 6, pp. 1938–1954, Nov. 1995.

[LVS95] Y. Li, B. Vucetic, and Y. Sato, "Optimum soft-output detection for channels with intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 41, no. 3, pp. 704–713, May 1995.

[LYHH93] J. Lodge, R. Young, P. Hoeher, and J. Hagenauer, "Separable MAP "filters" for the decoding of product and concatenated codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Geneva, Switzerland, May 1993, pp. 1740–1745.

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

[Mac03] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[McE96] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 4, pp. 1072–1092, July 1996.

[MDP00] A. Matache, S. Dolinar, and F. Pollara, "Stopping rules for turbo decoders," TMO Progress Report, Tech. Rep. 42-142, Aug. 2000.

[MG98] M. Moher and T. A. Gulliver, "Cross-entropy and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, no. 7, pp. 3097–3104, Nov. 1998.

[MN97] D. J. MacKay and R. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Letters*, vol. 33, no. 6, pp. 457–458, Mar. 1997.

[Moh93] M. Moher, "Decoding via cross-entropy minimization," in *Proc. IEEE Globecom Conf.*, Houston, Texas, USA, Dec. 1993, pp. 809–813.

[Moo96] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Mag.*, vol. 13, no. 6, pp. 47–60, Nov. 1996.

[Moq02] P. Moqvist, "Serially concatenated continuous phase modulation," Ph.D. dissertation, Chalmers University of Technology, Göteborg, Sweden, 2002.

[MS88] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1988.

[NS95] C. Nill and C.-E. W. Sundberg, "List and soft symbol output Viterbi algorithms: Extensions and comparisons," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 277–287, Feb./Mar./Apr. 1995.

[NS97] K. R. Narayanan and G. L. Stüber, "A novel ARQ technique using the turbo coding principle," *IEEE Commun. Lett.*, vol. 1, no. 2, pp. 49–51, Mar. 1997.

[PAT04] E. Papagiannis, M. A. Ambroze, and M. Tomlinson, "Approaching the ML performance with iterative decoding," in *Proc. Int. Zurich Seminar on Communications (IZS)*, Zurich, Switzerland, Feb. 2004, pp. 220–223.

[PM03] J. Park and J. Moon, "Alternative structure for computing APPs of the Markov source," *IEEE Trans. Inform. Theory*, vol. 49, no. 4, pp. 1027–1029, Apr. 2003.

[PR96] L. Papke and P. Robertson, "Improved decoding with the SOVA in a parallel concatenated (turbo-code) scheme," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dallas, USA, June 1996, pp. 102–106.

[RHV97] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *Europ. Trans. Telecommun.*, vol. 8, no. 2, pp. 119–125, Mar./Apr. 1997.

[Rob94] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systmatic (turbo) codes," in *Proc. IEEE Globecom Conf.*, San Francisco, Dec. 1994, pp. 1298–1303.

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

[RU01a] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[RU01b] ——, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[Say03] J. Sayir, "Why turbo codes cannot achieve capacity," in *Proc. Int. Symp. on Turbo Codes & Rel. Topics*, Brest, France, Sept. 2003.

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

[SLF99] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–120, Aug. 1999.

[Sor02] U. Sorger, "Information transmission," Habilitationsschrift, TU Darmstadt, Germany, 2002. [Online]. Available: http://www.tu-darmstadt.de/fb/et/uet/nesi/uli/

[SS94] N. Seshadri and C.-E. W. Sundberg, "List Viterbi algorithm with applications," *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 313–323, Feb./Mar./Apr. 1994.

[SS03] S. Shamai (Shitz), personal communication, Sept. 2003.

[SSSZ03] I. Sutskover, S. Shamai (Shitz), and J. Ziv, "Extremes of information combining," in *Proc. Allerton Conf. on Communications, Control, and Computing*, Monticello, Illinois, USA, Oct. 2003.

[SSSZ05] ——, "Extremes of information combining," *IEEE Trans. Inform. Theory*, vol. 51, no. 4, pp. 1313–1325, Apr. 2005.

[Tan81] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533 –547, Sept. 1981.

[tB99a] S. ten Brink, "Convergence of iterative decoding," *Electron. Letters*, vol. 35, no. 13, pp. 1117–1119, June 1999.

[tB99b] ——, "Convergence of iterative decoding," *Electron. Letters*, vol. 35, no. 10, pp. 806–808, May 1999.

[tB00a] ——, "Iterative decoding trajectories of parallel concatenated codes," in *Proc. Int. ITG Conf. on Source and Channel Coding*, Munich, Germany, Jan. 2000, pp. 75–80.

[tB00b] ——, "Rate one-half code for approaching the Shannon limit by 0.1 dB," *Electron. Letters*, vol. 36, no. 15, pp. 1293–1294, July 2000.

[tB01a] ——, "Code characteristic matching for iterative decoding of serially concatenated codes," *Ann. Télécommun.*, vol. 56, no. 7-8, pp. 394–408, 2001.

[tB01b] ——, "Code doping for triggering iterative decoding convergence," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Washington, DC, USA, June 2001, p. 235.

[tB01c] ——, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[tBK03] S. ten Brink and G. Kramer, "Design of repeat-accumulate codes for iterative detection and decoding," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2764–2772, Nov. 2003.

[tBSY98] S. ten Brink, J. Speidel, and R.-H. Yan, "Iterative demapping for QPSK modulation," *Electron. Letters*, vol. 34, no. 15, pp. 1459–1460, July 1998.

[TL04] R. Thobaben and I. Land, "Blind quality estimation for corrupted source signals based on a-posteriori probabilities," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Chicago, USA, June/July 2004, p. 304.

[TtBH02] M. Tuechler, S. ten Brink, and J. Hagenauer, "Measures for tracing convergence of iterative decoding algorithms," in *Proc. Int. ITG Conf. on Source and Channel Coding*, Berlin, Germany, Jan. 2002, pp. 53–60.

[Ung03] G. Ungerboeck, "Iterative soft decoding of Reed-Solomon codes," in *Proc. Int. Symp. on Turbo Codes & Rel. Topics*, Sept. 2003.

[Var98] A. Vardy, "Trellis structure of codes," in *Handbook of Coding Theory*, V. Pless and W. Huffman, Eds. Amsterdam: Elsevier, 1998, pp. 1981–2117.

[vDJK03] M. van Dijk, A. J. Janssen, and A. G. Koppelaar, "Correcting systematic mismatches in computed log-likelihood ratios," *Europ. Trans. Telecommun.*, vol. 14, pp. 227–244, 2003.

[Vit67] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967.

[VY00] B. Vucetic and J. Yuan, *Turbo Codes - Principles and Applications*. Kluwer Academic Publishers, 2000.

[WHW00] A. Worm, P. Hoeher, and N. Wehn, "Turbo-decoding without SNR estimation," *IEEE Commun. Lett.*, vol. 4, no. 6, pp. 193–196, June 2000.

[Wib96] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linkoping University, Sweden, 1996.

[WLK95] N. Wiberg, H.-A. Loeliger, and R. Koetter, "Codes and iterative decoding on general graphs," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Sept. 1995, p. 468.

[Wol78] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inform. Theory*, vol. 24, no. 1, pp. 76–80, Jan. 1978.

[WWE00] Y. Wu, B. D. Woerner, and W. J. Ebel, "A simple stopping criterion for turbo decoding," *IEEE Commun. Lett.*, vol. 4, no. 8, pp. 258–260, Aug. 2000.

[WZ73] A. D. Wyner and J. Ziv, "A theorem on the entropy of certain binary sequences and applications: Part I," *IEEE Trans. Inform. Theory*, vol. 19, no. 6, pp. 769–772, Nov. 1973.

[ZSB99] V. Zyablov, S. Shavgulidze, and M. Bossert, "An introduction to generalized concatenated codes," *Europ. Trans. Telecommun.*, vol. 10, no. 6, pp. 609–622, Nov./Dec. 1999.