Project no: FP6-2002-IST-C 020023-2

Project title: FlexCode

Instrument: STREP

Thematic Priority: Information Society Technologies

D2.2: Baseline Channel Coder

Due date of deliverable: 2008-02-01

Actual submission date: 2008-02-01

Start date of project: 2006-07-01 Duration: 36 Months

Organisation name of lead contractor for this deliverable: RWTH Aachen

Revision: 1.1

Project co-funded by the European Commission within the Sixth Framework Programme 2002-2006		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Dipl.-Ing. Laurent Schmalen:
Institute of Communication Systems and Data Processing
RWTH Aachen University
Aachen, Germany
`schmalen@ind.rwth-aachen.de`

Prof. Dr.-Ing. Peter Vary:
Institute of Communication Systems and Data Processing
RWTH Aachen University
Aachen, Germany
`vary@ind.rwth-aachen.de`

Dr.-Ing. Thorsten Clevorn:
now with Infineon Technologies
COM Development Center NRW
Duisburg, Germany

Table of Contents

Chapter 1

Introduction

The FlexCode concept of flexible source coding poses interesting challenges for channel coding and transmission. At a first glance, the variety of different FlexCode scenarios defined in [Fle07b] seems to require not only a single, but several distinct channel coders for the different scenarios, transmission schemes and storage media. For instance, in storage scenarios, channel coding is not necessary, but a strong compression of the data is required. To accomplish a flexible, near-entropy compression, *arithmetic codes* [BCW90], [BCK07] are frequently used. However, already a single bit error in the arithmetically coded bitstream causes a complete decoder failure and the loss of large parts of the original data. This means that in wireless transmission scenarios, strong channel coding has to be employed in order to guarantee error-free transmission. For this reason, in most of the current speech and audio codecs the redundancy in the audio signal is removed using techniques like linear prediction and/or vector quantization [VM06]. The concept of *constrained entropy* quantization, which is a candidate to be used in the FlexCode source coder, leaves a large amount of redundancy in the quantized signal. This residual redundancy is then usually removed using standard data compression algorithms such as arithmetic coding. Instead of removing this redundancy, the source encoder can leave the redundancy in the bitstream and use it at the receiver to help combat any effects introduced by channel noise.It has been shown that it is possible to efficiently exploit this residual redundancy of the parameters in an iterative Turbo-like process at the receiver [AVS01]. Furthermore, the best practical channel coders known so far utilize iterative receivers [RU08]. Therefore, channel codes with iterative receivers have been chosen as a candidate for the FlexCode source coder.
In this report, the FlexCode baseline channel coder is introduced. The report is organized as follows: Chapter 2 presents the basics of modern iterative channel coding techniques. Turbo codes are introduced as well as analysis tools such as EXIT charts and basic concepts of interleaver design. Chapter 3 introduces the concept of soft decision source decoding (SDSD) and extends it for the deployment in iterative Turbo-like decoders. Chapter 4 presents some of the advances and optimizations of the iterative source-channel decoding scheme that were made during the FlexCode project in order to achieve a more flexible transmission scheme. Chapter 5 then presents the adaptation of the ISCD scheme to the FlexCode source coding concept. Finally, Chapter 6 briefly outlines the practical realization of the channel encoder and the interface between source and channel coder.

Chapter 2

Modern Channel Coding Techniques

The purpose of this chapter is to give a brief overview of the concepts used in modern channel coding. We mainly focus on Turbo codes as the Turbo principle [Hag02] allows near Shannon-limit error correction and transmission and leaves the transmission chain flexible enough to fulfill the FlexCode needs. On the other hands, Low-Density Parity-Check (LDPC) codes, originally introduced by Gallager [Gal63], [Gal62], then forgotten and rediscovered by MacKay [Mac99], [MN96], have not been chosen for the FlexCode channel coder. The reason is that LDPC codes need large precomputed parity check matrices. These are usually optimized using an offline, time-consuming search algorithm. Once a parity check matrix is found, the code is fixed, i.e., the block sizes and the rates are fixed and cannot be adapted on the fly. Changing for instance the code rate implies that a whole new parity check matrix (and thus a different generator matrix) has to be chosen. For this reason, a channel coding scheme using the Turbo principle instead of LDPC codes have been chosen for FlexCode.
In this chapter, the underlying theoretical concepts of the FlexCode channel coder are presented. First, a brief introduction of the Turbo principle is given in Sec. 2.1, followed by introducing the key element in iterative decoders, the so-called extrinsic information, in Sec. 2.2. Afterwards, Turbo codes are introduced in Sec. 2.3, followed by EXIT charts, which are the most widely used tool to tracing convergence and optimization of iterative transmission schemes, in Sec. 2.4. Finally, the importance of a proper interleaver design is highlighted in Sec. 2.5.
Parts of this chapter are reproduced from [Cle06]

2.1 The Turbo Principle

The Turbo principle [Hag97], [Hag02] describes the iterative beneficial exchange of extrinsic information between separate components at the receiver. The extrinsic information provided by one component improves the performance of another component by serving as a priori information. The key elements in this iterative process are the propagation of extrinsic soft information, which is generated by the components, and the independence of this extrinsic information, which is usually ensured by interleaving. By Turbo processing it is possible to obtain a performance close to the capacity limits by using relatively simple components. The Turbo principle was discovered in 1993 for the parallel concatenation of two convolutional codes [BG96]. The Turbo decoding of serially concatenated channel codes is discussed for instance in [BDMP98b]. Furthermore, various combinations of parallel and serial concatenation are possible [BM02]. The Turbo principle is not restricted to channel codes but can easily be applied to other components of the receiver chain as long as these components can provide suitable extrinsic soft information. Examples are *Bit Interleaved Coded Modulation with Iterative Decoding* [LR97], [CTB98] (BICM-ID), *Iterative Source-Channel Decoding* [AVS01], [Gor01], *Turbo equalization* [DPD$^+$95], [TKS02], Turbo multiuser detection [Poo00], and Turbo DeCodulation [CBAV05], [Cle06].

2.2 Extrinsic Information

The term extrinsic information describes the novel additional information a Turbo component, e.g., an appropriately designed Trellis based BCJR or SISO channel decoder [BCJR74], [BMDP98], can generate for a certain bit based on the a priori information for all bits of the bit stream, excluding the a priori (or intrinsic) information for the considered bit itself. The combination of the extrinsic information of a bit with its own intrinsic a priori information is the a posteriori information. The a priori information for a bit can comprise, e.g., the extrinsic information of other components of the Turbo process (dynamically changing in decoding iterations), direct information on the received symbols or bits (channel related knowledge, static per frame), and a priori information due to unequally distributed bits (static). This latter classification of input (a priori) information for a Turbo component is depicted in Fig. 2.1. Thus, there exist two different classifications for a priori information, one considering the relation, intrinsic or extrinsic, to a certain bit position and the other one considering the source of the information (see Fig. 2.1).

The information in a Turbo process are reliabilities for the bits, which are usually represented as probabilities or log-likelihood ratios (L-values) [HOP96]. An L-value $L(x^j)$ is the natural logarithm of the ratio of the probabilities for the two realizations of a binary random variable x^j, possibly conditioned on other variables,

$$L(x^j) \triangleq L(x^j|\ldots) = \log \frac{P(x^j = 0|\ldots)}{P(x^j = 1|\ldots)}\,. \tag{2.1}$$

In the logarithmic domain of L-values the multiplication and division of probabilities transforms to addition and subtraction, respectively, which can be convenient for efficient implementations. The addition of probabilities yields a so-called box-plus $\boxplus$ operation [HOP96] for the corresponding L-values.

The key element of the Turbo principle is the refinement of extrinsic information by its exchange between several components. Exemplarily, Fig 2.2 shows the spreading or collection of extrinsic information in

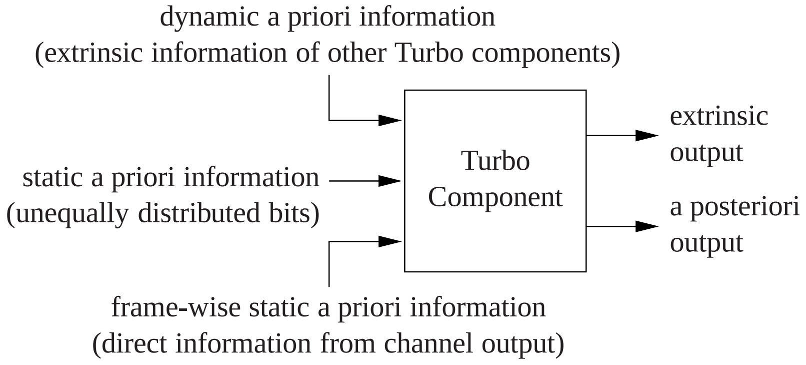

dynamic a priori information
(extrinsic information of other Turbo components)
static a priori information
(unequally distributed bits)
Turbo
Component
extrinsic
output
a posteriori
output
frame-wise static a priori information
(direct information from channel output)

Figure 2.1: Types and sources of input (a priori) information for Turbo components.

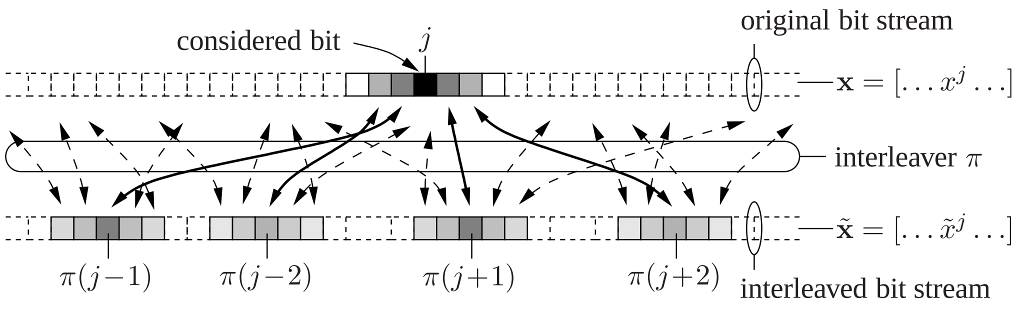
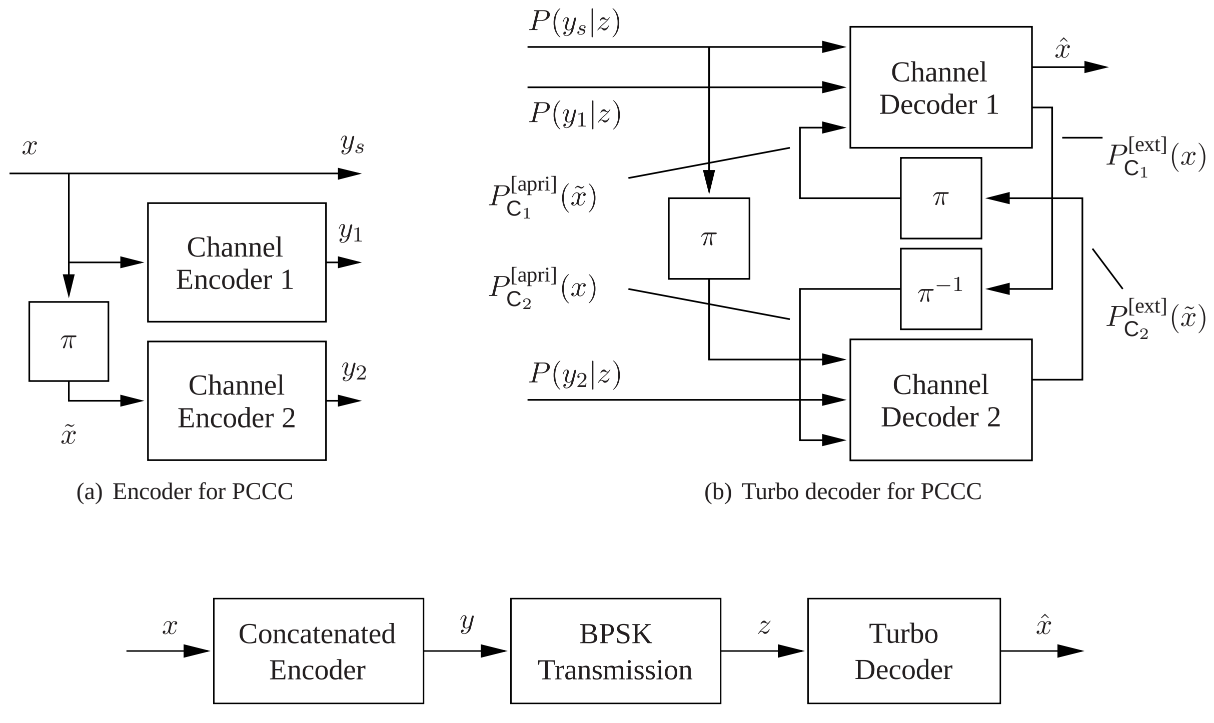


Figure 2.2: Spreading (downwards) or collection (upwards) of extrinsic information.



(a) Encoder for PCCC

(b) Turbo decoder for PCCC

(c) Baseband transmission model

Figure 2.3: Encoder and Turbo decoder for parallel concatenated convolutional codes.

a Turbo process with two components connected via an interleaver π. The information for bit j in the original bit stream is propagated two bit positions in both directions. After interleaving this extrinsic information it is used in four far apart positions in the interleaved bit stream. There, it is again propagated to the neighboring positions. Until now $4 \cdot 5 = 20$ bits profited from the single information. In the next step this extrinsic information is deinterleaved and propagated again in the original bit stream. If the interleaver is well designed and sufficiently large so that no overlapping occurs in the propagation, the information for the bit at position j is now spreaded already to close to 100 other bits. The inverse process can be considered as the collection of extrinsic information. Both, the spreading and the collection of extrinsic information, take place in a Turbo process. On the one hand, every input information shall be spread as far as possible to improve the decoding at other bit positions. On the other hand, as much as possible extrinsic information on a certain bit shall be collected for a reliable decision.

2.3 Turbo Codes

2.3.1 Parallel Concatenation

Originally, the Turbo principle was discovered for parallel concatenated convolutional codes (PCCC) [BG96]. The encoder of a PCCC similar to [BG96] is depicted in Fig. 2.3(a). The data bits x are encoded separately by the two channel encoders to the encoded bits y_1 and y_2. However, before encoding the data bits by the second encoder, they are permuted by a bit interleaver π. For the case of PCCC the component codes are usually rate $r_{\mathsf{C}} = 1$ convolutional codes. Additionally, the data bits are transmitted as systematic bits $y_s = x$. The overall code rate then is $r_{\mathsf{C}_{\text{PCCC}}} = 1/3$. A $r_{\mathsf{C}_{\text{PCCC}}} = 1/2$ PCCC can be obtain by alternately puncturing of half of the parity bits y_1 and y_2. Due to the deterministic interleaving, the bits y_s can serve as systematic bits for both components codes. Thus, the sub-codes consisting

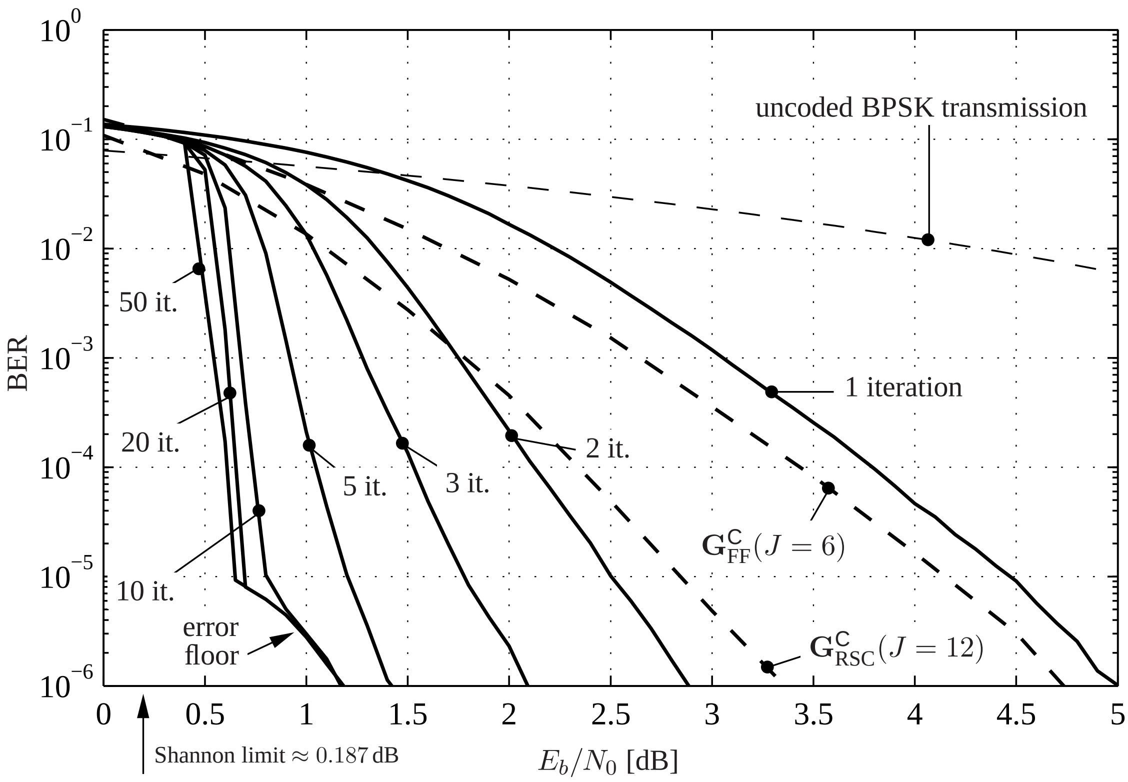
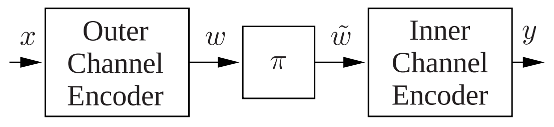
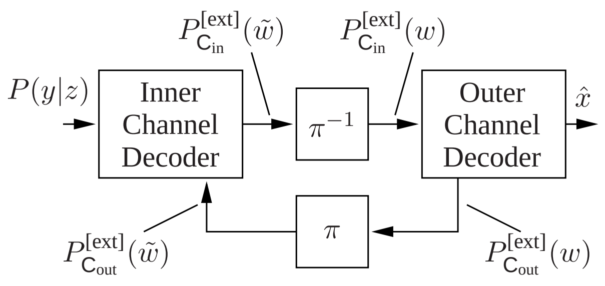


Figure 2.4: BER performance for Turbo decoding of the PCCC of [BG96]. $\mathbf{G}_{\text{RSC}}^{\mathsf{C}_{1,2}} = \{1, 21/37\}_8$, $r_{\mathsf{C}_{1,2}} = 2/3$, $r_{\mathsf{C}_{\text{PCCC}}} = 1/2$, $V = 65536$ bit/frame, AWGN.

of the systematic bit and the partially punctured parity bit can be considered as recursive systematic convolutional (RSC) codes with $r_{\mathsf{C}_{1,2}} = 2/3$ (or $r_{\mathsf{C}_{1,2}} = 1/2$ for $r_{\mathsf{C}_{\text{PCCC}}} = 1/3$).
The bits y are transmitted to the receiver. As transmission model (see Fig. 2.3(c)) serves BPSK transmission over an AWGN channel. The received value for each transmitted bit y is denoted by z.
Figure 2.3(b) shows the Turbo decoder for the PCCC encoder of Fig. 2.3(a). Each of the component decoders receives three input reliabilities: two reliabilities, $P(y_s|z)$ and $P(y_{1,2}|z)$ directly from the channel (or demodulator) and one properly (de)interleaved extrinsic reliability $P_{\mathsf{C}_{2,1}}^{[\text{ext}]}$ from the other decoder as a priori input $P_{\mathsf{C}_{1,2}}^{[\text{apri}]}$. Appropriately designed SISO decoders [BMDP98] are used as component decoders. For simplicity, we assume equiprobable bits $x = y_s$, i.e., $P(y_s = 0) = P(y_s = 1) = \frac{1}{2}$. A priori information on the bits y_s could be included in the reliabilities $P(y_s|z)$. After the desired number of iterations, in which the two decoders alternately update their own extrinsic information by using the extrinsic information supplied by the other decoder, the final (hard decision) output $\hat{x}$ is determined. For more details we refer to the literature [BG96], [HLY02], [VY00].
In Fig. 2.4 the bit error rate (BER) performance of the original exemplary Turbo decoding of a PCCC in [BG96] is depicted for BPSK transmission. The two effective constituent codes $\mathsf{C}_{1,2}$ are memory $J = 4$, rate $r_{\text{RSC}} = 1/2$ RSC codes with generator polynomials $\mathbf{G}_{\text{RSC}}^{\mathsf{C}_{1,2}} = \{1, 21/37\}_8$ punctured to $r_{\mathsf{C}_{1,2}} = 2/3$ to achieve an overall code rate of $r_{\mathsf{C}_{\text{PCCC}}} = 1/2$. A pseudo-random interleaver of size $V = 65536$ bits is used.
Tremendous gains for the first five iterations can be observed. For more iterations the particular additional gain decreases. Furthermore, an error floor can be observed. For comparison, the performance of an uncoded BPSK transmission and two standalone convolutional codes is given. One is the memory $J = 6$,



(a) Encoder for SCCC


$P(y|z)$
$P^{[\text{ext}]}_{\mathsf{C}_{\text{in}}}(\tilde{w})$
$P^{[\text{ext}]}_{\mathsf{C}_{\text{in}}}(w)$
Inner Channel Decoder
π^{-1}
Outer Channel Decoder
$\hat{x}$
π
$P^{[\text{ext}]}_{\mathsf{C}_{\text{out}}}(\tilde{w})$
$P^{[\text{ext}]}_{\mathsf{C}_{\text{out}}}(w)$

(b) Turbo decoder for SCCC

Figure 2.5: Encoder and Turbo decoder for serial concatenated convolutional codes.

$r_{\text{FF}} = 1/2$ feed-forward (FF) convolutional code with the generator polynomial $\mathbf{G}^{\mathsf{C}}_{\text{FF}} = \{171, 133\}_8$ which is used, e.g., in the WLAN 802.11a standard. The other one is a memory $J = 12$, $r_{\text{RSC}} = 1/2$ RSC code with $\mathbf{G}^{\mathsf{C}}_{\text{RSC}} = \{1, 10533/17551\}_8$. After already two iterations the PCCC with in total $2{\cdot}2^{J=4} = 32$ states outperforms both of these relatively strong codes with $2^{J=6} = 64$ resp. $2^{J=12} = 4096$ states. Note, the Shannon limit for this case is $(E_b/N_0)_{\text{min}} \approx 0.187\,\text{dB}$ and with 20 or 50 iterations we can approach it at a BER of $P_{\text{b}} = 10^{-5}$ by approximately $\Delta_{E_b/N_0} \approx 0.5\,\text{dB}$.

2.3.2 Serial Concatentation

Another possibility for the concatenation of channel codes is the serial concatenation as used for serial concatenated convolutional codes (SCCC) [BM02], [Tüc04]. This type of concatenation usually better resembles the serial design of the receiver chain, especially when the Turbo process comprises also other components than channel decoding. In Fig. 2.5 the encoder and the Turbo decoder for a SCCC are depicted. Note that the bits w, which experience the Turbo processing, are the encoded bits of the outer code but the decoded bits of the inner code. The decoders must be accordingly designed to provide the required extrinsic information.
In Fig. 2.6 simulation results are shown for a SCCC presented in [tB00a]. A memory $J = 2$, $r_{\mathsf{C}_{\text{out}}} = 1/2$ RSC code is concatenated with a $J = 1$, $r_{\mathsf{C}_{\text{in}}} = 1$ recursive non-systematic convolutional (RNSC) code as inner component. The respective generator polynomials are $\mathbf{G}^{\mathsf{C}_{\text{out}}}_{\text{RSC}} = \{1, 5/7\}_8$ and $\mathbf{G}^{\mathsf{C}_{\text{in}}}_{\text{RNSC}} = \{2/3\}_8$. Due the $r_{\mathsf{C}_{\text{in}}} = 1$ inner code the overall code rate is still $r_{\mathsf{C}_{\text{SCCC}}} = 1/2$. Furthermore, $r_{\mathsf{C}_{\text{in}}} = 1$ complies with the findings in [AKtB02], [AKtB04], [Tüc04] that the inner component of a serially concatenated system should be of rate $r_{\text{in}} = 1$. Similar to [tB00a] a large frame size of 200000 bit/frame is used, resulting in an interleaver of size $V = 400000$ bit.
Even with these very weak convolutional component codes with only 4 and 2 Trellis states, the Shannon limit can be approached by less than $\Delta_{E_b/N_0} < 1\,\text{dB}$. The reference standalone convolutional codes (see Section 2.3.1) are outperformed after two and three iterations, respectively. An almost vertical waterfall region for 10, 20, and 50 iterations can be observed. Furthermore, no error floor is visible in contrast to the PCCC case in Fig. 2.4.

2.4 EXIT Charts

Many different approaches have been studied for the analysis of the convergence behavior of Turbo processes. They are targeted at a suitable design method and a better understanding of the Turbo processing. Some approaches are compared in [TtBH02]. The most widely used tool today is the extrinsic information transfer (EXIT) chart [tB01a], [tB99] which is based on the mutual information measure [CT06]

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)\,. \tag{2.2}$$

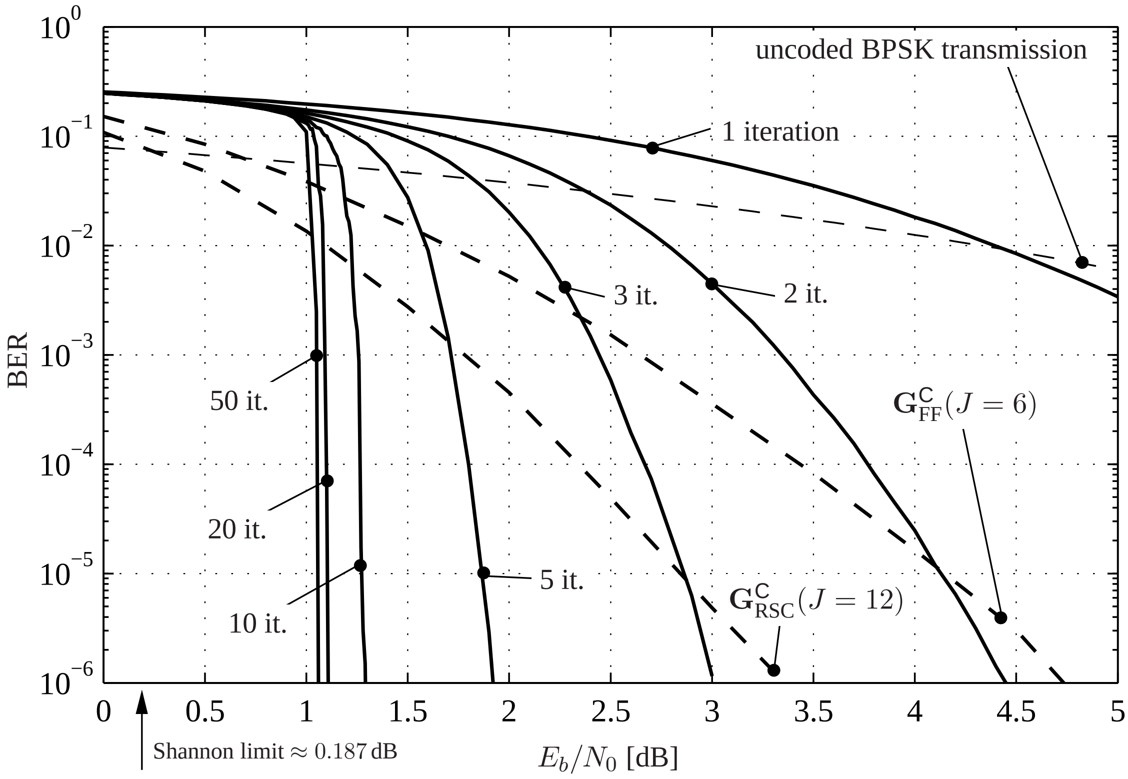


Figure 2.6: BER performance for Turbo decoding of the SCCC of [tB00a]. $\mathbf{G}^{\mathsf{C}_{\mathrm{out}}}_{\mathrm{RSC}} = \{1, 5/7\}_8$, $r_{\mathsf{C}_{\mathrm{out}}} = 1/2$, $\mathbf{G}^{\mathsf{C}_{\mathrm{in}}}_{\mathrm{RNSC}} = \{2/3\}_8$, $r_{\mathsf{C}_{\mathrm{in}}} = 1$, $r_{\mathsf{C}_{\mathrm{SCCC}}} = 1/2$, 200000 bit/frame.

The mutual information $I(X;Y)$ specifies the amount of information one random variable X contains in average about another random variable Y.
In an EXIT chart the extrinsic mutual information $I^{[\mathrm{ext}]}$ exchanged in each iteration of a Turbo process is plotted as a decoding trajectory. Furthermore, each component is additionally separately described by its EXIT characteristic $\mathcal{T}$, i.e., the relation between the extrinsic mutual information $I^{[\mathrm{ext}]}$ the component can generate for a certain input a priori mutual information $I^{[\mathrm{apri}]}$.
Usually it is assumed that the L-values L of the respective bipolar variable $\ddot{l}$ is Gaussian distributed with variance σ_L^2 and mean $\mu_K = \sigma_L^2/2$ [tB01a], i.e., the conditional probability density function for the L-value is

$$P(L|\ddot{l} = \pm 1) = \frac{1}{\sqrt{2\pi}\sigma_L} \exp\left(-\frac{(L - \mu_L \ddot{l})^2}{2\sigma_L^2}\right) = \frac{1}{\sqrt{2\pi}\sigma_L} \exp\left(-\frac{(L - \frac{\sigma_L^2}{2}\ddot{l})^2}{2\sigma_L^2}\right). \tag{2.3}$$

In general the mutual information is given by [tB01a]

$$I = \frac{1}{2} \sum_{\ddot{l}=\pm 1} \int_{-\infty}^{\infty} P(L|\ddot{l}) \log_2 \frac{2 \cdot P(L|\ddot{l})}{P(L|\ddot{l} = +1) + P(L|\ddot{l} = -1)} \mathrm{d}L\,. \tag{2.4}$$

With the assumption of (2.3) this function is abbreviated by

$$I = \mathcal{J}(\sigma_L) \tag{2.5}$$

and cannot be expressed in close form [tB01a], implying usually numerical evaluation.

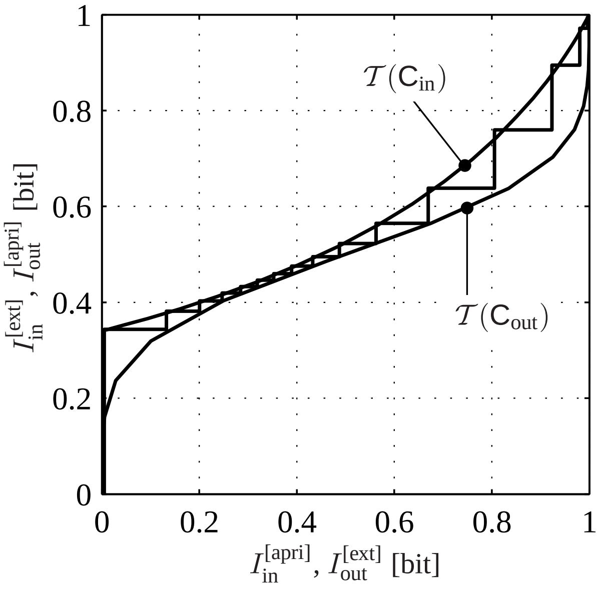


Figure 2.7: EXIT chart and characteristics for Turbo decoding of the SCCC of [tB00a] at $E_b/N_0 = 1.1\,\text{dB}$ for 50 iterations. $\mathbf{G}_{\text{RSC}}^{\mathsf{C}_{\text{out}}} = \{1, 5/7\}_8$, $r_{\mathsf{C}_{\text{out}}} = 1/2$, $\mathbf{G}_{\text{RNSC}}^{\mathsf{C}_{\text{in}}} = \{2/3\}_8$, $r_{\mathsf{C}_{\text{in}}} = 1$, $r_{\mathsf{C}_{\text{SCCC}}} = 1/2$, 200000 bit/frame.

Based on the inverse function $\sigma_L = \mathcal{J}^{-1}(I^{[\text{apri}]})$ Gaussian distributed L-values are generated and processed in the considered Turbo receiver component together with other potential a priori information, which does not originated from the virtual other Turbo component. The output L-values are recorded in two histograms for $\ddot{l} = +1$ and $\ddot{l} = -1$. Note that the "random" values of $\ddot{l}$ must of course fulfill the inherent coding rules of the Turbo component. When filled with a sufficient number of samples, finally the histograms are evaluated using (2.4) to compute the extrinsic mutual information $I^{[\text{ext}]}$. For the decoding trajectory similar histograms are measured for every extrinsic information $I^{[\text{ext}]}$ in every iteration of a simulation of the complete Turbo system.

In Fig. 2.7 the EXIT characteristics $\mathcal{T}(\mathsf{C})$ of the component codes of the SCCC example in Fig. 2.6 are depicted. As visible in Fig. 2.5(b), the inner component decoder accepts two types of input information, channel related knowledge $I^{[\text{channel}]}$ and refined *a priori* knowledge $I^{[\text{apri}]}$ from the outer decoder. Thus, the EXIT characteristic $\mathcal{T}(\mathsf{C}_{\text{in}}) : I_{\text{in}}^{[\text{apri}]} \mapsto I_{\text{in}}^{[\text{ext}]}$ depends on the channel quality, too. On the other hand , the outer component decoder only accepts refined *a priori* knowledge $I^{[\text{apri}]}$ from the inner decoder. Thus, the EXIT characteristic $\mathcal{T}(\mathsf{C}_{\text{out}}) : I_{\text{out}}^{[\text{apri}]} \mapsto I_{\text{out}}^{[\text{ext}]}$ does not depend on the channel quality. Furthermore, the EXIT characteristics $\mathcal{T}(\mathsf{C})$ depend of course also on the channel code C itself. In Fig. 2.7 it can be observed that the EXIT characteristic $\mathcal{T}(\mathsf{C}_{\text{in}})$ for $\mathbf{G}_{\text{RNSC}}^{\mathsf{C}_{\text{in}}} = \{2/3\}_8$ matches very well to the swapped EXIT characteristic $\mathcal{T}(\mathsf{C}_{\text{out}})$ of the outer code and a small decoding tunnel is open at $E_b/N_0 = 1.1\,\text{dB}$.

In an EXIT chart the EXIT characteristics of the two Turbo components are plotted together in one diagram. However, since the extrinsic output of one component is the a priori input of the other one, the EXIT characteristic of one component is plotted with swapped axes. These EXIT characteristics form a kind of bound. Additionally, the EXIT chart contains the decoding trajectory. This is a step curve which depicts the generated extrinsic information by each component in each iteration. In Fig. 2.7 an upward step of the trajectory corresponds to the additional available extrinsic information generated by the inner decoder C_{in}. An advancement to the right denotes the same for the outer decoder C_{out}. The aim is that this decoding trajectory reaches the upper right corner of the EXIT chart because with perfect mutual information error free decoding is possible.

For more details on EXIT charts and their application to the different iterative transmission schemes, we refer to the literature [AKtB04], [TtBH02], [tB01a], [tB01b], [tB00a], [tB99], [tB00b], [SVAC07], [AV05], [ABCV05], [SCV07].
The original EXIT chart papers mainly considered recursive convolutional codes. However, feed forward convolutional codes have been used in a variety of existing communication systems, e.g., GSM and UMTS. However, due to their suboptimal performance when used in iterative decoders [BDMP98a], not much analysis has been performed for Turbo-like systems employing feed forward convolutional codes as inner channel codes. However, it can be beneficial to perform iterative decoding also in existing systems. For instance, the application of iterative source-channel decoding might be advantageous in speech transmission systems like GSM and UMTS [PKH01], [AV05], [GS04]. To analyze such systems, EXIT charts can be a powerful tool.
In Appendix A we analyze the EXIT charts of feed forward convolutional codes, especially the property that no perfect extrinsic information can be generated even if perfect *a priori* knowledge is available [SVAC07]. The mutual information of the extrinsic output is upper bounded, depending on the code and the channel quality. First, we give an easy explanation of this behavior using the Trellis diagram of the convolutional code and secondly, we show by analytical means that the maximum attainable mutual information only depends on the channel quality and the Hamming weight of the impulse response of the code. We also give an expression to calculate this mutual information.
An interesting property of the EXIT charts concerns the areas $\mathcal{A}$ under the EXIT characteristics. The area $\mathcal{A}_{\text{in}}$ under the EXIT characteristic $\mathcal{T}_{\text{in}}$ of the inner component consists of the light and the dark shaded region. As the EXIT characteristic $\mathcal{T}_{\text{out}}$ of the outer component is plotted with swapped axes, the dark shaded area $1 - \mathcal{A}_{\text{out}}$ is relevant. A necessary (but not sufficient) condition for successful decoding, i.e., an open decoding tunnel in the EXIT chart, is [AKtB02], [AKtB04]

$$\mathcal{A}_{\text{in}} > 1 - \mathcal{A}_{\text{out}} \quad \text{or} \quad \mathcal{A}_{\text{in}} - (1 - \mathcal{A}_{\text{out}}) > 0\,. \tag{2.6}$$

This relation provides a quick assessment if successful decoding may be possible.

2.5 Interleaver Design

A key design element regarding the performance of a Turbo process is the type and the size of the interleaver(s) [BDMP98b], [DDP98b], [DDP98a], [BM02]. Often very large random interleavers are considered for demonstration, e.g., $V = 65536$ bit in [BG96].
In Fig. 2.8 the performance of random interleavers and block interleavers is compared for different block sizes using the PCCC of Section 2.3.1. As random interleavers we use S-random interleavers [DP95b], [DP95a], [VY00], which give a random permutation with the additional constraint that the distances between all interleaved positions of S_1 adjacent bits are at least S_2. Often, the parameters S_1 and S_2 are chosen such that $S_1 = S_2 = S$. Such interleavers can be generated with low computational complexity if $S < \sqrt{\frac{V}{2}}$. Technically, an interleaver realizes a permutation $\pi(i)$, on the interval $[0, V) \subset \mathbb{N}$, with π being the bijective function

$$\begin{aligned} \pi : [0;V) &\rightarrow [0;V) \\ i &\mapsto \pi(i)\,. \end{aligned}$$

An interleaver is an S-random interleaver with $S = S_1 = S_2$ if for any two indices i and j such that

$$0 < |i - j| \leq S\,, \tag{2.7}$$

the S-random design imposes that

$$|\pi(i) - \pi(j)| > S\,. \tag{2.8}$$

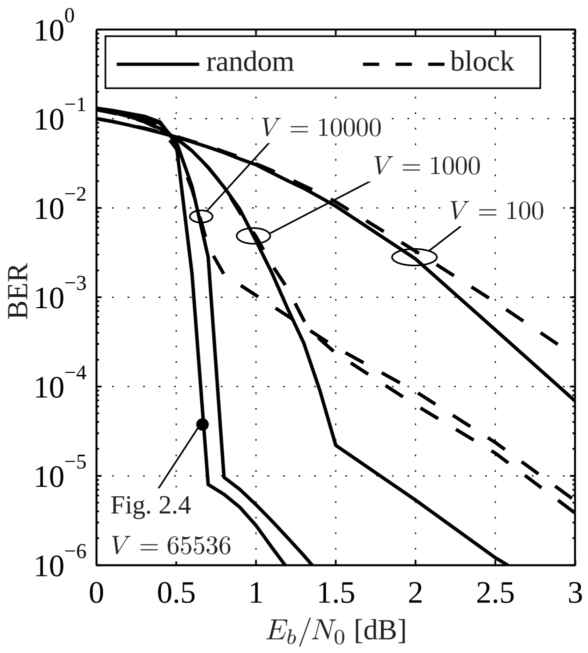


Figure 2.8: Effects of interleaver size V and type (random or block) on bit error rate performance of Turbo decoding. PCCC of [BG96], 20 iterations

The conditions (2.7) and (2.8) are called the S-conditions for interleavers.
As shown in Fig. 2.8 the simulation results degrade for a decreasing interleaver size and block interleavers yield an unacceptable higher error floor. The degradation when block interleaving is used is caused by the disadvantageous spreading (or collection) of the extrinsic information by the interleaver.

Chapter 3

Iterative Source-Channel Decoding

In the first section of this chapter, the basics of *Soft Decision Source Decoding* (SDSD) and *Iterative Source-Channel Decoding* (ISCD) are briefly presented.
In order to describe the algorithms in this chapter, we do not use the FlexCode source coder model which will be introduced in Sec. 5.1 but a rather simplified model. A source generates samples u which are grouped into a vector $\mathbf{u}$ and then quantized to $\bar{\mathbf{u}}$ by the Q-level quantizer with representation levels $\bar{\mathbf{u}}^{(i)}$, $i = 1, \ldots, Q$. In the case that scalar quantization is utilized, the parameter vector $\mathbf{u}$ and the quantized representation $\bar{\mathbf{u}}$ only contain a single element. After quantization, the index assignment maps a bit pattern $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ of length M to the quantized vector $\bar{\mathbf{u}}$. Usually M is chosen such that $M = \lceil \log_2(Q) \rceil$ (with $\lceil \nu \rceil$ denoting the smallest integer number larger than ν). As a distinct bit pattern exists for each quantizer level, a total number of Q distinct bit patterns $\mathbf{x}^{(i)}$ with $i = 1, \ldots, Q$ exist. The bit pattern $\mathbf{x}$ may then be subject to channel coding or storage/transmission.
The simple transmission system used to explain Soft Decision Source Decoding (SDSD) is depicted in Fig. 3.1. Instead of considering a specific channel encoder and channel, an equivalent channel which models the behavior of channel encoder, transmission, and channel decoder is utilized. The equivalent channel delivers the transmitted bit sequence $\tilde{\mathbf{x}}$ as well as reliability information about the different bits of the sequence or average reliability information about the whole sequence.

3.1 Soft Decision Source Decoding (SDSD)

A robust algorithm for parameter decoding is the so-called *Soft Decision Source Decoding* (SDSD) [FV01], [VM06] which performs an estimation of the received parameters (e.g. the residual signal radius) using soft information, i.e., reliabilities/probabilities of the received values. SDSD uses the knowledge of the channel quality and of the residual redundancy remaining in the signal after source

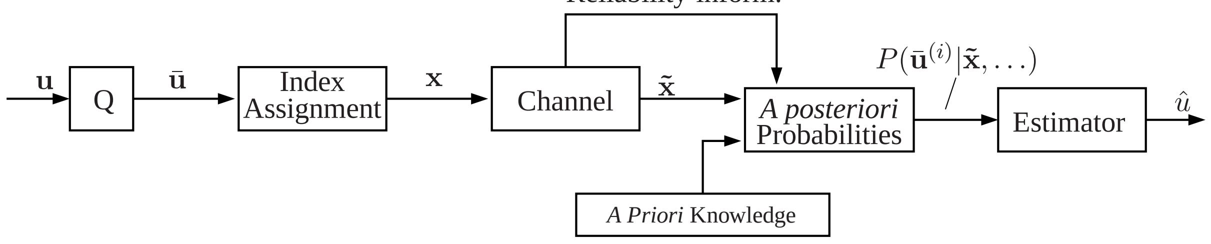


Figure 3.1: Abstract transmission model for Soft Decision Source Decoding

coding to compute a better reconstruction of the original parameter. In order to perform the estimation, the receiver needs reliability information on the received bit patterns. This reliability information is delivered by the equivalent channel; in the case that channel coding is utilized, the reliability information may be obtained, e.g., using a soft-in/soft-out (SISO) channel decoder, or even a hard-in/soft-out (HISO) channel decoder (see Sec. 4.1.1).
For each of the Q possible bit patterns $\mathbf{x}^{(i)}$, the receiver first calculates the probabilities

$$P(\tilde{\mathbf{x}}|\mathbf{x}^{(i)}) = \prod_{\kappa=1}^{M} P(\tilde{x}_\kappa|x_\kappa^{(i)})\,, \qquad i = 1, \ldots, Q\,. \tag{3.1}$$

If L-values [HOP96] are used to represent the reliability information, the probabilities $P(\tilde{x}_\kappa|x_\kappa^{(i)})$ are determined as [HOP96], [VM06]

$$P(\tilde{x}_\kappa|x_\kappa^{(i)} = \pm 1) = \frac{1}{1 + \exp\left(\mp L\left(x_\kappa^{(i)}\right)\right)}\,.$$

The probabilities $P(\tilde{\mathbf{x}}|\mathbf{x})$ could already be used to estimate the transmitted bit pattern $\hat{\mathbf{x}}$ and thus get an estimate of the reconstructed parameter $\hat{u}$. This approach is denoted in SDSD-NAK (No *A priori* Knowledge) in [Fin98], [FV01]. However, usually the parameters which have to be transmitted exhibit different amounts of *a priori* knowledge, which can be for instance an unequal probability of occurrence and correlations in time. In the following, both types of *a priori* knowledge that can be exploited by the SDSD will be presented.
The remaining residual redundancy in the parameters can modelled as a Markov process. A Markov process of order N is defined by

$$P(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}, \ldots, \mathbf{x}_{\tau-N}, \mathbf{x}_{\tau-N-1}, \ldots) = P(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}, \ldots, \mathbf{x}_{\tau-N}). \tag{3.2}$$

For complexity reasons, the SDSD algorithm only considers *a priori* knowledge of order 0 and of order 1. Those are given by

$$\begin{aligned} P(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}, \ldots) &= P(\mathbf{x}_\tau) && \text{(AK0)} \\ P(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}, \ldots) &= P(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}) && \text{(AK1)}\,. \end{aligned} \tag{3.3}$$

and denoted as AK0 (*a priori* knowledge of 0th order) and AK1 (*a priori* knowledge of 1st order) according to [Fin98]. Under the condition that the transitions from $\mathbf{x}_{\tau-1}$ to $\mathbf{x}_\tau$ describe a stationary process, the *a priori* probabilities can be either measured using large training database or determined analytically. The analytical determination of *a priori* knowledge will be used within the FlexCode project.
Using the probabilities of the received bit patterns calculated in (3.1) as well as the *a priori* knowledge of the source parameters, *a posteriori* probabilities can be calculated. For delay reasons, the algorithm only relies on the past transmitted parameters. If only the unequal distribution of parameters is exploited by the source decoder, the approach to determine the *a posteriori* probabilities is denoted as AK0 and the *a posteriori* probabilities are given by

$$P(\mathbf{x}^{(i)}|\tilde{\mathbf{x}}) = \frac{1}{C} P(\tilde{\mathbf{x}}|\mathbf{x}^{(i)}) \cdot P(\mathbf{x}^{(i)}) \qquad \text{(AK0)}\,. \tag{3.4}$$

If temporal dependencies shall be utilized by the SDSD, the *a posteriori* probabilities can be calculated using the following recursion

$$P(\mathbf{x}_\tau^{(i)}|\tilde{\mathbf{x}}_\tau, \tilde{\mathbf{x}}_{\tau-1}) = \frac{1}{C} \cdot P(\tilde{\mathbf{x}}_\tau|\mathbf{x}_\tau^{(i)}) \cdot \sum_{j=1}^{Q} P(\mathbf{x}_\tau^{(i)}|\mathbf{x}_{\tau-1}^{(j)}) P(\mathbf{x}_{\tau-1}^{(j)}|\tilde{\mathbf{x}}_{\tau-1}, \tilde{\mathbf{x}}_{\tau-2}) \qquad \text{(AK1)}\,. \tag{3.5}$$

For time instant $\tau = 0$, the previous *a posteriori* probabilities needed in the recursion are initialized by the probability mass function of the parameter distribution:

$$P(\mathbf{x}_{\tau-1}^{(j)}|\tilde{\mathbf{x}}_{\tau-1}, \tilde{\mathbf{x}}_{\tau-2})\Big|_{\tau=0} = P(\mathbf{x}_{-1}^{(j)}|\tilde{\mathbf{x}}_{-1}, \tilde{\mathbf{x}}_{-2}) = P(\mathbf{x}^{(j)})\,. \tag{3.6}$$

Using the *a posteriori* probabilities from (3.4) or (3.5), the parameters can be estimated using MMSE or MAP estimation [Fin98]. In the FlexCode channel coder, only MAP estimation of the parameters is considered. In the case of the AK1 algorithm, the MAP estimator which estimates the optimal reconstructed parameter vector $\hat{\mathbf{u}}$ can be described as

$$\hat{\mathbf{u}}_\tau = \bar{\mathbf{u}}^{(\nu)} \quad \text{with} \quad \nu = \arg\max_i P(\mathbf{x}_\tau^{(i)}|\tilde{\mathbf{x}}_\tau, \tilde{\mathbf{x}}_{\tau-1})\,. \tag{3.7}$$

If the AK0 algorithm is used, the term $P(\mathbf{x}_\tau^{(i)}|\tilde{\mathbf{x}}_\tau, \tilde{\mathbf{x}}_{\tau-1})$ in (3.7) has to be replaced by $P(\mathbf{x}^{(i)}|\tilde{\mathbf{x}})$ from (3.4).
The quality of the soft decision source decoding process can be further enhanced by utilizing parameter individual block codes resulting in redundant index assignments [Gör98], [FHV99], [CAV06]. Usually M bits are used to represent the $Q = 2^M$ quantizer reproduction levels. However, also $M^* > M$ bits can be used to represent the $Q = 2^M$ levels. In this case the index assignment is considered to be redundant and it represents in fact a (possibly non-linear) code. This redundant index assignment can be represented for instance using a (M^*, M) block code with generator matrix $\mathbf{G}$. This block code is used to encode the bit pattern $\mathbf{x}$ into a bit pattern $\mathbf{y}$ with

$$\mathbf{y} = \mathbf{x} \cdot \mathbf{G}\,. \tag{3.8}$$

The only change in the source decoder is the evaluation of (3.1), where the product has to be performed for all M^* bits of the bit pattern.

3.2 Extension of SDSD towards Iterative Source-Channel Decoding (ISCD)

With the discovery of Turbo codes, channel coding close to the Shannon limit became possible with moderate computational complexity. In the past years, the Turbo principle of exchanging *extrinsic* information between separate channel decoders has also been adapted to other receiver components.
To exploit the residual redundancy in source coded parameters such as scale factors or predictor coefficients for speech, audio, and video signals in a Turbo process, *iterative source-channel decoding* (ISCD) has been presented in [AVS01, Gor01] as a means to further improve the quality of *soft decision source decoding* (SDSD).
In order to utilize the soft decision source decoder in an iterative Turbo-like decoding process, extrinsic information for the bits x_κ of the bit sequence $\mathbf{x}$ has to be generated. Instead of giving the complete derivation of the equations, only the results are presented. The complete derivation can be found, e.g., in [AV05], [Gor01], [AVS01]. In order to state the equations, $\mathbf{x}_\kappa^{\text{[ext]}}$ is defined as the bit pattern $\mathbf{x}$ without the considered bit x_κ, i.e.,

$$\mathbf{x}_\kappa^{\text{[ext]}} = \mathbf{x}_{\backslash\kappa} = (x_1, \ldots, x_{\kappa-1}, x_{\kappa+1}, \ldots, x_M)\,. \tag{3.9}$$

If *a priori* knowledge of order 0 (AK0) is utilized, the equations for determining the extrinsic information can be given by a weighted sum over all possible permutations of $\mathbf{x}_\kappa^{\text{[ext]}}$.

$$P^{\text{[ext]}}(x_\kappa = \pm 1) = \sum_{j=1}^{2^{M-1}} P\left(\mathbf{x}_\kappa^{\text{[ext]}(j)}, x_\kappa = \pm 1\right) \cdot P\left(\tilde{\mathbf{x}}|\mathbf{x}_\kappa^{\text{[ext]}(j)}\right)\,. \tag{3.10}$$

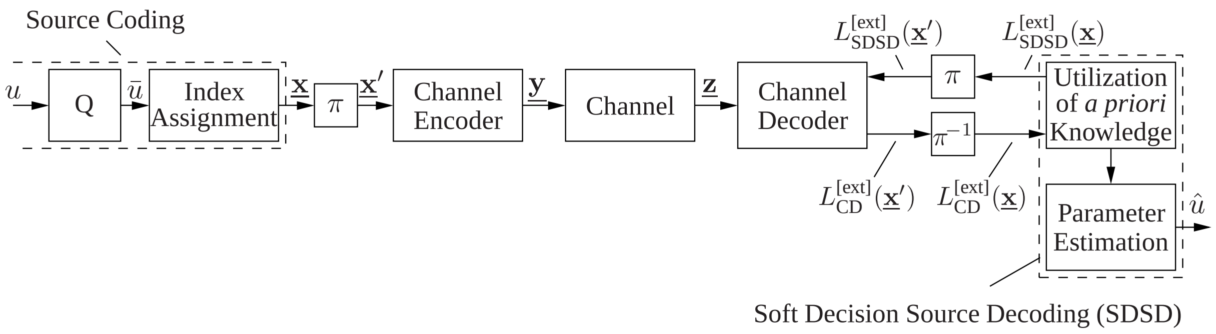

Source Coding
u
Q
$\bar{u}$
Index Assignment
$\underline{\mathbf{x}}$
π
$\underline{\mathbf{x}}'$
Channel Encoder
$\underline{\mathbf{y}}$
Channel
$\underline{\mathbf{z}}$
Channel Decoder
$L^{\text{[ext]}}_{\text{SDSD}}(\underline{\mathbf{x}}')$
$L^{\text{[ext]}}_{\text{SDSD}}(\underline{\mathbf{x}})$
π
π^{-1}
$L^{\text{[ext]}}_{\text{CD}}(\underline{\mathbf{x}}')$
$L^{\text{[ext]}}_{\text{CD}}(\underline{\mathbf{x}})$
Utilization of *a priori* Knowledge
Parameter Estimation
$\hat{u}$
Soft Decision Source Decoding (SDSD)

Figure 3.2: Model of an ISCD system

Frequently, L-values are utilized in iterative receiver schemes, due to better numerical properties. If L-values shall be used at the input and at the output of the SDSD, the extrinsic information is given by

$$L^{\text{[ext]}}_{\text{SDSD}}(x_\kappa) = \ln \frac{\sum\limits_{j=1}^{2^{M-1}} P\left(\mathbf{x}_\kappa^{\text{[ext]}(j)}, x_\kappa = +1\right) \cdot \theta(\mathbf{x}_\kappa^{\text{[ext]}(j)})}{\sum\limits_{j=1}^{2^{M-1}} P\left(\mathbf{x}_\kappa^{\text{[ext]}(j)}, x_\kappa = -1\right) \cdot \theta(\mathbf{x}_\kappa^{\text{[ext]}(j)})} \tag{3.11}$$

with

$$\theta(\mathbf{x}_\kappa^{\text{[ext]}}) = \exp\left(\sum_{\ell=1}^{M-1} \frac{x_{\kappa,\ell}^{\text{[ext]}}}{2} \cdot L^{\text{[input]}}(x_{\kappa,\ell}^{\text{[ext]}})\right) . \tag{3.12}$$

In (3.12), $x_{\kappa,\ell}^{\text{[ext]}}$ denotes the bit at position ℓ in $\mathbf{x}_\kappa^{\text{[ext]}}$. Note that if parameter individual block codes of rate M/M^* are used, M has to be replaced by M^* in the summations in (3.10), (3.11), and (3.12). Iterative source-channel decoding with *a priori* knowledge of order 0 (AK0) is used for most parameters in the FlexCode baseline channel coder as most of the parameters only feature very little correlation over time. For the equations for determining the extrinsic information by considering *a priori* knowledge of order 1 (AK1), we refer the reader to the literature [AV05], [AVS01], [Gor01].
Figure 3.2 depicts a model of a transmission system deploying iterative source-channel decoding at the receiver. The basic concept corresponds to the one depicted in Fig. 3.1. Continuous source samples u are quantized, and after quantization a bit pattern $\mathbf{x}$ is assigned to each sample. In this system, several bit patterns are grouped to a frame of patterns $\underline{\mathbf{x}}$ which are then interleaved by the permutation π to $\underline{\mathbf{x}}'$. After channel encoding and transmission over a channel, the receiver performs iterative source-channel decoding: channel decoder and source decoder iteratively exchange extrinsic information and after a certain number of iterations, the parameters can be estimated using the estimation rule (3.7).

Chapter 4

Iterative Source-Channel Decoding Advancements

The FlexCode baseline channel coder is based on the concept of *iterative source-channel decoding* (ISCD), as described in Chapter 5. In order to adapt the ISCD approach to FlexCode, several changes and optimizations had to be made to the ISCD in order to get a joint source-channel approach suitable to work with the FlexCode source coder. In this chapter, several of these optimizations are presented.
In Section 4.1, the concept of irregular index assignments as well as the application to a hard-output AWGN channel, which can be used for instance to model a packet transmission with bit errors in the packet, are presented. Furthermore, the proposed scheme permits a simple error detection on index basis (error detection is performed independently for each sample) leading to a stopping criterion for the iterative receiver. Such a stopping criterion is indispensable in, e.g., mobile terminals as it considerably reduces the power consumption. In Section 4.2, the concept of soft decision source decoding (SDSD) is extended towards multiple description codes and first steps to apply ISCD to multiple descriptions are indicated.

4.1 Irregular Index Assignments and the Hard-Output AWGN Channel

Most previous publications on ISCD have been focusing on the AWGN channel with perfect soft information available at the receiver. However, in some transmission scenarios it might not be possible to transfer soft information from the physical layer to an upper layer where source-channel decoding may take place. For this reason, we consider a transmission over an AWGN channel with binary quantization of the received values. This channel can be considered as a *binary symmetric channel* (BSC).
However, as the application of known ISCD approaches did not lead to near-optimum decoding, an advancement using irregular index assignments (bit mappings) has been proposed. In this section we present the concept of the irregular index assignments based on the concept of irregular codes. The concept of irregular codes is used as (redundant) index assignment, i.e., the assignment of bit patterns to codebook indices, of a (scalar) quantizer. This extends the concept of redundant index assignments [AVC05], [CAV06], [CVA06]. Additionally, the utilized design guidelines permit to implement a very simple stopping criterion at the receiver, limiting the necessary amount of iterations performed in the case of good channel conditions. Such a stopping criterion is extremely important in mobile applications where the reduction of the power consumption is one of the main optimization objectives. The concept is applied to a hard-output channel, however, the proposed scheme is not restricted to this kind of channel and can be applied to all kinds of channels. It has also been investigated what can be done if no channel state information (CSI) is available at the receiver. Often, no CSI, such as the instantaneous

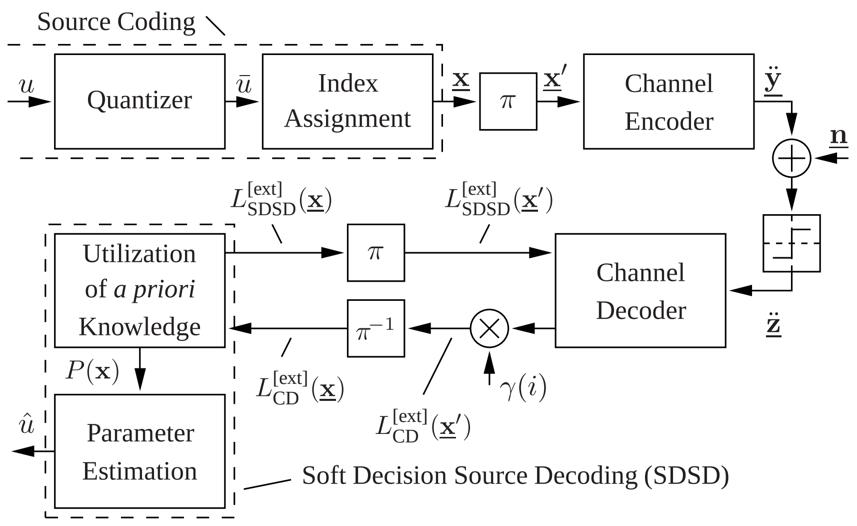

Source Coding
u
Quantizer
$\bar{u}$
Index Assignment
$\underline{\mathbf{x}}$
π
$\underline{\mathbf{x}}'$
Channel Encoder
$\underline{\ddot{\mathbf{y}}}$
$\underline{\mathbf{n}}$
$L^{[\text{ext}]}_{\text{SDSD}}(\underline{\mathbf{x}})$
$L^{[\text{ext}]}_{\text{SDSD}}(\underline{\mathbf{x}}')$
Utilization of *a priori* Knowledge
π
Channel Decoder
π^{-1}
$\underline{\ddot{\mathbf{z}}}$
$P(\mathbf{x})$
$L^{[\text{ext}]}_{\text{CD}}(\underline{\mathbf{x}})$
$\gamma(i)$
$\hat{u}$
Parameter Estimation
$L^{[\text{ext}]}_{\text{CD}}(\underline{\mathbf{x}}')$
Soft Decision Source Decoding (SDSD)

Figure 4.1: Baseband model of the utilized ISCD system (simplified notation, e.g., u instead of $u_{\kappa,\tau}$)

bit error rate or the channel signal-to-noise ratio, is available at the receiver. Without CSI, the *maximum a posteriori* (MAP) algorithm, often employed as channel decoder in ISCD systems, is not able to successfully decode, even in good channel conditions. By applying several measures to compensate for the unknown CSI, we show that a performance can be achieved which is comparable to that of the corresponding system with full CSI. Most of this Section has been published in [SVCS08].

4.1.1 System Model

In what follows, we will give a brief review of the utilized abstract *iterative source-channel decoding* (ISCD) utilized abstract transmission system. In Fig. 4.1 the baseband model of ISCD is depicted. At time instant τ a source encoder generates a frame $\underline{u}_\tau = (u_{1,\tau}, \dots u_{K_S,\tau})$ of K_S unquantized source codec parameters $u_{\kappa,\tau}$, with $\kappa \in \{1, \dots, K_S\}$ denoting the position in the frame. The single elements $u_{\kappa,\tau}$ of $\underline{u}_\tau$ are assumed to be statistically independent. Each value $u_{\kappa,\tau}$ is individually mapped to a quantizer reproduction level $\bar{u}_{\kappa,\tau}$, with $\bar{u}_{\kappa,\tau} \in \mathbb{U}_\kappa = \{\bar{u}^{(0)}_{\kappa,\tau}, \dots, \bar{u}^{(Q_\kappa-1)}_{\kappa,\tau}\}$. The set $\mathbb{U}$ denotes the quantizer codebook with a total number of $|\mathbb{U}_\kappa| = Q_\kappa$ codebook entries. The number of quantizer levels is assumed to be $Q_\kappa = 2^{M_\kappa}$. A unique bit pattern $\mathbf{x}_{\kappa,\tau}$ of M^*_κ bits (with $M^*_\kappa > M_\kappa$) is assigned to each quantizer level $\bar{u}_{\kappa,\tau}$ selected at time instant τ according to the index assignment

$$\begin{aligned} \Gamma_\kappa : \quad & \mathbb{U}_\kappa \rightarrow \mathbb{F}_2^{M^*_\kappa} \\ & \bar{u}_{\kappa,\tau} \mapsto \mathbf{x}_{\kappa,\tau} \end{aligned}$$

with $\mathbb{F}_2 = \{0, 1\}$. In the following, we assume that all codec parameters are quantized using the same codebook, i.e., $\mathbb{U}_\kappa = \mathbb{U}$ and $M_\kappa = M, \forall \kappa \in \{1, \dots, K_S\}$. Although the number of quantization levels is assumed to be identical for all parameters, the index assignment can differ from parameter to parameter. For notational convenience we omit the time index τ in the following.

The single bits of a bit pattern $\mathbf{x}_\kappa$ are indicated by $x^{(m)}_\kappa$, $m \in \{1, \dots, M^*_\kappa\}$. If $M^*_\kappa > M = M_\kappa$, the index assignment Γ_κ introduces redundancy and can then be considered to be the composite function $\Gamma_\kappa = \zeta_\kappa \circ \breve{\Gamma}_{\text{NB}}$ (i.e., $\Gamma_\kappa(\bar{u}) = (\zeta_\kappa \circ \breve{\Gamma}_{\text{NB}})(\bar{u}) = \zeta_\kappa(\breve{\Gamma}_{\text{NB}}(\bar{u}))$) with

$$\begin{aligned} \breve{\Gamma}_{\text{NB}} : \mathbb{U} &\rightarrow \mathbb{F}_2^M \\ \bar{u} &\mapsto \breve{\mathbf{x}} \end{aligned} \qquad \text{and} \qquad \begin{aligned} \zeta_\kappa : \mathbb{F}_2^M &\rightarrow \mathbb{F}_2^{M^*_\kappa} \\ \breve{\mathbf{x}} &\mapsto \mathbf{x}_\kappa \,. \end{aligned}$$

The function $\breve{\Gamma}_{\text{NB}}$ performs a *natural binary* index assignment, i.e., the binary representation of the codebook index of $\bar{u}$ is assigned to $\breve{\mathbf{x}}$. The function ζ_κ can be regarded as being a (potentially non-linear) block code of rate $r_\kappa^{\mathsf{IA}} = M/M_\kappa^*$. The concept of non-linear block codes employed as redundant index assignments has been successfully utilized in, e.g., [HV05], [CVA06]. In this Section however, we only consider linear block codes and refer to Section 4.1.2 for a detailed description. After the index assignment, K_S bit patterns are grouped to a frame of bit patterns $\underline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_{K_S})$ consisting of $\sum_{\kappa'=1}^{K_S} M_{\kappa'}^* = K_S \cdot \bar{M}^*$ bits. The overall rate of the index assignment is thus

$$r^{\mathsf{IA}} = \frac{K_S \cdot M}{\sum\limits_{\kappa'=1}^{K_S} M_{\kappa'}^*} = \frac{M}{\bar{M}^*}\,, \tag{4.1}$$

with $\bar{M}^*$ the average number of bits per parameter. The frame $\underline{\mathbf{x}}$ of bits is re-arranged by a bit interleaver π in a deterministic, pseudo-random like manner. The interleaved frame with $K_S \cdot \bar{M}^*$ bits is denoted as $\underline{\mathbf{x}}'$.

For channel encoding of a frame $\underline{\mathbf{x}}'$, we use a recursive convolutional code of constraint length $J+1$ and of rate r^{C}. In this Section, we restrict ourselves to rate $r^{\mathsf{C}} = 1$ recursive, non-systematic convolutional codes. The encoded frame is denoted by $\underline{\mathbf{y}}$. The bits y_k of $\underline{\mathbf{y}}$ are indexed by $k \in \{1, \ldots, K_S \cdot \bar{M}^* + J\}$. Prior to transmission over the channel, the encoded bits y_k are mapped to bipolar bits $\ddot{y}_k$ forming a sequence $\underline{\ddot{\mathbf{y}}} \in \{\pm 1\}^{K_S \cdot \bar{M}^* + J}$. We only consider BPSK modulation in this Section in order to demonstrate the concept, which can easily be extended to include higher order modulation schemes [CBAV05] or channel equalization [SCV07]. Note that in Fig. 4.1 the baseband model is considered.

The Hard-Output Channel

On the channel, the modulation symbols $\ddot{y}_k$ (with symbol energy $E_{\text{s}} = 1$) are subject to additive white Gaussian noise (AWGN) with known power spectral density $\sigma_n^2 = N_0/2$. After transmission, a hard-decision is performed on the received symbols z_k, i.e., $\ddot{z}_k = \text{sign}\{z_k\}$. This implies that the channel can be modelled as a *binary symmetric channel* (BSC) with bit error probability

$$P_b = P\left(\ddot{z}_k \neq \ddot{y}_k\right) = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_{\text{s}}}{N_0}}\right)\,, \tag{4.2}$$

with erfc denoting the complementary error function.

The capacity of the Hard-Output, Binary-Input AWGN (HO-BIAWGN) channel can be determined using the capacity of a BSC channel [CT06]

$$\mathcal{C}^{[\text{BSC}]} = h(P_b) = P_b \log_2 P_b + (1-P_b)\log_2(1-P_b) \tag{4.3}$$

by replacing P_b by the expression of (4.2). This leads to

$$\mathcal{C}^{[\text{HO-BIAWGN}]} = \frac{1}{2}\left(\text{erfc}\left(\lambda\right)\log_2 \text{erfc}\left(\lambda\right) + \text{erfc}\left(-\lambda\right)\log_2 \text{erfc}\left(-\lambda\right)\right) \tag{4.4}$$

with

$$\lambda = \sqrt{\frac{E_{\text{s}}}{N_0}} = \left(2\sigma_n^2\right)^{-\frac{1}{2}}\,. \tag{4.5}$$

The channel capacity of the AWGN channel with BPSK mapping (with soft output) is given by [Ung82]

$$\mathcal{C}^{[\text{BIAWGN}]} = 1 - \frac{1}{2}\sum_{j=0}^{1} \text{E}\left\{\log_2 \sum_{i=0}^{1} \exp\left(-\frac{|\ddot{y}_j + n - \ddot{y}_i|^2 - |n|^2}{2\sigma_n^2}\right)\right\} \tag{4.6}$$

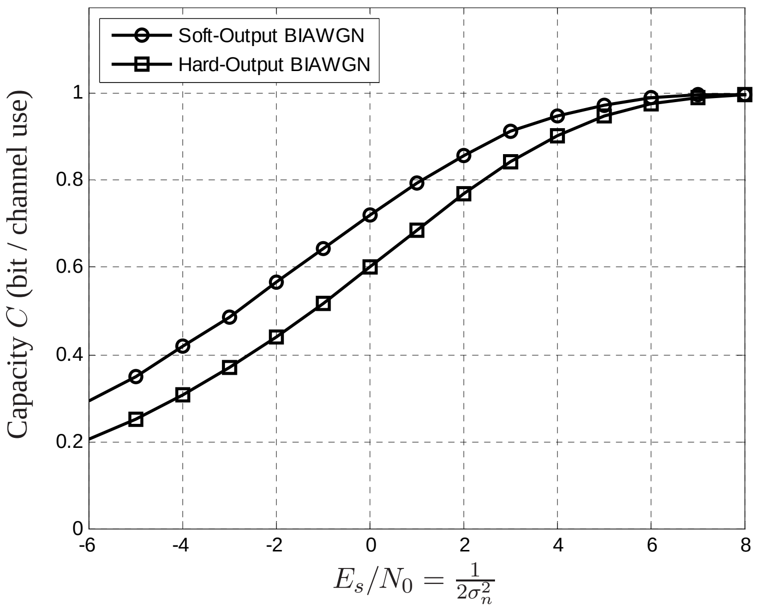


Figure 4.2: Channel capacities of the AWGN channel and the hard-output AWGN channel

with n Gaussian distributed noise samples with

$$n \sim \mathcal{N}\left(0, \frac{1}{2\frac{E_\mathrm{s}}{N_0}}\right) = \mathcal{N}\left(0, \sigma_n^2\right) .$$

Both capacity curves ($\mathcal{C}^{[\mathrm{BSC}]}$ and $\mathcal{C}^{[\mathrm{BIAWGN}]}$) are depicted in Fig. 4.2. It can be seen that performing a hard decision at the channel output causes a loss of approximately $2\,\mathrm{dB}$ at a channel coding rate of $1/2$. This corresponds to the well-known result in channel coding, that exploiting soft-information at a channel decoder leads to a gain of approximately $2\,\mathrm{dB}$ [Hag94].

The Receiver

The received symbols $\ddot{z}_k \in \{\pm 1\}$ are transformed to L-values [HOP96] prior to being evaluated in a Turbo process which exchanges *extrinsic* reliabilities between channel decoder (CD) and soft decision source decoder (SDSD). If *channel state information* (CSI) is available at the receiver, the L-values of the received symbols are obtained by [HOP96]

$$L(\ddot{z}_k) = \log_\mathrm{e}\left(\frac{1-P_b}{P_b}\right) \cdot \ddot{z}_k =: L_c \cdot \ddot{z}_k \tag{4.7}$$

and if no CSI (i.e., $\frac{E_\mathrm{s}}{N_0}$ or P_b) is available at the receiver, the L-values are given by

$$L(\ddot{z}_k) = \check{L}_c \cdot \ddot{z}_k \,, \tag{4.8}$$

with $\check{L}_c$ being a receiver parameter. The adjustment of $\check{L}_c$ will be explained in Section 4.1.5. After channel decoding, the L-values at the decoder output can optionally be scaled by a factor $\gamma(i)$, where i denotes the iteration counter, i.e., $\gamma(i)$ is constant during one iteration.
The channel decoder used in this Section is based on the LogMAP algorithm [BCJR74], [HOP96] or on the MaxLogMAP approximation [RVH95]. For the equations for computing the *extrinsic* probabilities

or their respective L-values of the SDSD, the reader is referred to the literature, e.g., [Gor01], [FV01], [AV05]. Note that the redundancy of the index assignment, introduced by the function ζ_κ, is not explicitly decoded at the receiver but implicitly used to calculate better estimates of the codebook indices given the input L-values.

4.1.2 Irregular Index Assignments

It is known that the inner channel code of a capacity-achieving serially concatenated system should be of rate $r \geq 1$ [AKtB04] and recursive [KHC06]. If this inner channel code is fixed, the outer code can be matched quite well to the inner code using the principles of irregular codes [Tüc04], [TH02]. Irregular codes allow a simple optimization of the outer component by making use of EXIT charts [tB01a].
For a given channel code, the goal is to find a perfectly matching outer component (source code in our case) to the given rate-1 channel code. This task can be solved for example by the concept of irregular codes [TH02], [Tüc04]. Irregular codes, originally proposed for convolutional codes, use several component codes of different rates in one block (e.g., by changing the puncturing rule) to obtain an overall rate-r^{Outer} outer code. As the EXIT characteristic of the resulting code corresponds to the weighted sum of the component codes' characteristics (where the weights correspond to the fractions of code bits being encoded by the respective component code), an optimization algorithm can be formulated [TH02]. This algorithm allows to optimize the weights in order to get an (almost) perfectly matching characteristic.
We extend the concept of irregular codes to the index assignment in order to obtain *irregular index assignments* (IIA). As stated in Section 4.1.1, the index assignment for the parameter u_κ comprises a block code ζ_κ of rate $r_\kappa^{\text{IA}} = M/M_\kappa^*$. Instead of using the same amount of bit redundancy $M_\kappa^* = \bar{M}^*$ for each parameter in order to achieve an overall rate $M/\bar{M}^*$ outer encoding, we use the concept of irregular codes and vary M_κ^* for each parameter. This allows us to use the optimization algorithm in [TH02] to optimize the index assignments and to get an SDSD EXIT characteristic which matches the channel decoder characteristic considerably well.
In the following, we present a simple design guideline in order to generate redundant index assignments with rates $r_\kappa^{\text{IA}} = M/M_\kappa^*$, $M_\kappa^* \in \{M+1, \ldots, M_{\max}^*\}$ needed for the optimization of the IIA. The guideline starts with an (almost) arbitrary generator matrix $\mathbf{G} = (g_{i,j})_{M \times M_{\max}^*}$ of size $\dim \mathbf{G} = M \times M_{\max}^*$ and with elements $g_{i,j} \in \mathbb{F}_2$. A generator matrix $\mathbf{G}_{M^*}$ for a rate $r_\kappa^{\text{IA}} = M/M_\kappa^*$ index assignment is then obtained by

$$\mathbf{G}_{M^*} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{I}_{M^\star} \\ \mathbf{0} \end{pmatrix} \tag{4.9}$$

with $\mathbf{I}_{M^*}$ denoting the $M^* \times M^*$ identity matrix and $\mathbf{0}$ the $(M_{\max}^* - M^*) \times M^*$ all-zero matrix. The only conditions we fix for $\mathbf{G}$ are:

a) $\mathbf{G}$ is a generator matrix for a systematic linear block code, i.e., $\mathbf{G}$ can be written as

$$\mathbf{G} = \begin{pmatrix} \mathbf{I}_M & \mathbf{P} \end{pmatrix}. \tag{4.10}$$

b) the block code generated by $\mathbf{G}_{M+1}$ has a minimum Hamming distance $d_{\text{Ham}}(\mathbf{G}_{M+1}) \geq 2$.

The second condition is necessary for the EXIT characteristic to reach the $(1,1)$ point [CVA06] and is accomplished if $\mathbf{G}_{M+1}$ realizes a parity check code, i.e., $\mathbf{G}_{M+1} = (\mathbf{I}_M \;\; \mathbf{1})$.
We illustrate the generation of irregular index assignments by means of an example. $K_S = 250$ source parameters modelled by K_S independent 1st order Gauss-Markov processes with auto-correlation $\rho = 0.9$ are quantized using a $Q = |\mathbb{U}| = 16$ level Lloyd-Max codebook, i.e., $M_\kappa = M = 4$. Furthermore, we assume that the overall coding rate of the index assignment shall be of rate $r^{\text{IA}} = \frac{1}{2}$, which

gives an average number of $\bar{M}^* = 8$ bits per source parameter. The channel code is a memory $J = 3$, rate-1 recursive convolutional code with generator polynomials $G_{\text{CC}}(D) = \left(\frac{1}{1+D+D^2+D^3}\right)_8$. An exemplary generator matrix $\mathbf{G}$ for $M = 4$ and $M^*_{\text{max}} = 15$, fulfilling conditions a) and b) and generating redundant index assignments with rates $4/5, 4/6, \ldots, 4/15$ could be

$$\mathbf{G} = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,0\,0\,0 \\ 0\,1\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1\,0\,1 \\ 0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\,1 \end{pmatrix} . \tag{4.11}$$

The index assignments generated by using the generator matrix of (4.11) with (4.9) for realizing the Block Code ζ_κ are denoted by $\text{BC}^{Q}_{M^*}$. They are summarized in Table 4.1 in octal notation with the left-most non-zero bit corresponding to $x^{(1)}$.

Example: The block code index assignment BC^{16}_6 is given by $\text{BC}^{16}_6 = \{\mathbf{x}|\mathbf{x} = \Gamma(\bar{u}), \bar{u} = \bar{u}^{(0)}, \ldots, \bar{u}^{(Q-1)}\} = \{0, 6, 13, 15, 23, 25, 30, 36, 43, 45, 50, 56, 60, 66, 73, 75\}$ in octal representation with the least significant bit corresponding to $x^{(M^*)}$. For instance, to the quantizer reproduction level $\bar{u}^{(5)}$, the natural binary representation $\breve{\mathbf{x}} = (0101)_2$ is assigned, leading to

$$\mathbf{x} = \breve{\mathbf{x}} \cdot \mathbf{G}_6 = (0101)\begin{pmatrix} 1\,0\,0\,0\,1\,1 \\ 0\,1\,0\,0\,1\,1 \\ 0\,0\,1\,0\,1\,1 \\ 0\,0\,0\,1\,1\,0 \end{pmatrix} = (010101)_2 = (25)_8.$$

For an overall rate-$\frac{1}{2}$ transmission with the given parameters, it can be observed that a minimum channel quality of $E_\text{s}/N_0 \approx -2.83\,\text{dB}$ is necessary to reach a reconstruction SNR of the decoded parameters of $\approx 20\,\text{dB}$. This channel quality is obtained by calculating the *optimum performance theoretically attainable* (OPTA) [CSVA06], using the capacity of the Hard-Output, Binary-Input AWGN (HO-BIAWGN) channel derived in Section 4.1.1. See Fig. 4.4 for an illustration of the OPTA limit in this case. We perform the optimization, however, using a slightly higher channel quality of $E_\text{s}/N_0 = -2.6\,\text{dB}$.

The EXIT characteristics of the channel decoder $\mathcal{T}^{\mathsf{CD}}$ and the characteristics of the different index assignments are illustrated in Fig. 4.3. It can be seen that the trajectory of the index assignment BC^{16}_8, meeting the rate requirements, has an intersection with the channel decoder characteristic, resulting in a decoder failure. The optimization of the irregular index assignment leads to the characteristic $\mathcal{T}^{\mathsf{Irr}}$, matching considerably well the channel decoder characteristic with an open decoding tunnel.

The results of the optimization are summarized in Table 4.2. The optimization determines the weights α_ℓ. The outcome of the algorithm is that not all index assignments have to be used in order to generate a good matching irregular index assignment but only five of them. The α_ℓ are the weighting factors of the EXIT characteristics and they also determine the fraction of bits to be assigned to each index assignment. From these fractions $\alpha_\ell K_S \bar{M}^*$ the corresponding $K_{S,\ell}$ (number of source parameters assigned to each index assignment) can be calculated by

$$K_{S,\ell} = \text{rnd}\left[\alpha_\ell K_S \bar{M}^* \frac{r^{\mathsf{IA}}_\ell}{M}\right] = \text{rnd}\left[\alpha_\ell K_S \frac{r^{\mathsf{IA}}_\ell}{r^{\mathsf{IA}}}\right], \tag{4.12}$$

with rnd being an appropriate rounding operation such that $\sum_{\forall i} K_{S,\ell} = K_S$. Note that the concept of irregular index assignments introduces no noteworthy additional computational complexity at the receiver, which mainly depends on the number of quantization levels per parameter (which has been fixed to $Q = 2^M$ in this contribution).

The decoding trajectory using the IIA system is also depicted in Fig. 4.3. It can be seen that during the first iterations, the trajectory overshoots the source decoder characteristic. This behavior has been analyzed and described in [ACBV06]. During the last iterations, the trajectory overshoots the characteristic

Table 4.1: Index assignments Γ generated by the generator matrix given in (4.11)

Γ, $(\cdot)_{M^\star}^{Q}$	$\{\mathbf{x}\}_8 = \Gamma(\bar{u})$, $\bar{u} \in \{\bar{u}^{(1)}, \ldots, \bar{u}^{(Q)}\}$
BC_5^{16}	0,3,5,6,11,12,14,17,21,22,24,27,30,33,35,36
BC_6^{16}	0,6,13,15,23,25,30,36,43,45,50,56,60,66,73,75
BC_7^{16}	0,15,26,33,47,52,61,74,107,112,121,134,140,155
	166,173
BC_8^{16}	0,33,55,66,116,125,143,170,217,224,242,271,301
	332,354,367
BC_9^{16}	0,67,133,154,235,252,306,361,436,451,505,562,603
	664,730,757
BC_{10}^{16}	0,157,266,331,472,525,614,743,1075,1122,1213
	1344,1407,1550,1661,1736
BC_{11}^{16}	0,336,555,663,1164,1252,1431,1707,2173,2245,2426
	2710,3017,3321,3542,3674
BC_{12}^{16}	0,674,1332,1546,2351,2525,3063,3617,4367,4513
	5055,5621,6036,6642,7304,7570
BC_{13}^{16}	0,1570,2665,3315,4723,5253,6146,7436,10756
	11226,12133,13443,14075,15505,16610,17360
BC_{14}^{16}	0,3361,5552,6633,11647,12526,14315,17074,21734
	22455,24266,27107,30173,33212,35421,36740
BC_{15}^{16}	0,6743,13325,15466,23516,25255,30633,36170
	43670,4513,50555,56216,60366,66425,73043,75700

of the channel decoder. This behavior has already been observed in [TH02] and is due to the relatively small bit interleaver of size 2000 bits.

4.1.3 Stopping Criterion

The generation of the (irregular) index assignments using a generator matrix as presented in Section 4.1.2 enables the receiver to apply a simple yet effective stopping criterion. In good channel conditions, it is generally not necessary to perform more than only a few iterations. We use a well-known concept from *low-density parity-check* (LDPC) decoding [Gal63], [CF02] and evaluate the parity check equations of the index assignment after each iteration. If all equations are fulfilled, the iterative process can be aborted. If the generator matrix $\mathbf{G}$ is systematic according to (4.10), the parity check matrix $\mathbf{H}_{M^*}$ for the index assignment generated by $\mathbf{G}_{M^*}$ can easily be determined by

$$\mathbf{H}_{M^*} = \begin{pmatrix} \mathbf{P} \cdot \begin{pmatrix} \mathbf{I}_{M^*-M} \\ \mathbf{0} \end{pmatrix} \\ \mathbf{I}_{M^*-M} \end{pmatrix}^T \tag{4.13}$$

with $\mathbf{0}$ denoting the $(M^*_{\text{max}} - M^*) \times (M^* - M)$ all-zero matrix.
Hence, a total number of $M^*_\kappa - M$ parity check equations can be evaluated for each parameter u_κ. The parity checks are performed based on the hard decisions of the extrinsic L-values $L^{[\text{ext}]}_{\text{SDSD}}(\underline{\mathbf{x}})$ at the output of the source decoder. If all parity checks of all parameters in one block $\underline{u}$ are fulfilled, the iterations can be stopped and the parameters $\hat{u}_\kappa$ can be estimated, e.g., using an MMSE estimator [FV01].

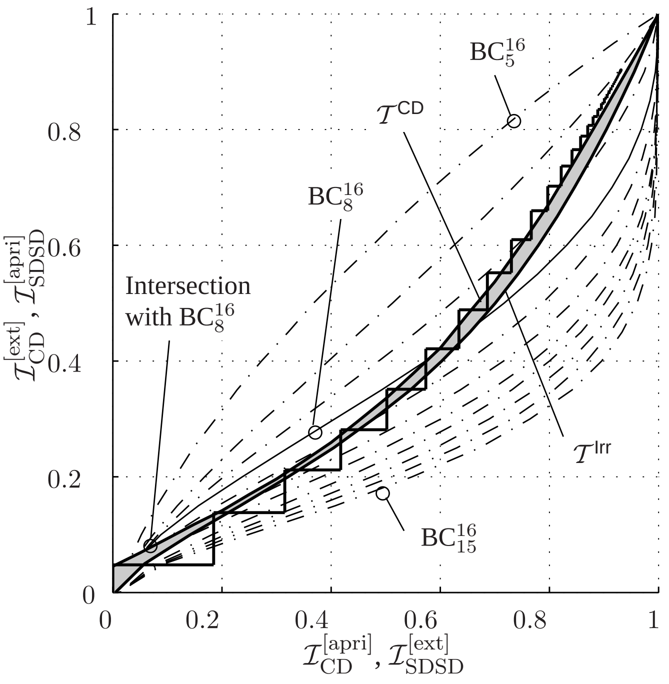


Figure 4.3: EXIT chart analysis of the irregular index assignments at $E_s/N_0 = -2.6\,\mathrm{dB}$ ($E_u/N_0 = 6.43\,\mathrm{dB}$)

4.1.4 Suboptimal Decoding Without Channel State Information

In certain circumstances, no channel state information is available at the receiver. Then, MAP (or LogMAP) decoding fails and the suboptimal MaxLogMAP algorithm becomes an alternative [RVH95]. However, the application of the suboptimal MaxLogMAP algorithm in the ISCD framework leads to performance losses of $\approx 0.7\,\mathrm{dB}$. The reason for these losses is that the MaxLogMAP decoder overestimates the extrinsic information at its output. A simple yet effective remedy against this overestimation is the “normalized MaxLogMAP” algorithm described for instance in [CF02]. After iteration i, the extrinsic output of the MaxLogMAP decoder is multiplied by an (iteration dependent) constant $\gamma(i)$ as indicated in Fig. 4.1. The additional computational complexity introduced by this multiplication is negligible compared to the overall complexity of the MaxLogMAP decoder and the SDSD. The $\gamma(i)$ are determined once in advance by measurements as described in [CF02]: At a channel quality where the performance of LogMAP and MaxLogMAP decoding differ the most, the factor $\gamma(i)$ is obtained using

$$\gamma(i) = \frac{\mathrm{E}\left\{L_{\text{CD,LogMAP}}^{[\text{ext}]}\right\}}{\mathrm{E}\left\{L_{\text{CD,MaxLogMAP}}^{[\text{ext}]}\right\}} \tag{4.14}$$

where $\mathrm{E}\{\cdot\}$ denotes expectation. We only evaluate (4.14) for the cases where $\mathrm{sign}\left\{L_{\text{CD,LogMAP}}^{[\text{ext}]}\right\} = \mathrm{sign}\left\{L_{\text{CD,MaxLogMAP}}^{[\text{ext}]}\right\}$ and $\left|L_{\text{CD,LogMAP}}^{[\text{ext}]}\right| < \left|L_{\text{CD,MaxLogMAP}}^{[\text{ext}]}\right|$. In a first step, the factor $\gamma(1)$ is obtained. Using this factor, the measurement can then be carried out for 2 iterations to obtain $\gamma(2)$ etc. For details, we refer the reader to [CF02].

Table 4.2: Result of the irregular index assignment example

Rate r_ℓ^{IA}	Γ_ℓ	α_ℓ	$\alpha_\ell K_S \bar{M}^*$	$K_{S,\ell}$
$4/15$	BC_{15}^{16}	0.255	510	$K_S^{(4/15)} = 34$
$4/14$	BC_{14}^{16}	0.161	322	$K_S^{(4/14)} = 23$
$4/7$	BC_{7}^{16}	0.189	378	$K_S^{(4/7)} = 54$
$4/6$	BC_{6}^{16}	0.285	570	$K_S^{(4/6)} = 95$
$4/5$	BC_{5}^{16}	0.110	220	$K_S^{(4/5)} = 44$
$r^{\mathsf{IA}} = \frac{1}{2}$		$\sum = 1$	$\sum = K_S \bar{M}^* = 2000$	$\sum = K_S = 250$

4.1.5 Simulation Example

The capabilities of the proposed ISCD system with irregular index assignments scheme are demonstrated by a simulation example. The *parameter signal-to-noise ratio* (SNR) between the originally generated parameters u and the reconstructed estimated parameters $\hat{u}$ is used for quality evaluation. The parameter SNR is plotted for different values of E_u/N_0, with E_u denoting the energy per source parameter u_κ ($E_u = \bar{M}^* \cdot E_{\mathrm{s}}$). Additionally, the bit error probability P_b of the equivalent BSC channel is given on top of Fig. 4.4. Instead of using any specific speech, audio, or video encoder, we use the system setup already introduced in Section 4.1.2 with $K_S = 250$ statistically independent source parameters $\underline{u}$ modelled by K_S independent 1st order Gauss-Markov processes with auto-correlation $\rho = 0.9$. These auto-correlation values can be observed in typical speech and audio codecs, e.g., [Tho07b] for the scale factors in CELP codecs or MP3.

The simulation results are depicted in Fig. 4.4. A system with a non-redundant, natural binary index assignment, a rate $1/2$ convolutional code with memory $J = 3$, hard-decision Viterbi decoding and source decoding by table lookup serves as a reference.

While the system utilizing the regular index assignment BC_8^{16} achieves considerable gains compared to the reference, additional gains of ≈ 0.5 dB can be obtained by using the irregular index assignment from Section 4.1.2. If the *sphere packing bound* (SPB) is used to approximate the behavior for transmissions with a finite block length, the proposed system can reach the OPTA-SPB limit [CSVA06].

The number of utilized iterations is depicted in the lower part of Fig. 4.4. The maximum number of iterations for the system utilizing a regular index assignment is fixed to 20 (as the EXIT chart and simulations show that only up to 20 iterations are beneficial) while the system using the irregular index assignment is allowed to exploit up to 60 iterations due to the narrow decoding tunnel (see EXIT chart in Fig. 4.3). The number of utilized iterations rapidly decreases in the waterfall region, however, the system utilizing *irregular index assignments* (IIA) needs more iterations in the whole range of channel conditions.

Figure 4.5 depicts the evaluation of the inverse normalization factors $1/\gamma(i)$ for the 20 first iterations in both systems. The factors have been determined for the channel quality $E_u/N_0 = 7.3$ dB in the case of the regular index assignment BC_8^{16} and for $E_u/N_0 = 6.9$ dB in the case of the irregular index assignment given in Section 4.1.2. Instead of converging to $\gamma(i) = 1$ for large i as in [CF02], the $\gamma(i)$ converge to a value of about 0.65, i.e., the extrinsic information is continuously overestimated by the MaxLogMAP decoder.

If no CSI is available, the correction factors $\check{L}_c$ in (4.8) are determined by using the channel qualities which have been utilized to determine the normalization factors $\gamma(i)$ (as described above and in Section 4.1.4). First the bit error probability of the channel quality is determined using (4.2), then the

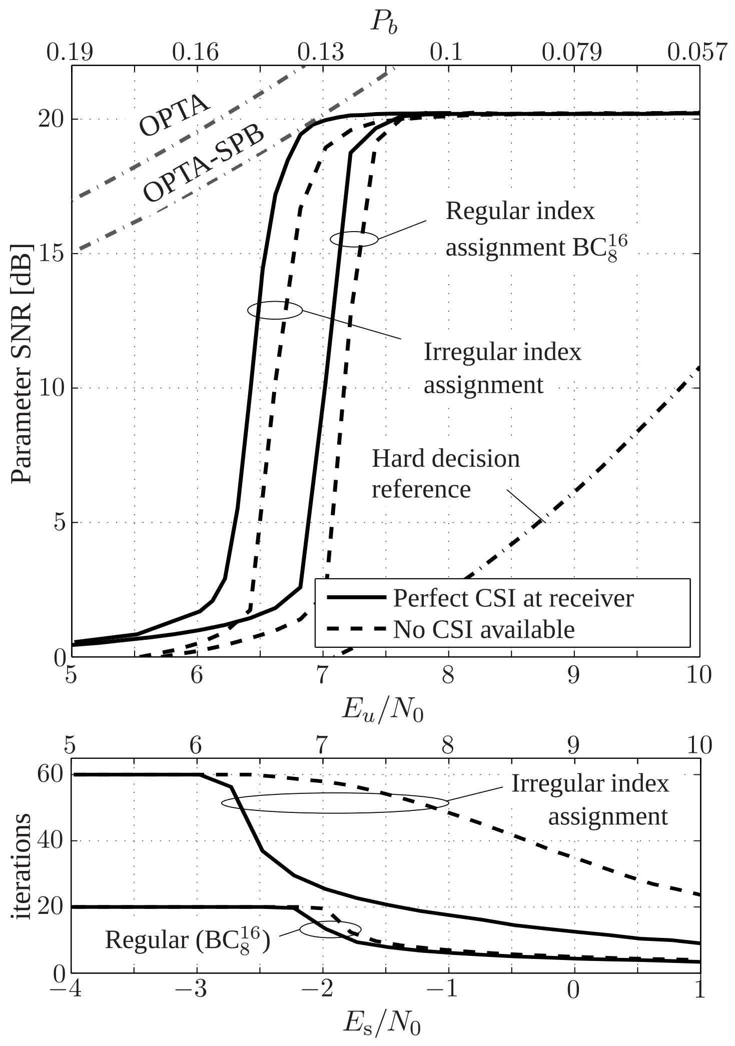


Figure 4.4: Parameter SNR and mean number of iterations for a system with regular and irregular index assignments

correction factor $\check{L}_c$ can be determined using

$$\check{L}_c = \log_\mathrm{e}\left(\frac{1-P_b}{P_b}\right) . \tag{4.15}$$

Therefore, we get $\check{L}_c = 2.13$ for the system utilizing the regular index assignment BC_8^{16} and $\check{L}_c = 2.02$ for the system employing the irregular index assignment. The number of iterations needed if no CSI is available is generally higher than for the system without CSI, especially in the case of the irregular index assignment, as can be seen in Fig. 4.4. This could be explained by the fact that the single iterations only achieve little improvement in terms of mutual information and thus many iterations are needed to iterate through the decoding tunnel.

In this Section, the concept of *irregular index assignments* (IIA) has been presented. Starting with a low-rate systematic generator matrix, the irregular index assignments are generated by multiplying the natural binary representation of the quantizer codebook indices with the first M^* rows of the generator matrix. Using the EXIT chart optimization algorithm known from the technique of irregular codes [TH02], the *iterative source-channel decoding* (ISCD) system can be optimized to yield near optimum performance.

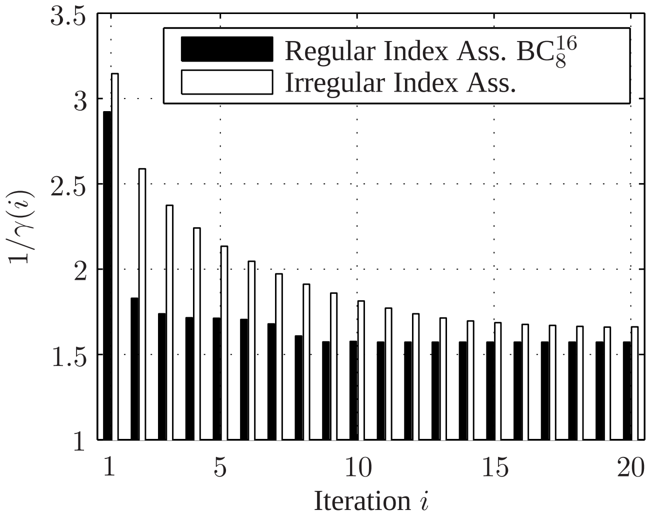


Figure 4.5: Normalization factors $\gamma(i)$ for regular and irregular index assignments

The generation of irregular index assignments using generator matrices yields a simple stopping criterion evaluated at the receiver. The iterative decoder can compute the parity check equations of the different parameters and if all parity check equations are fulfilled, decoding can be stopped. Furthermore, we have shown that in cases where no channel state information is available at the receiver, almost the same performance can be obtained if appropriate measures are taken, at the cost of an increased number of iterations, however.

4.2 Extension towards Multiple Description Coding

In order to ensure maximum flexibility, the FlexCode quantizer might utilize multiple description coding [Goy01], [Vai93]. Multiple description can either be used for error concealment or for a more general kind of hierarchical coding: it is possible to reconstruct the signal if parts of the signal (descriptions) are missing. Missing packets can have two reasons: either due to channel noise only parts of the signal are available at the receiver (as they have been rejected by the error detection mechanism) or due to bottlenecks in the network, parts of the packets have been rejected.

4.2.1 Soft Decision Source Decoding for Multiple Description Coding

Figure 4.6 depicts a transmission system for a multiple description transmission over two channels. The channels are in this case AWGN channel with packet erasures which means that the receiver can reject a complete packet, e.g., because of a receive power level below a certain threshold. The transmitter consists of (scalar) quantizer, followed by a multiple description (MD) indexing[1] [Vai93]. The MD indexing generates two indices $i_{[1]}$ and $i_{[2]}$ which are both mapped to bit patterns, channel encoded independently and transmitted over the two independent channels. The utilized MD indexing is given in Tab. 4.3. The central codebook has a total number of $Q = 21$ entries which are encoded to two descriptions with 8 possible indices each. The MD indexing realizes a one-to-one mapping $i \mapsto (i_{[1]}, i_{[2]})$ with i denoting

[1]The multiple description indexing is frequently called multiple description index assignment in the literature (e.g., [Vai93]). However, in order to avoid confusion with the bit mapping, which is also called index assignment in this report, the term "indexing" has been chosen

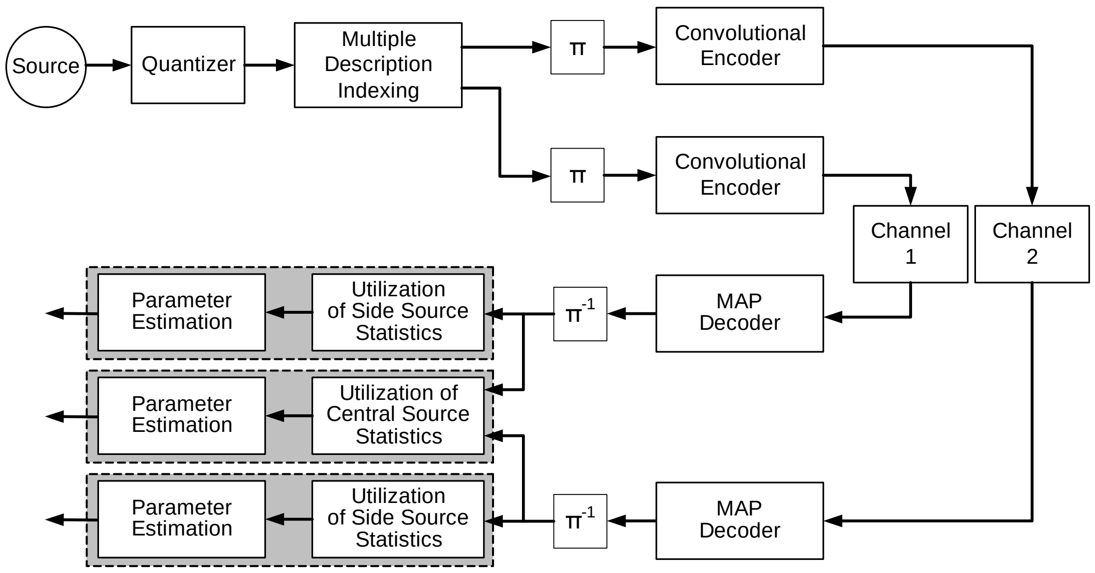


Figure 4.6: Soft decision source decoding for multiple descriptions

Table 4.3: Multiple description indexing utilized for the simulation results in Fig. 4.7

		$i_{[1]}$							
		0	1	2	3	4	5	6	7
	0	0							
	1	1	2	5					
	2		3	4	6				
	3			8	8	10			
$i_{[2]}$	4				9	11	12		
	5					13	14	16	
	6						15	17	18
	7							19	20

indices of the central codebook entries. To each of the indices $i_{[j]}$, $j \in \{1, 2\}$, a bit pattern of 3 bit is assigned. At the receiver, if a description is available (and has not been marked lost), the channel code is decoded by a MAP decoder [BCJR74] followed by soft decision source decoding. If both descriptions are available, the statistics for the central codebook can be utilized. On the other hand, if only one description is received, special optimized side have to be employed [Vai93]. In this case, the statistics of the side codebooks have to be used.

Simulation results (parameter SNR over channel quality E_u/N_0) for soft decision source decoding of multiple descriptions are depicted in Fig. 4.6. Two cases have been considered: both descriptions are received and only one description is received. The application of SDSD without considering *a priori* knowledge leads to a gain, as has been expected. If both descriptions are available, the gain increases to approximately 2 dB. The utilization of *a priori* knowledge of order 0 (AK0, unequal distribution of the source parameters) leads to a further gain. Even higher gains can be expected if *a priori* knowledge of first order is exploited.

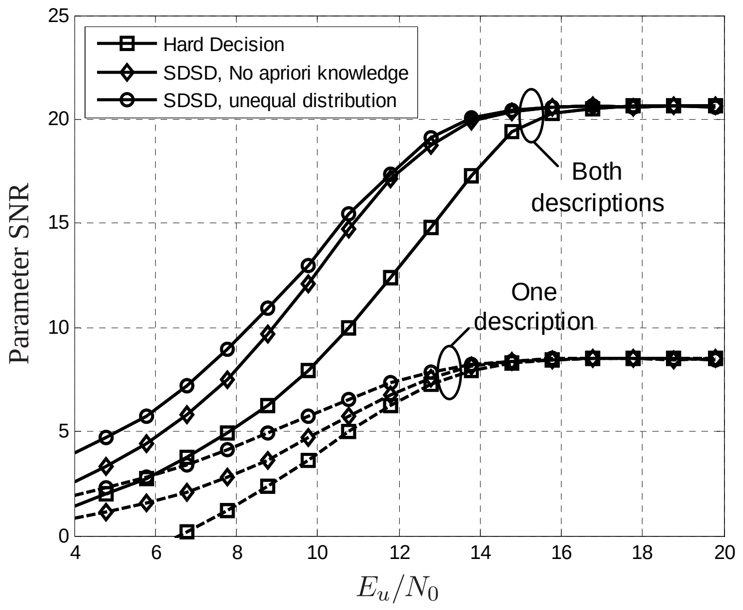


Figure 4.7: Simulation results for soft decision source decoding using multiple descriptions

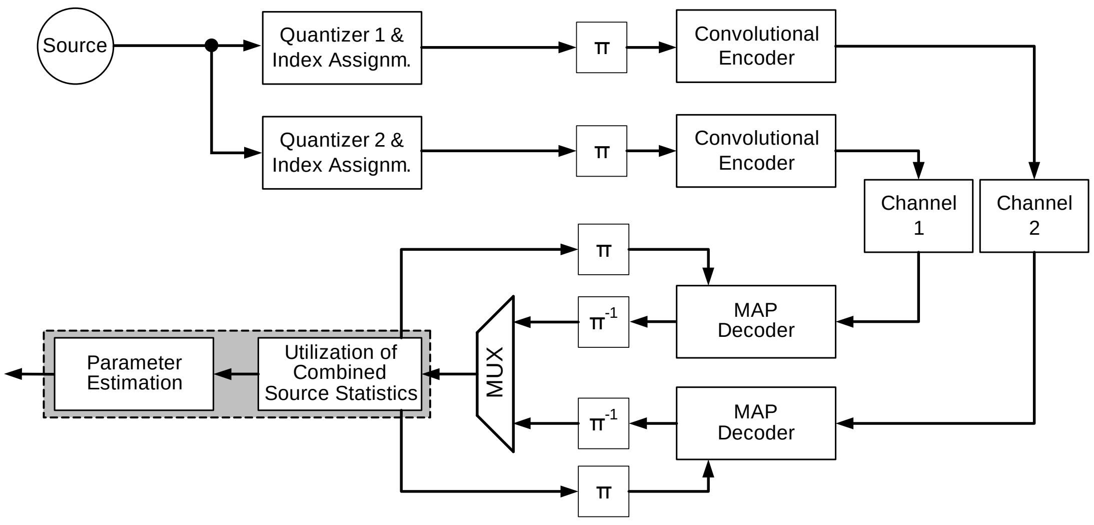


Figure 4.8: Simplified transmission system for multiple descriptions using iterative source-channel decoding

4.2.2 Iterative Source-Channel Decoding for Multiple Descriptions

The next step is the application of the concept of iterative source-channel decoding to multiple description coding. Several work has already been performed in order to exploit the redundancy of the MD indexing in an iterative Turbo-like algorithm [BHG02]. In this Section however, the residual redundancy included in the source samples is exploited by the soft decision source decoder and used to calculate extrinsic information (see Sec. 3.2).

Instead of applying ISCD directly to the multiple description system introduced in Sec. 4.2.1, a simplified system is presented in order to show the principle of how ISCD could be applied to a system utilizing multiple descriptions. This simplified transmission system is depicted in Fig. 4.8. Starting with a Lloyd-Max optimized quantizer codebook $\mathbb{U}_{\mathrm{LM}} = \left\{\bar{u}_{\mathrm{LM}}^{(0)}, \ldots, \bar{u}_{\mathrm{LM}}^{(Q-1)}\right\}$, two sub-codebooks $\mathbb{U}_1$ and $\mathbb{U}_2$ are determined by

$$\mathbb{U}_{[1]} = \left\{\bar{u}_{\mathrm{LM}}^{(j)} \in \mathbb{U}_{\mathrm{LM}} \mid j = 2 \cdot \ell, \forall \ell \in \left\{0, 1, \ldots, \left\lfloor \frac{Q-1}{2} \right\rfloor\right\}\right\} \tag{4.16}$$

$$\mathbb{U}_{[2]} = \left\{\bar{u}_{\mathrm{LM}}^{(j)} \in \mathbb{U}_{\mathrm{LM}} \mid j = 2 \cdot \ell + 1, \forall \ell \in \left\{0, 1, \ldots, \left\lfloor \frac{Q}{2} \right\rfloor - 1\right\}\right\} \tag{4.17}$$

The signal is quantized using both codebooks $\mathbb{U}_{[1]}$ and $\mathbb{U}_{[2]}$ and to each of the resulting indices $i_{[1]}$ and $i_{[2]}$ a bit pattern $\mathbf{x}_{[j]}$, $j \in \{1, 2\}$ is assigned. As an example, the source generates Gaussian distributed samples with zero mean and unit variance. The codebook $\mathbb{U}_{\mathrm{LM}}$ is the 16-level Lloyd-Max codebook which means that both codebooks $\mathbb{U}_{[1]}$ and $\mathbb{U}_{[2]}$ contain 8 entries each, leading to bit patterns $\mathbf{x}_{[j]}$ of size $M_{[j]} = 3$. After separate channel encoding of the descriptions, each description is transmitted over an AWGN channel with frame erasure probability ϵ. After separate channel decoding, a combined soft decision source decoder is utilized to generate the extrinsic information. In the following, the concept behind this combined source decoder will be presented.
Using both codebooks $\mathbb{U}_{[1]}$ and $\mathbb{U}_{[2]}$, a combined codebook $\mathbb{U}_{\text{comb}}$ can be determined by

$$\begin{aligned}\mathbb{U}_{\text{comb}} &= \left\{\bar{u}_{\text{comb}}^{(0)}, \ldots, \bar{u}_{\text{comb}}^{(Q-2)}\right\} \\ &= \left\{\frac{1}{2}\left(\bar{u}_{[1]}^{(0)} + \bar{u}_{[2]}^{(0)}\right), \frac{1}{2}\left(\bar{u}_{[1]}^{(1)} + \bar{u}_{[2]}^{(0)}\right), \frac{1}{2}\left(\bar{u}_{[1]}^{(1)} + \bar{u}_{[2]}^{(1)}\right), \ldots, \frac{1}{2}\left(\bar{u}_{[1]}^{(Q-1)} + \bar{u}_{[2]}^{(Q-1)}\right)\right\}.\end{aligned} \tag{4.18}$$

The entries of the combined codebook correspond to the mean between two neighboring entries of the scalar codebooks $\mathbb{U}_{[1]}$ and $\mathbb{U}_{[2]}$. The total number of entries of $\mathbb{U}_{\text{comb}}$ amounts to $Q-2$, i.e., one entry less than the Lloyd-Max codebook. This means that the reconstruction performance is sub-optimal if the combined codebook is utilized. Using this combined codebook, *a priori* knowledge of order 0 (AK0) an order 1 (AK1) can be determined by exploiting the source properties. To each of these combined codebook entries, a combined bit pattern can be assigned by concatenating the bit patterns of the neighboring entries. This results in the set of bit patterns $\mathbb{X}_{\text{comb}}$ with

$$\begin{aligned}\mathbb{X}_{\text{comb}} &= \left\{\mathbf{x}_{\text{comb}}^{(0)}, \ldots, \mathbf{x}_{\text{comb}}^{(Q-2)}\right\} \\ &= \left\{\left(\mathbf{x}_{[1]}^{(0)}\mathbf{x}_{[2]}^{(0)}\right), \left(\mathbf{x}_{[1]}^{(1)}\mathbf{x}_{[2]}^{(0)}\right), \left(\mathbf{x}_{[1]}^{(1)}\mathbf{x}_{[2]}^{(1)}\right), \ldots, \left(\mathbf{x}_{[1]}^{(Q-1)}\mathbf{x}_{[2]}^{(Q-1)}\right)\right\},\end{aligned} \tag{4.19}$$

with $\left(\mathbf{x}_{[1]}^{(0)}\mathbf{x}_{[2]}^{(0)}\right)$ denoting the concatenation of $\mathbf{x}_{[1]}^{(0)}$ and $\mathbf{x}_{[2]}^{(0)}$. In the above example, the combined bit patterns are of length $M_{\text{comb}} = 6$ (concatenation of two 3-bit patterns). The combined index assignment can be regarded as a redundant index assignment of rate $\frac{\log_2(Q-1)}{M_{\text{comb}}}$ with $M_{\text{comb}} = M_{[1]} + M_{[2]}$. The soft decision source decoder operates identically to the soft decision source decoder introduced in Sec. 3.2 and uses the combined codebook and the redundant index assignment given by concatenation of both separate index assignments. If one description is missing, the corresponding bits of this index assignment are marked as erasure, i.e., the L-value of those is set to zero at the input of the SDSD. Note that this scheme does not feature the full multiple description functionality as the system in Sec. 4.2.1, which can make use of fully custom multiple description indexing.
Figure 4.9 shows simulation results for the proposed system. The plot depicts the parameter SNR for two AWGN channels with identical channel quality E_u/N_0 and a packet erasure probability ϵ ranging from 0 to 15%. The source generates Gaussian distributed samples with zero mean, unit variance and correlation $\rho = 0.9$. The above described setup with an Lloyd-Max codebook consisting of $Q = 16$ entries is

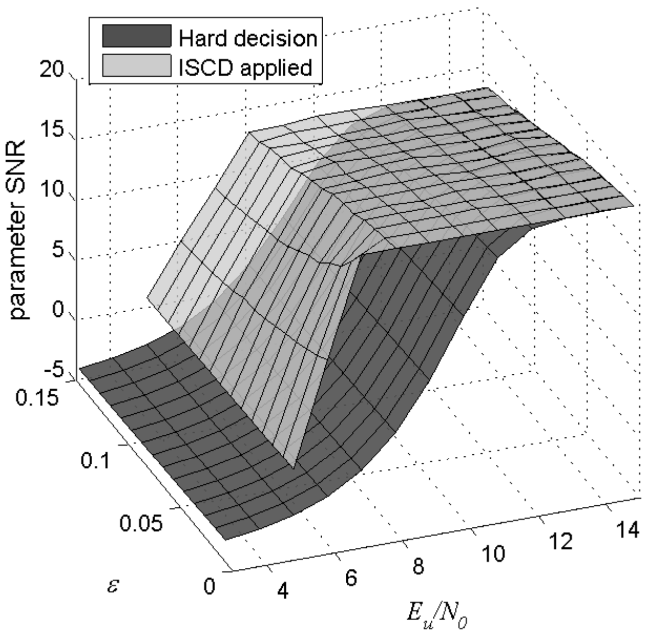
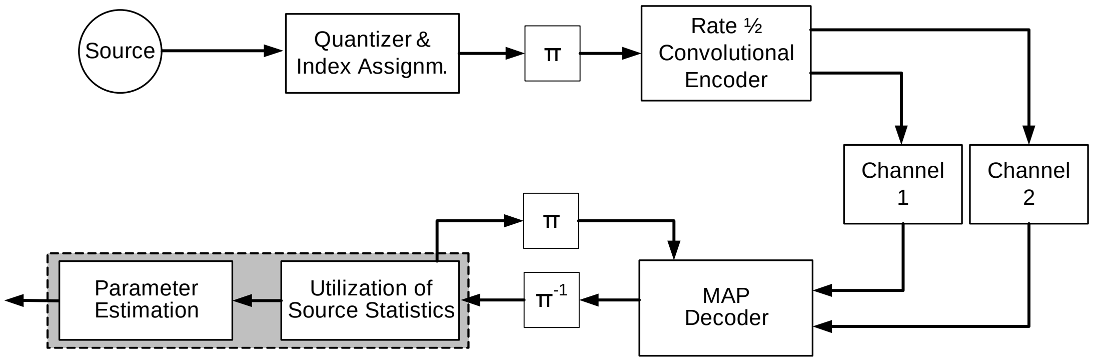


Figure 4.9: Simulation results for the proposed multiple description system with iterative source channel decoding of Fig. 4.8



Source
Quantizer & Index Assignm.
π
Rate ½ Convolutional Encoder
Channel 1
Channel 2
π
MAP Decoder
π-1
Utilization of Source Statistics
Parameter Estimation

Figure 4.10: Alternative multiple description scheme using the channel encoder to generate multiple descriptions

utilized. The convolutional code is a rate-1/2 recursive systematic convolutional code of constraint length $J = 4$ and generator polynomials $G_{\text{CC}}(D) = \left(1, \frac{1+D^2+D^3}{1+D+D^3}\right)$. A frame consists of 250 parameters. The dark surface shows the parameter SNR performance if the receiver is replaced by a conventional hard decision receiver using table lookup and the side codebooks $\mathbb{U}_{[1]}$ and $\mathbb{U}_{[2]}$ if one description is missing. If both descriptions are missing, the receiver outputs the zero sequence. If the iterative receiver (using 10 iterations) is employed, large gains can be observed as expected.

An alternative system for performing ISCD with multiple descriptions is the transmission setup depicted in Fig. 4.10. Instead of generating multiple descriptions immediately after quantization, a conventional scalar (Lloyd-Max) quantizer is used and the channel encoder generates multiple descriptions. For instance, both outputs of a rate-1/2 convolutional code can be treated as descriptions and can be transmitted

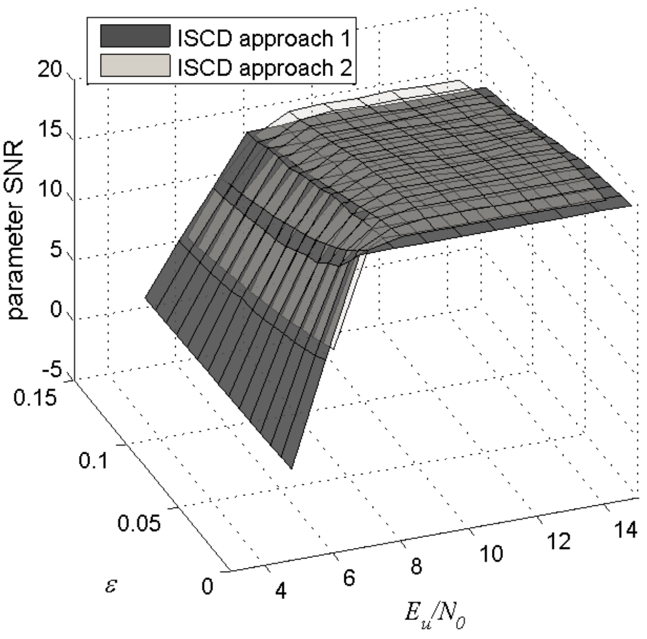
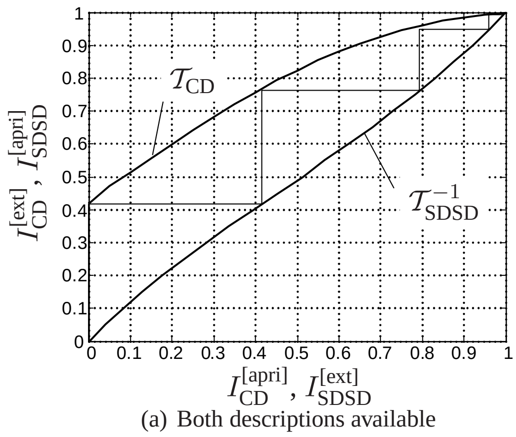


Figure 4.11: Comparison of parameter SNR performance between both systems ("ISCD approach 1" corresponds to the system depicted in Fig. 4.8 and "ISCD approach 2" to the system depicted in Fig. 4.10)

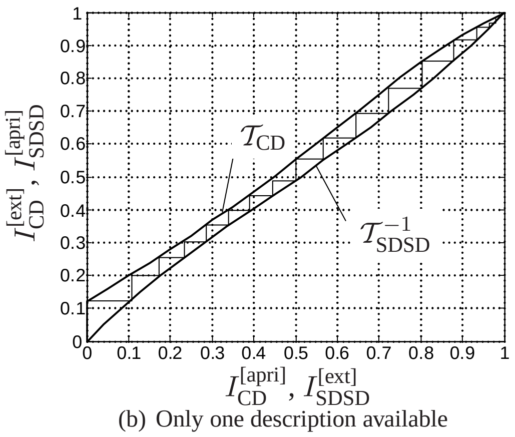
independently over the channels. These can also be interpreted as encoding the quantized bitstream twice by a rate-1 convolutional code and transmitting the outputs of the rate-1 code. If an iterative receiver is deployed, it is well known that rate-1 codes lead to a good system performance [AKtB04], [CVA06]. The receiver consists of a conventional ISCD receiver, as described in Sec. 3.2. If one of the descriptions is lost, the input to the channel encoder is marked as erased, i.e., zeros if L-values are used.

For a simulation example, a Gaussian source with zero mean and unit variance has been used. The utilized quantizer is a 16-level Lloyd-Max quantizer. The index assignment is a redundant index assignment with a $(6, 4)$-block code. The utilized convolutional code is a rate-1/2 recursive non-systematic code with generator polynomials $G(D) = \left(\frac{1}{1+D+D^2+D^3}, \frac{1}{1+D+D^2+D^3}\right)$. Again, 250 parameters are grouped into one frame. Figure 4.11 depicts simulation results of the proposed system (denoted "ISCD approach 2"). The reference system is the first ISCD approach depicted in Fig. 4.8 (denoted "ISCD approach 1"). The waterfall region of both systems occurs at approximately the same channel quality. However, in good channel conditions ($E_u/N_0 > 8$ dB), the proposed multiple description approach using the channel encoder to generate the descriptions shows significant gains in parameter SNR compared to the original approach. This is due to the fact that no side codebooks are utilized. In the first approach, if one description is missing, a suboptimal reconstruction using a smaller codebook is performed. In the new approach, the iterative receiver is still able to exploit the full codebook if only one description has been received.

Figure 4.12 shows the EXIT chart analysis of the proposed multiple description system at a channel quality of $E_u/N_0 = 8.8$ dB. The plot in Fig. 4.12(a) shows the case that both descriptions are available. A wide decoding tunnel is open and after a few iterations convergence (and thus perfect decoding) is reached. On the other hand, the plot in Fig. 4.12(b) shows the EXIT chart when only one description is available. A narrow decoding tunnel is still open, however, the decoder has to perform a larger number of iterations in order to reach convergence. After this larger number of iterations, the same reconstruction



(a) Both descriptions available



(b) Only one description available

Figure 4.12: EXIT chart analysis for the proposed new multiple description approach at $E_u/N_0 = 8.8\,\text{dB}$

quality is reached as for the case when both descriptions were available.

Both proposed iterative source-channel decoding schemes only present a very basic implementation of the multiple description approach and they still lack of the flexibility which is needed for FlexCode. For instance, the first approach does not yet feature full multiple description indexing, but only a fixed indexing (also known as staggered indexing). Furthermore, the utilized codebook is not optimal, but results from a combination of the optimal codebook entries. Furthermore, no optimized side codebooks are exploited in this case. All these issues are currently addressed in the ongoing research work.

Chapter 5

Baseline FlexCode Channel Coder

In this chapter, the baseline channel coder of FlexCode is explained. The baseline channel coder of FlexCode is a joint source-channel coding approach based on the concept *iterative source-channel decoding* (ISCD), which has been introduced in Sec. 3.2.
In traditional system design, the source encoder delivers a bitstream which is then encoded by the channel encoder prior to transmission. In contrast to traditional system design, we place the boundary between source and channel coder on a different level. The FlexCode source encoder outputs indices to the quantized samples which are then transformed into a bitstream inside the channel encoder. This means that indices are passed between source and channel encoder in the FlexCode realization. At the receiver, similarly, the channel decoder reconstructs the bitstream *and* the quantizer indices which are then passed to the source decoder. This separation between source and channel coders has been chosen such as this facilitates the application of joint source-channel (de-)coding.

5.1 Preliminary FlexCode Source Coder Model

In order to analyze the capabilities of a channel coding scheme, it is generally not feasible to use the output of the codec during development as it is difficult to compare channel coders using a given speech file: the proposed scheme might work well for a given speech segment but perform worse for a different speech segment. Furthermore, it is difficult to get reproducible results with a certain encoded speech file. Therefore, a random source that approximates the statistical properties of the preliminary codec output is utilized during the development of channel coding scheme as well as for joint source-channel coding schemes. For instance, most of the joint source-channel coding schemes have been developed and tested using a Gauss-Markov source of first order [AV05], [Gor01], [Tho07b], [GFGR01].
Such a simple source model, however, is not accurate for developing a channel coding scheme for the FlexCode source coder. In order to develop a statistical source model, we utilized a preliminary version of the source coder with 16 kHz sampling rate, constrained entropy quantization, and an average bit rate of ≈ 29.5 kbit/s. Note that for different setups, the parameters of this artificial source have to be adapted. In the following, we will explain the codec based on this setup, however, note that the proposed concept is highly flexible and can easily adapt to different setups.
The output of this codec setup is summarized in Table 5.1, where the parameters contained in one block of the codec are described. Roughly, the output of the codec can be divided into two parts: model and signal. The model part of one frame (of length 20 ms) contains an index to the component of the GMM which is used to quantize the LSFs followed by the quantized LSFs. The signal part is composed of 4 subframes (length 5 ms) which consist of a gain factor (which is in fact also part of the model part) and of the encoded signal samples.

Table 5.1: Parameters contained in one block of the utilized codec setup

Type	Number of elements
GMM Index	1
LSFs	16
Gain	1
Signal	80
Gain	1
Signal	80
Gain	1
Signal	80
Gain	1
Signal	80

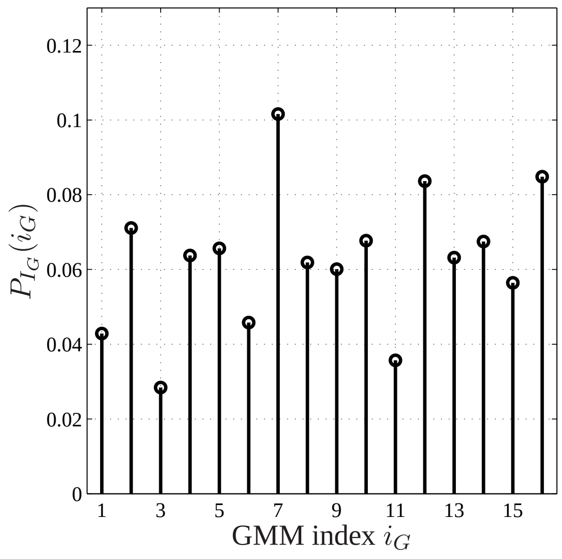


Figure 5.1: Probabilities of occurrence of the different Gaussian mixtures

In the constrained entropy case all the parameters are quantized using a uniform stepsize. The probability distribution of all parameters is known (to a certain extent) such that the probabilities of occurrence of each parameter can be determined. These probabilities of occurrence are used in the entropy coding part of the source encoder. As uniform quantization is used, usually a lot of residual redundancy is contained in the parameters after quantization which is afterwards eliminated using an entropy coder, e.g., arithmetic coding [BCW90], [BCK07].

In the following, the properties and the generation of the different parts of the codec are described in detail:

GMM Index The GMM Index can take one out of 16 values. The probabilities of these values correspond to the weights of the Gaussian Mixture model and are depicted in Fig. 5.1. These probabilities can be considered as *a priori* knowledge. An artificial GMM index can be generated by using this distribution.

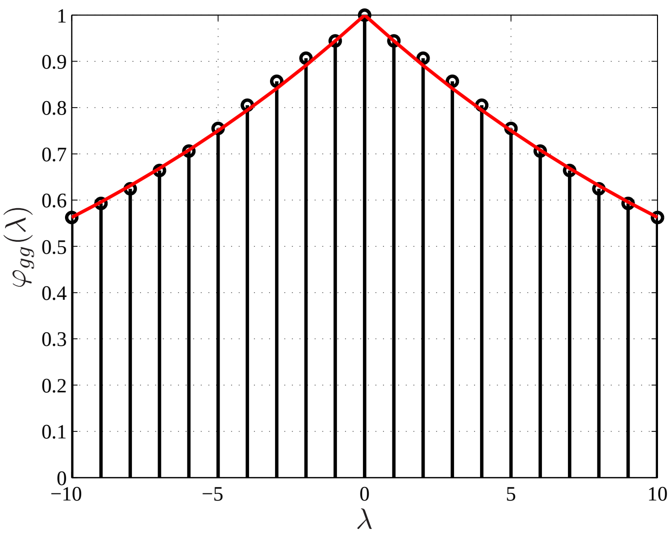


Figure 5.2: Correlation of gains

LPC coefficients The LPC coefficients are converted to *Line Spectral Frequencies* (LSFs) [VM06] prior to transmission. They are quantized using uniform quantization with variable stepsize. These stepsizes are determined by the GMM. Using the GMM index, a set of stepsizes is determined by the source encoder.

Gain A constant stepsize is utilized for the gain. The gain is assumed to be Gaussian distributed (unit variance) and is quantized using a uniform quantization with the constant stepsize. As the signal which is to be encoded usually does not vary significantly in amplitude, the values of the gain are highly correlated. The correlation of the gain factors is shown in Fig. 5.2. The correlation of the gains can be approximated quite well by an AR(1) process with $\rho_g \approx 0.95$. The autocorrelation function of an AR(1) process is described by

$$\varphi_{gg}(\lambda) = \rho_g^{|\lambda|} \,. \tag{5.1}$$

The approximated correlation function is indicated by the red line in Fig. 5.2 and it can be seen that the values quite well match the measured correlation of the gains.

Signal (Transform coefficients) Using the model (LSFs and gain) as well as a ringing part of the previous (sub-)frame, the quantization stepsizes for the whole block are determined by the encoder. The source encoder delivers the indices as already quantized samples. In the constrained entropy case, the signal samples to be quantized are assumed to be Gaussian distributed. The source encoder estimates the variances of samples which are quantized using a uniform quantizer with stepsize 1.0. Using the variance it is then possible to calculate the probabilities of occurrence of the single samples. However, for WP2, we assume that the samples are Gaussian distributed with unit variance and that the stepsize of the quantizer changes. If the source encoder estimates the variance at position k to be σ_k^2, the quantizer stepsize for the unit variance case simply is

$$s_s(k) = \frac{1}{\sigma_k} \,. \tag{5.2}$$

Measurements have shown that the stepsizes show a particular distribution which is depicted in Fig. 5.3(a). The distribution of the stepsizes $p_S(s)$ can be approximated pretty well with the *generalized*

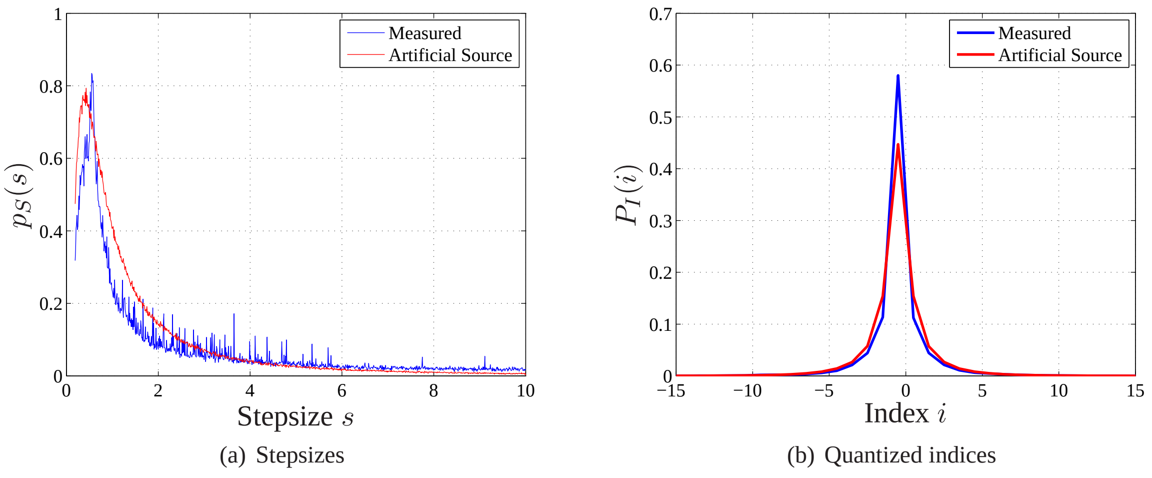


Figure 5.3: Stepsizes and quantized indices of the transform coefficients for the FlexCode artificial source. Comparison of artificial source with measured codec output

extreme value (GEV) distribution

$$p_{\mathrm{GEV}}(\lambda) = \frac{1}{\sigma}\left(1 + K\frac{\lambda-\mu}{\sigma}\right)^{-1-\frac{1}{K}} \exp\left(-\left(1 + K\frac{\lambda-\mu}{\sigma}\right)^{-\frac{1}{K}}\right) \tag{5.3}$$

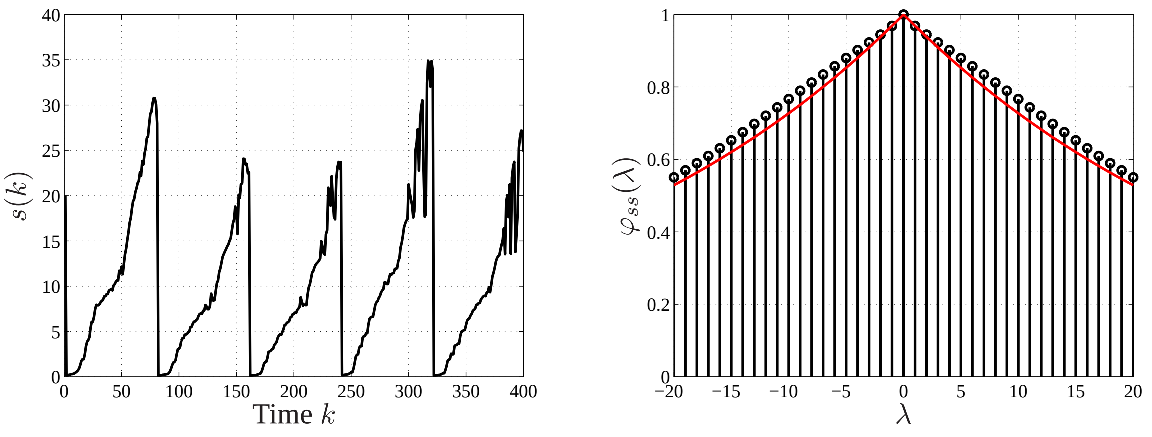
i.e., $p_S(s) \approx p_{\mathrm{GEV}}(s)$. Using a uniform random number generator and, e.g., the rejection method [PTVF92], GEV-distributed samples can be generated very easily. The generation of the quantized indices follows immediately: a Gaussian random source [PTVF92] generates Gaussian distributed samples with zero mean and unit variance which are quantized using a uniform quantizer with stepsize $s_s(k)$, with $s_s(k)$ distributed according to (5.3). The measured probabilities of the quantized indices as well as those originating from the artificial source are depicted in Fig. 5.3(b).

The stepsizes are highly correlated due to the slow changes of a speech or audio signal. An example of the behavior of the transform coefficient stepsize is given in Fig. 5.4(a). The measured correlation is depicted in Fig. 5.4(b). Again, as for the correlation of the gains, the correlation of the transform coefficient stepsizes can be modelled by an AR(1) process with $\rho_s \approx 0.975$. The theoretical correlation of the AR(1) process is indicated by the red line in Fig. 5.4(b).

Modeling the correlation of the these stepsizes is a difficult task however. Generating correlated Gaussian distributed samples is easy, as white Gaussian noise which is filtered by the transfer function of an AR(1) process $H(z)$ with

$$H(z) = \frac{C}{1+\rho z^1} \tag{5.4}$$

shows a Gaussian distribution. However, the stepsizes are not Gaussian distributed (see Fig. 5.3(a)). One way to generate non-Gaussian correlated random numbers is the algorithm described in [CE81] which uses the characteristic function and the cumulant generating function of the wanted distribution to determine the distribution, which, if filtered by $H(z)$, equals the wanted distribution, i.e., the GEV distribution in this case. However, numerical problems avoided the use of this algorithm for modeling the stepsizes and their correlation: the cumulants of the given distribution grow very fast such that the numerical calculation failed. The generation of random numbers possessing the GEV distribution and the given correlation will be part of the second phase of developing the FlexCode channel coder.



(a) Stepsizes over time (b) Correlation of stepsizes

Figure 5.4: Large correlation of the transform coefficients stepsizes. Evaluation over time and correlation

5.1.1 FlexCode Dummy Source Decoder

As the channel decoder and source decoder are interacting heavily at the receiver side (see Sec. 5.3), a dummy decoder modeling the behavior of the real source decoder had to be developed. This dummy decoder models exactly the behavior of the source decoder and knows about the current state of the encoder. During the interaction between channel decoder and source decoder, the channel decoder delivers a block of decoded indices to the dummy source decoder, which decodes them and then compares them to the indices that the source would have produced (which are known, as the internal state of the dummy source decoder corresponds to the internal state of the source encoder). If all indices needed to produce the next set of stepsizes are decoded correctly, the dummy decoder forwards these stepsizes to the channel decoder. If a decoding error has occurred, the dummy decoder is not able to determine the correct stepsizes. This corresponds to a decoding failure and two cases are distinguished:

- The maximum failure case: The stepsizes delivered to the channel decoder emanate from a random process, realizing the GEV distribution (5.3).
- Partly failure case: A Gaussian distributed decoding error is added to the correct stepsizes and the absolute value is taken (in order to be sure that the stepsize is positive).

5.2 Flexible Interleavers

The FlexCode baseline channel coder uses the Turbo principle of exchanging extrinsic information between two component codes (here between channel decoder and soft decision source decoder). An essential element of a transmission or storage system employing the Turbo principle is the interleaver (see Sec. 2.5). As the FlexCode source coder can adapt on the fly to different scenarios and source conditions, the size of the data to be channel coded might be subject to frequent changes. Furthermore, the constrained entropy coder leaves different amounts of redundancy inside the quantized data such that, after possible data compression (if variable length codes (VLCs), such as, e.g., arithmetic codes, are used) the size of the packets to transmit varies significantly. An example of the variation of the blocksize over time for a 5 second speech sample is depicted in Fig. 5.5. The size of the encoded blocks varies between approximately 200 and 1050 bits per block for the given setup. Therefore, interleavers are needed which

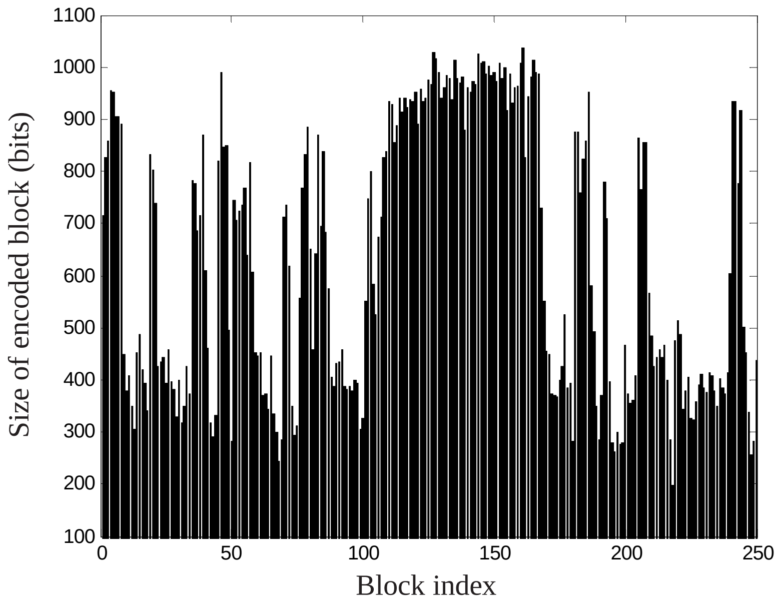


Figure 5.5: Size of an encoded block (20 ms, i.e., 320 samples per block) over time

can change their size on the fly with moderate computational complexity. Such interleavers are called *prunable* interleavers [TDB07], [DB05b], [DB05a], [DB05c].

Several communication systems, such as UMTS, already employ prunable, variable-size interleavers [3GP]. These interleavers are based on the Zech logarithm and utilize Galois Field arithmetic. A comparable prunable interleaver is described in [EH99]. Starting with a conventional block interleaver of size V_{max}, the prunable interleaver is obtained by permuting the columns using a row-dependent function based on the Galois field arithmetic. The different rows are then permuted using the bit flipping algorithm.

The advantage of this design principle is that the implementation is very easy as Galois field computations can be implemented using lookup tables and that the pruning is very easy. The disadvantage is however that the pruning is restricted, such that only interleavers of size $V_{\text{max}}/2 \leq V \leq V_{\text{max}}$ can be obtained. If other sizes V shall be obtained, several starting interleavers have to be stored and the corresponding one has to be chosen depending on the block size of the data to be interleaved. Furthermore, the interleavers need (pre-computed) row-dependent starting parameters which have to be optimized offline. Even with those optimized parameters, the performance of this approach is still suboptimal compared to an S-random interleaver.

For those reasons, a different approach for generating prunable interleavers has been chosen for the FlexCode baseline channel coder. The FlexCode interleaver will be based on [FSB02]. In order to generate a prunable S-random interleaver π of maximum size V_{max}, we start with a small interleaver π_{min} of size V_{min} which fulfills the S-condition (see (2.7) and (2.8)). Then the additional $V_{\text{max}} - V_{\text{min}}$ entries are inserted using the following algorithm [FSB02]

a) set the first V_{min} elements of π identical to those of π_{min}

b) generate the sequence P of integers from $V_{\text{min}} + 1$ to V_{max}

c) set $M = V_{\text{min}}$ and $k = 1$

d) loop until $k = V_{\text{max}} - V_{\text{min}}$

e) generate a random position $j \in [1, M]$

f) compare $P(k)$ with $\pi_{\text{min}}(L)$, with $L \in [j - S, j + S - 1]$. If any $|P(k) - \pi_{\text{min}}(L)| < S$, go back to step 5.

g) set $\pi(L + 1) = \pi(L)$, for $L \in [M, M - 1, M - 2, \ldots, j]$

h) set $\pi(j) = P(k)$

i) $M = M + 1$ and $k = k + 1$

j) end of loop

The pruning to size V can be easily performed on the fly during (de-)interleaving: if an index $\pi(i) \geq V$ appears, increase i and test whether $\pi(i + 1) < V$. If not, continue increasing i.
The interleaver generated in this way fulfills the S-condition. However, as the S-condition has to be fulfilled for the smallest interleaver of size V_{min}. However, as has been noticed in Sec. 2.5, S needs to be smaller than $\sqrt{\frac{V_{\text{min}}}{2}}$ in order to find a permutation (with reasonable computational complexity on standard desktop computers). However, if V_{max} is large, the value of S could be larger. For this reason, it can be advantageous to store several prunable S-random interleavers of size $V_{\text{max},i}$ with $S < \sqrt{\frac{V_{\text{min,i}}}{2}}$ but $S > \sqrt{\frac{V_{\text{min,i-1}}}{2}}$ (for $i \geq 2$).

5.3 Interaction between Channel Decoder and Source Decoder

In this section, the interaction between channel decoder and source decoder is described. The interaction between channel decoder and source decoder is visualized in Fig. 5.6. First the channel decoder decodes the bitstream and reconstructs the GMM index. The GMM index is then immediately fed to the source decoder which uses it to generate stepsizes for the LSFs. These step sizes are for instance needed by an arithmetic decoder (see Sec. 5.4) to reconstruct the probability intervals. If the channel decoder has decoded the LSFs and the gain, they are fed to the source decoder which calculates stepsizes for the transform coefficients. After decoding of this first subframe, another gain is decoded and together with the LSFs, new stepsizes can be generated for the transform coefficients of the second subframe. Possibly, depending on the chosen codec setup, the decoded coefficients of the previous subframe need to be considered for determining the stepsizes (which is indicated by the dashed line in Fig. 5.6). This procedure is repeated for all four subframes of a block.

5.4 Reference Channel Coding

In order to compare the FlexCode channel coder with existing approaches, a reference channel coder has been defined. The reference channel coding approach is depicted in Fig. 5.7. The FlexCode source coder outputs quantized indices i which are entropy coded using an arithmetic coder [BCK07], [BCW90]. The arithmetic coder generates the bitstream and can thus be seen as a part of the channel coder.
In order to get the best possible compression by the arithmetic decoder, an appropriate model has to be used. The easiest possibility would be to use the average probabilities of occurrence of the quantized indices, however, this would not be very accurate as the speech and audio file to be encoded describes generally a highly non-stationary process thus leading to sub-optimal compression ratios. Therefore, an adaptive model is used in order to accurately calculate the probabilities of occurrence of the different indices to be encoded. The source encoder assumes the different transform coefficients to be Gaussian

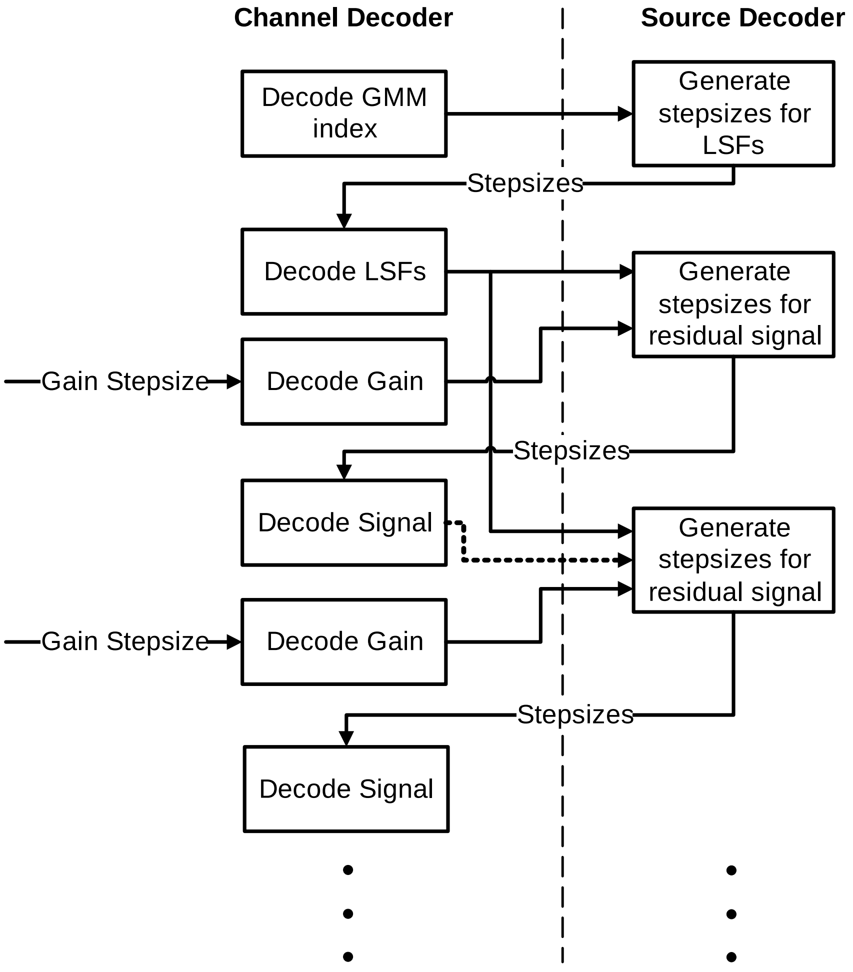


Figure 5.6: Flowchart of the interaction between channel decoder (left) and source decoder (right)

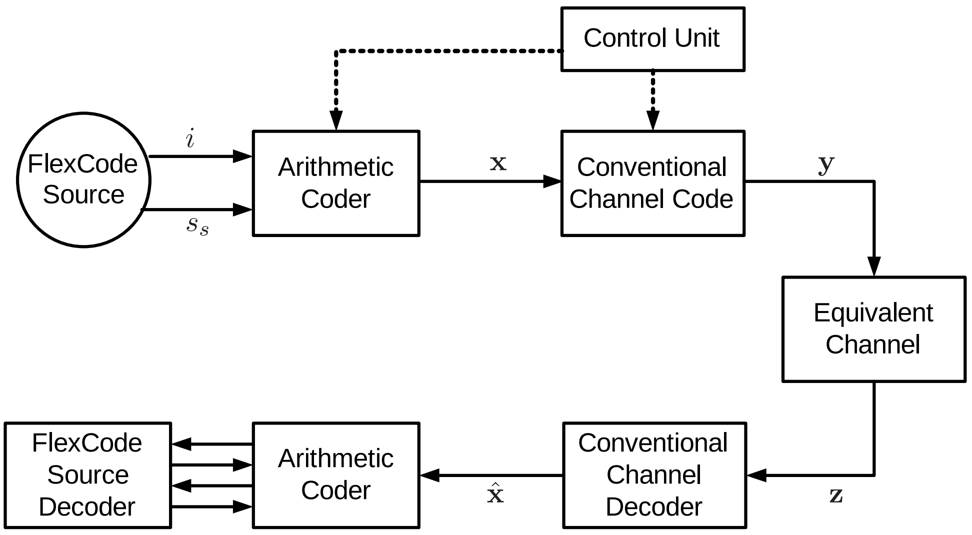


Figure 5.7: Reference channel coding

distributed with unit variance and then uses uniform quantization with a given stepsize $s_s(k)$. The stepsize is determined by the source encoder (and decoder) and then passed to the channel encoder, i.e., to the arithmetic encoder in the reference system. The stepsize is used to calculate the probabilities of occurrence of the different transform coefficients (or LSFs or gains) by evaluating

$$P(i(k)|s_s(k)) = \frac{1}{2}\left(\mathrm{erfc}\left(\frac{s_s(k)\left(i(k)-\frac{1}{2}\right)}{\sqrt{2}}\right) - \mathrm{erfc}\left(\frac{s_s(k)\left(i(k)+\frac{1}{2}\right)}{\sqrt{2}}\right)\right) \tag{5.5}$$

if the quantizer reproduction levels are described by $i(k) \cdot s_s(k)$, $i(k) \in \mathbb{Z}$. The arithmetic coder, as described for instance in [BCW90], starts by dividing the probability interval $[0;1)$ into smaller intervals which correspond to the probabilities of occurrence of the different symbols. The encoder picks one of those intervals according to the symbol which shall be encoded and then further subdivides this interval. The bitstream finally corresponds to one number inside this interval (usually the number the representation of which needs the least bits). The implementation of the arithmetic decoder utilizes fixed-point arithmetic and is based on [BCK07]. If the GMM index shall be encoded, the realization is simple as a fixed set of 16 different indices with *a priori* known probabilities is available. This set is used to partition the probability interval. The situation is different however for the other parameters (LSFs, gains, transform coefficients): theoretically, an infinite number of quantized indices can occur. For this reason, the cumulative distribution function (cdf) is used. For a Gaussian distribution with unit variance and zero mean, the cdf can be given by

$$F_n(a) = \int_{-\infty}^{a} p_n(\zeta)\mathrm{d}\zeta = 1 - \frac{1}{2}\mathrm{erfc}\left(\frac{a}{\sqrt{2}}\right) . \tag{5.6}$$

During encoding, the arithmetic encoder selects the lower bound of the selected probability interval to be $F_n\left(s_s(k)\left(i(k)-\frac{1}{2}\right)\right)$ and the upper bound to be $F_n\left(s_s(k)\left(i(k)+\frac{1}{2}\right)\right)$.
The arithmetically encoded bitstream is then encoded using a conventional channel decoder. The conventional channel decoder can be one of the following:

- a feed-forward convolutional code of rate $1/2$ and constraint length $J = 5$ as employed in the GSM EFR speech transmission. The generator polynomials are chosen to be $(23, 33)_{\text{oct}}$ [Pro01]. The convolutional encoder can easily cope with variable input block lengths as they are provided by the source encoder (in the constrained entropy case).

- The rate $1/2$ Turbo code specified by Berrou and Glavieux in [BG96]. As the interleaver has to be prunable in the constrained entropy case (variable size of the bitstream), the prunable S-random interleaver described in Sec. 5.2 is utilized.

After transmission over the channel model, the receiver first decodes the bitstream using a conventional channel decoder (which can be either a Viterbi decoder or a Turbo decoder, depending on the utilized channel code) and then reconstructs the quantized indices using the arithmetic decoder, which has to interact with the source decoder in order to get the stepsizes needed to calculate the probability intervals. In order to get an impression of the performance of the reference system, a reference simulation over an AWGN channel has been carried out. Instead of a specific audio or speech file, the source model described in Sec. 5.1 together with the dummy decoder described in Sec. 5.1.1 has been utilized. The simulation results in terms of *symbol error rate* (SER) are depicted in Fig. 5.8. By utilizing the Turbo code instead of the convolutional code a gain of approximately 3 dB is obtained at a symbol error rate of 10^{-2}. However, due to the small block size an error floor becomes visible at quite small symbol error rates. This error floor strongly depends on the S-value of the utilized interleaver (see also Sec. 2.5): the larger the S-value the lower the error floor. If the bit error probability would be depicted, the error floor

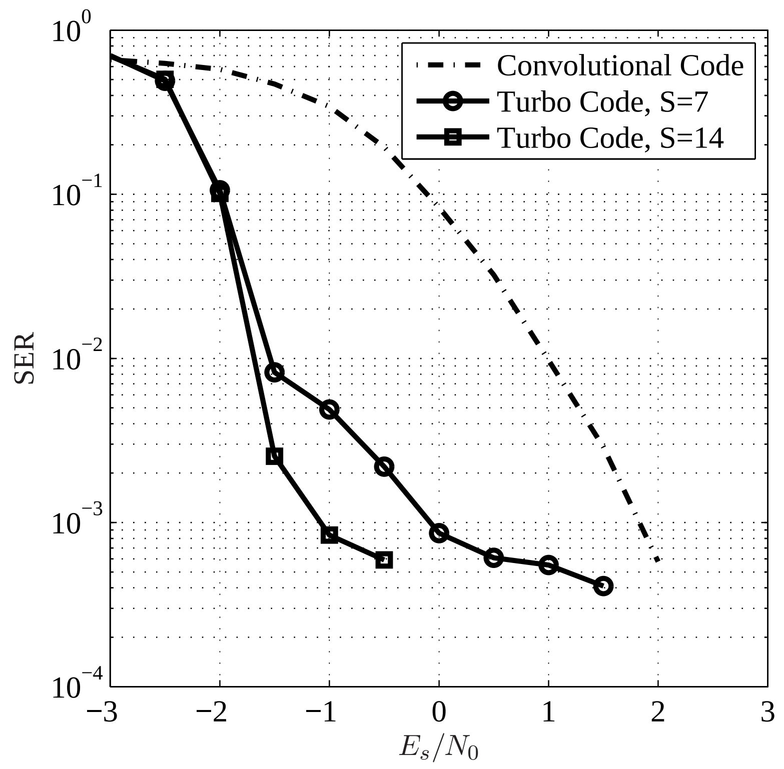


Figure 5.8: Symbol error rate for the reference transmission system depicted in Fig. 5.7 for two different channel codes

would be lower, but as a single error might lead to a loss of synchronization of the arithmetic decoder, the symbol error rate becomes considerably high.
One goal of the FlexCode baseline channel coder is to find an alternative to variable-length coding, such as arithmetic or Huffman coding. The drawback of arithmetic or Huffman coding is that single bit errors lead to a loss of synchronization and the resulting decoded parameters are either completely wrong (in the case of arithmetic coding) or insertion and deletion errors can occur (in the case of Huffman coding). Extensive work has been performed in order to apply soft decision decoding and iterative source-channel decoding to variable-length codes [Tho07b], [BH00], [TK05], [GFGR01], [GG04]. However, most of these algorithm show quite high computational complexity requirements and it has been shown that the ISCD approach using small block codes presented in Sec. 3.2 can have a similar decoding performance in terms of symbol error rate and parameter SNR as iterative source-channel decoding schemes for variable-length codes with a significantly lower computational complexity [Tho07a].

5.5 Preliminary Baseline Channel Coder

The proposed baseline channel coder is depicted in Fig. 5.9. Basically, the proposed system corresponds to a specialization of the system proposed in [SVCS08] and Sec. 4.1.1. The main difference is the control unit and the puncturing unit. The puncturing unit is mainly responsible for rate adaptation according to the current transmission channel and the rate requirements. The (global) control unit controls the selection of the block codes, of the convolutional codes and of the puncturing, according to the source and channel requirements.

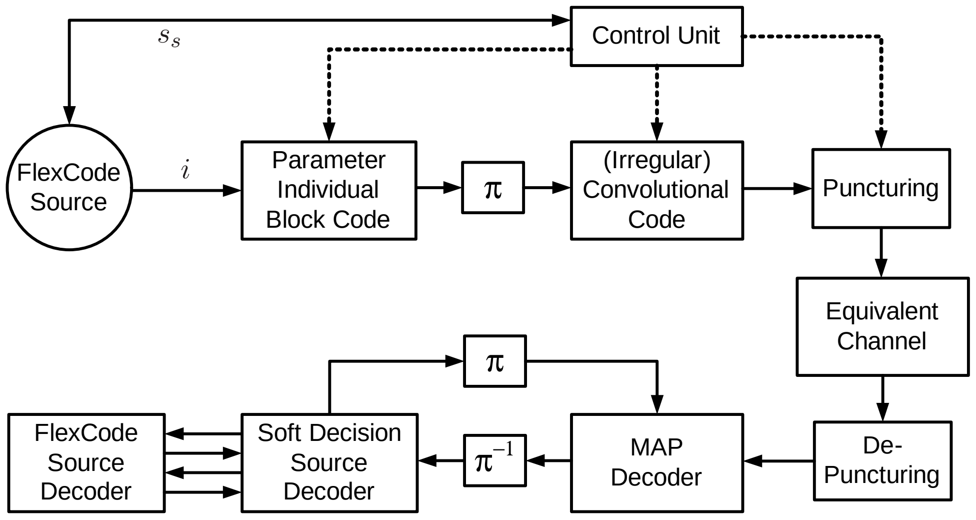


Figure 5.9: Block diagram of the preliminary FlexCode baseline channel coder

In the following, the FlexCode baseline channel coder will be described in detail. The quantizer indices $i(k)$ are first truncated (as theoretically an infinite number of indices can occur), and then encoded by a parameter individual block code (as described in Sec. 4.1.1), interleaved and then channel encoded. The channel encoder can for instance be a regular convolutional code, like a rate-$\frac{1}{2}$ recursive, systematic convolutional code. It has been shown [AKtB04], [AKtB02] that the inner component of a transmission system deploying iterative decoding should be of rate ≥ 1 in order to be capacity achieving. Therefore, the puncturing unit should at least puncture half of the bits in order to fulfill this constraint if an iterative decoder shall be employed. However, in order to be flexible, a rate $\leq \frac{1}{2}$ code is used as not all receivers will be able to perform iterative decoding but will use a non-iterative receiver, due to, e.g., power constraints of computational complexity constraints. On the other hand, the convolutional encoder can also realize an irregular convolutional code [TH02] which partitions the frame into several sub-frames which are encoded by individual convolutional codes. This enables a very easy optimization of the system performance by using the EXIT chart technique (see Sec. 2.4).
The receiver of Fig. 5.9 consists of a depuncturing unit, which transforms the punctured bits (which have not been transmitted) into erasures. The following decoder can be iterative or non-iterative and consists of a MAP decoder. The MAP decoder is usually realized as in the logarithmic domain is then called LogMAP decoder [HOP96]. A suboptimal approximation is the maxLogMAP decoder, described in [RVH95]. After deinterleaving, a soft-decision source decoder (SDSD) [Fin98], [FV01] decodes the quantized indices by performing a MAP decision (see Sec. 3.1). The soft decision source decoder can exploit the *a priori* knowledge of the source to estimate the quantized indices. The knowledge utilized here is mainly the redundancy introduced by the parameter individual block code and the *a priori* knowledge of order zero, i.e., the unequal distribution of the different indices, which is given by the stepsize and the Gaussian distribution, and can be evaluated by (5.5), if a stepsize is already available for the given parameter. If the decoder shall perform more than one iteration, extrinsic information is generated by the SDSD [AVS01], [Gor01], [AV05], [Cle06] (see also Sec. 3.2) and fed back to the channel decoder which uses this information as *a priori* information.

5.5.1 Parameter Individual Block Codes

In this section, first approaches for selecting the parameter individual block codes are presented. Note that these approaches are only first experiments and do not yet present a fully optimized approach.

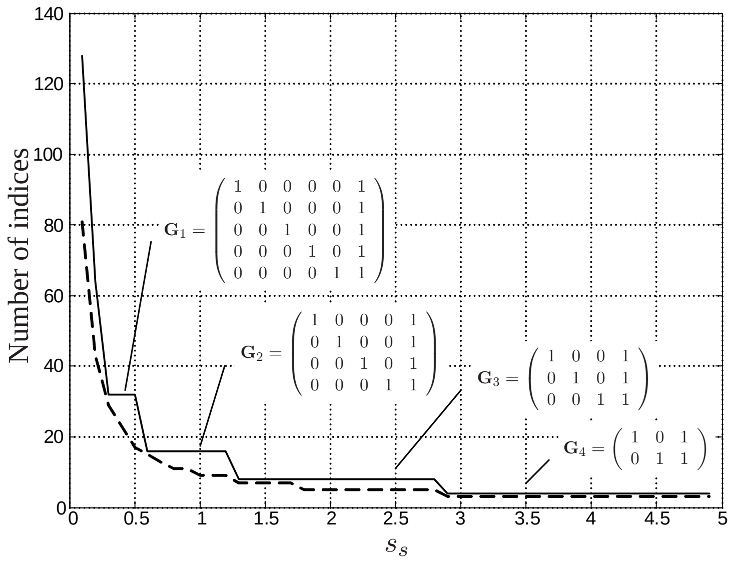


Figure 5.10: Number of indices with probability of occurrence $> 10^{-5}$ and assigned blockcodes as a function of the stepsize s_s

Stepsize-dependent code

As (most of the) parameters are assumed to be Gaussian distributed, the number of quantizer indices that occur with a certain probability varies with the stepsize of the quantizer. For instance, Fig. 5.10 depicts the number of different quantizer indices that occur with probability $> 10^{-5}$ as a function of the quantizer stepsize (dashed line). Let this curve be called $n_i(s_s)$. The solid line is then described by

$$\bar{n}_i(s_s) = 2^{\lceil \log_2(n_i(s_s)) \rceil} \tag{5.7}$$

The function $\bar{n}_i(s_s)$ is given by the solid line in Fig. 5.10. Then, according to the number of different indices, an appropriate block code can be chosen, for example, an $(\lceil \log_2(n_i(s_s)) \rceil, \lceil \log_2(n_i(s_s)) \rceil + h)$ block code, with h being the number of parity bits. The block codes $\mathbf{G}_1$ to $\mathbf{G}_4$ are examples for $h = 1$ (systematic single parity check code).
Figure 5.11 depicts the EXIT characteristics for stepsizes $s_s \in \{s_s \in \mathbb{R} | s_s = k \cdot 0.1, k \in \mathbb{N}$ and $0.1 \leq s_s \leq 4.9\}$. A single parity check code is utilized and the code size changes as a function of the stepsize mas described above. Of course, the quantizer entries have to be truncated by the source encoder if too large values do occur as the proposed system is not able to handle infinitely large indices.
If the stepsizes are known, the EXIT characteristic for the current block can be determined by averaging the corresponding characteristics [Tüc04], [TH02], [AKtB04]. On the other hand, by utilizing the pdf of the stepsizes, which can for example be approximated by (5.3), an average characteristic for the soft decision source decoder. This allows the optimization of the convolutional code and the respective puncturing pattern according to the current needs. Note that the EXIT characteristics do not reach the $(1, 1)$ point. This behavior has already been observed in [ABCV05] and is due to the unequal distribution of the bits at the output the encoder. The EXIT characteristic can only reach the point $(1, H(X))$.
With this setup, after source encoder, the size of a frame is of variable length, as due to the different stepsizes, different block codes are utilized. Therefore a flexible interleaver, such as the one described in Sec. 5.2, has to be used. The puncturing of the channel code can be fixed, leading to a variable transmission blocksize.

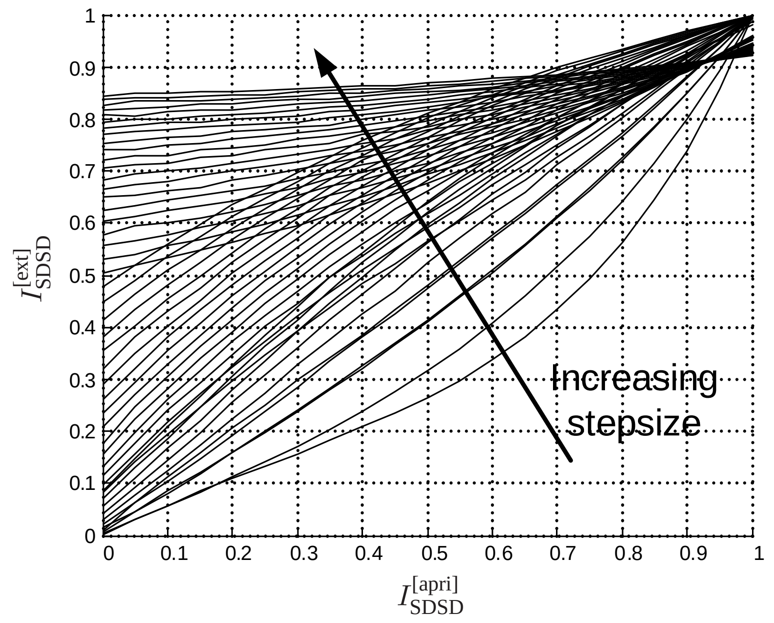


Figure 5.11: EXIT characteristics of the SDSD as a function of the stepsize (different codes per stepsize used)

The main problem of this proposed scheme can be found at the receiver side and occurs due to the interaction between source and channel decoder as described in Sec. 5.3. If only the GMM Index is not decoded correctly, no valid stepsizes are delivered by the source decoder and thus, the decoder can not decide which block-code has been used by the encoder, leading to a decoding failure. If only the first part of the block is decoded (e.g., GMM index and LSFs), and the stepsizes for the rest of the block cannot be determined, then only extrinsic information for the first few bits of the block can be calculated. With this little additional information however, the channel decoder is not able to generate a surplus of new extrinsic information itself. Therefore, the decoding fails if bit errors occur in the first part of the block. One way to avoid this behavior would be to use *unequal error protection* (UEP), which is frequently utilized in conventional transmission systems, like GSM or in video and image transmission schemes. However, little work has been done so far on UEP for joint source-channel coding schemes with iterative decoding.

Constant code

In order to avoid the decoding problems that occur with the previously presented scheme utilizing stepsize-dependent block codes, a second scheme using fixed codes is presented. All parameters are encoded with the same block code: for the illustrative example in this chapter, we chose a $(7, 8)$ single parity check code with generator matrix

$$\mathbf{G}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} . \tag{5.8}$$

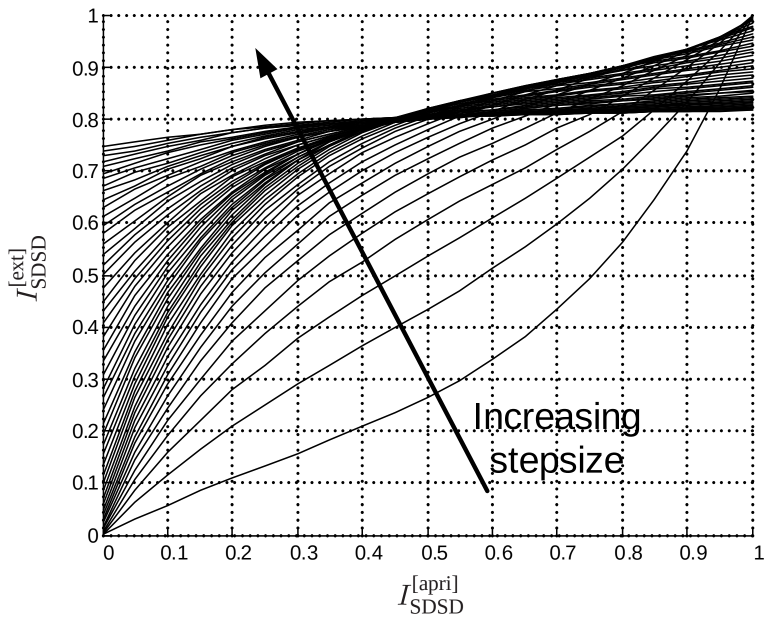


Figure 5.12: EXIT characteristics of the SDSD as a function of the stepsize

This means that a maximum number of $2^7 = 128$ quantizer reproduction levels can be encoded, regardless of the quantizer stepsize.
Figure 5.12 depicts the EXIT characteristics of the soft decision source decoder for stepsizes $s_s \in \{s_s \in \mathbb{R} \mid s_s = k \cdot 0.1, k \in \mathbb{N}$ and $0.1 \leq s_s \leq 4.9\}$. If the stepsize becomes larger, more *a priori* knowledge is available, as the probability that the quantizer indices 0 and ± 1 occur becomes large compared to the other indices. However, as for large stepsizes mainly 1 or 3 codewords occur, the entropy of the encoded bits $H(X)$ becomes considerably small. This can be seen in the EXIT chart: for small stepsizes, the EXIT characteristic is able to reach the $(1, 1)$ point, however, if the stepsize becomes larger, only $(1, H(X))$ is reached. For a stepsize of 4.9, the EXIT chart reaches the point $(1, 0.82)$. Again, either a specific characteristic for each block can be established by averaging the different characteristics or a global average characteristic (which only changes if the encoder settings change) can be computed. By choosing appropriate puncturing, the convolutional encoder can then be adapted to the source statistics.
At the decoder, again, the interaction between channel and source decoder, described in Sec. 5.3, causes some restrictions. During each iteration, the channel decoder tries to decode as much of the model as possible in order to get stepsizes which can be utilized to calculate the *a priori* knowledge in the SDSD, for instance using (5.5). Note that the complementary error function can be approximated by lookup tables in order to save computational power. During decoding, the parity check equations of each parameter individual block code can be evaluated (see also Sec. 4.1.3) and an error detection is performed. If an error has been detected, the stepsizes which will be determined by the source decoder will most likely be not correct and in this case no reliable *a priori* knowledge can be computed. In this case, the source decoder can utilize a conventional soft-input/soft-output block decoder (e.g., one belief propagation iteration [HOP96], [CF02] using the *boxplus* operation) or an SDSD with no *a priori* knowledge.
In a first experiment, no error detection has been utilized, as the utilized code (single parity check code defined by the generator matrix in (5.8)) possesses quite weak error detection capabilities. Therefore, if wrong stepsizes are determined by the source decoder, no appropriate measures are taken by the decoder (such as SDSD without *a priori* knowledge) but the wrong stepsizes are used in order to determine the

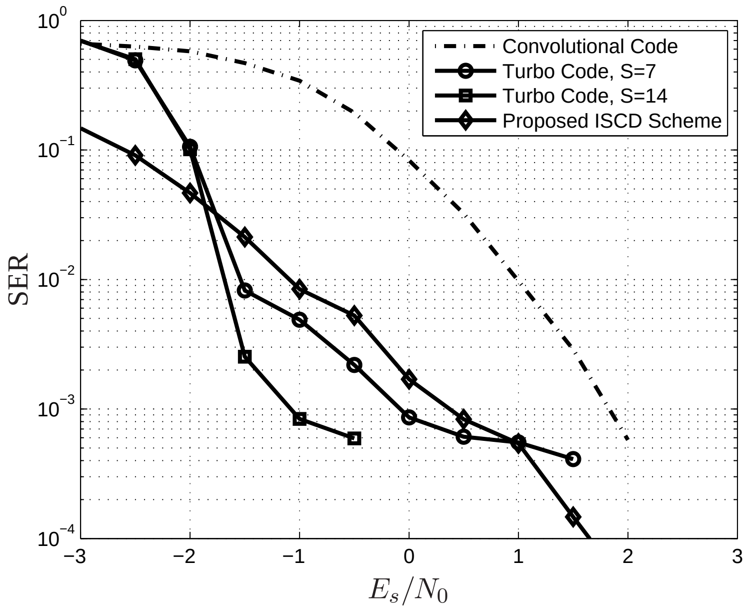


Figure 5.13: Simulation results of the proposed ISCD scheme

a priori knowledge. A comparison of the proposed scheme with the reference is given in Fig. 5.13. The source model described in Sec. 5.1 is used together with the dummy decoder described in Sec. 5.1.1. The dummy decoder is set up such that the maximum failure stepsize generation is used. The interleaver is an S-random interleaver with $S = 20$. In bad channel conditions, the symbol error rate is considerably lower as for the reference (order 10%), such that error concealment in the source might still deliver an understandable speech and audio impression. However, a steep Turbo cliff and the typical waterfall behavior, as would have been expected by an iterative, Turbo-like decoder is not observed. An explanation for this behavior are the completely wrong stepsizes which are delivered by the source decoder if the model has not been decoded correctly. For example, if the GMM index has not been decoded correctly in the first iteration, wrong stepsizes are generated for the rest of the frame. Therefore, the SDSD will generate unreliable and wrong extrinsic information which is fed back to the channel decoder. Therefore, the gains expected by the iterative decoding are smaller than expected. Note that however, the amount of transmitted bits is about twice is high as for the reference transmission. This is due to the fact that no optimized puncturing has been applied.

It has been shown in this section that a joint source-channel coding scheme with iterative decoding can achieve similar performance as the reference system which shows already near-optimum performance. The close interaction between source and channel decoder complicates the design of the channel coder, especially if all *a priori* knowledge shall be determined and utilized. The first version of the decoder did not yet incorporate error detection and mechanisms to avoid the use of the wrongly calculated stepsizes. As the model is quite important for getting the correct stepsizes, an unequal error protection scheme should be utilized in the future in order to have a reliable transmission of the model and thus reliable stepsizes for the decoding of the transform coefficients.

5.5.2 Extension towards Different Quantization Schemes

In the previous sections, the concepts of the FlexCode channel coding scheme have been shown for the constrained entropy quantization case with scalar quantization. The concept however is not restricted to this case, it can be easily extended to other quantizers, as will be shown in what follows.

Constrained Resolution Quantization

If constrained entropy quantization is applied, a certain distortion is fixed. This results in a variable bitrate of the encoded data, which can cause problems in circuit switched scenarios. Therefore, constrained resolution quantization, where the rate is fixed, resulting in constant blocksize but variable distortion, can be utilized. In order to remain flexible, the FlexCode source coder does not use special, optimized codebooks for constrained resolution quantization but a uniform quantizer. Before quantization, a compressor preprocesses the signal such that it is (almost) uniformly distributed before being quantized. At the decoder, the corresponding expander has to be used. From the signal model, the bit allocation algorithm determines the number of quantizer reproduction levels (and thus the number of bits) to be used for each transform coefficient such that the overall number of bits remains constant. As the number of levels is fixed, an appropriate parameter individual block code can be chosen by the index assignment.
As the compressor causes the signal to be nearly uniformly distributed, the *a priori* knowledge on parameter level is almost negligible. However, due to the constraints concerning delay and thus block size, some *a priori* knowledge can be expected and thus utilized.

Non-Scalar Quantization

Instead of a scalar quantizer, vector quantization can be utilized. As the FlexCode paradigm prohibits the use of specialized, trained codebooks, lattice vector quantization may be chosen. The use of vector quantization can be fully integrated in the proposed channel coding scheme as the operation is being performed on index level. However, if after channel decoding several indices have been decoded wrongly, several measures can be taken in order to minimize the distortion of the reconstructed signal. One of those measures is an optimized assignment of quantizer reproduction levels to indices (index assignment) [Vas07], [VT03]. These robust index assignment techniques are studied and developed by Nokia and may be incorporated in the final channel coder if lattice quantization is used.

5.6 FlexCode Baseline Channel Coder

In the previous section, the basic concept of iterative source-channel decoding has been applied to a preliminary version of the FlexCode baseline source coder. The integration of ISCD with the constrained-entropy quantizer has been clarified as well as the integration with the resolution constrained quantizer, which is more or less straightforward (see Section 5.5.2).
It has been found (see Sections 5.1.1 and 5.5) that it is not feasible to perform a joint source-channel decoding of the model parameters and the transform coefficients. The source-channel decoder requires knowledge about the model in order to determine the encoding parameters of the transform coefficients like bit allocation and utilized PIBCs. Therefore, we propose to utilize a separate transmission of the model parameters and the transform coefficients. The resulting structure, which defines the FlexCode baseline channel coder is depicted in Fig. 5.14.
The model parameters, such as, e.g., GMM index, LSFs, gain, are grouped and, if entropy constrained quantization is utilized, compressed using an arithmetic coder. On the other hand, if resolution constrained quantization is employed, the arithmetic encoding does not need to be carried out. Afterwards, the grouped bit stream is encoded using a strong conventional channel code. This channel code could for instance be an iteratively decodable code such as a Turbo code or an LDPC code. However, as the bit rate for transmitting the model parameters is rather small (around 5 kbit/s, see [KO07]), LDPC and Turbo codes might not be perfectly suited due to their relatively high error floor for small block sizes

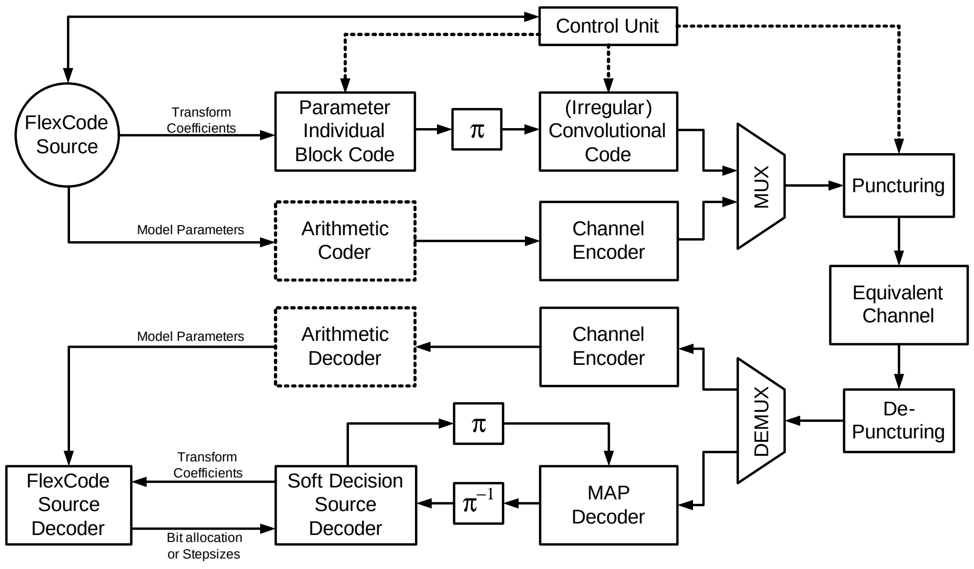


Figure 5.14: Block diagram of the FlexCode baseline channel coder

(see Fig. 2.8 for the example of Turbo codes). Therefore, it might be advantageous to deploy a "conventional" channel coding scheme such as the concatenation of a Reed-Solomon code and a convolutional code. This concatenation has been widely employed in existing communication systems [CHIW98]: The convolutional decoder at the receiver, which might be a Viterbi decoder, produces burst errors at its output which can be effectively corrected by the Reed-Solomon decoder. By using puncturing of the convolutional code, the rate and the robustness requirements can be effectively adjusted.
The transform coefficients on the other hand are encoded using the iterative source-channel coding system presented above. Using the model parameters, the source encoder determines the bit allocation (in the case of constrained-resolution quantization) or the step sizes (in the case of constrained-entropy quantization) which is fed to the channel encoder in order to select an appropriate parameter individual block code and a puncturing pattern. At the receiver, the joint decoding of model and transform coefficients, as proposed in Secs. 5.3 and 5.5 is no longer necessary. First, the model parameters are decoded and fed to the FlexCode source decoder which determines either the step sizes or the bit allocation. This information can then be used by the soft decision source decoder in the iterative source-channel decoding process. In contrast to the preliminary coder discussed in Sec. 5.5, the source decoder has now full knowledge about bit allocation and step sizes and can perform the decoding without any restrictions which considerably increases the performance of the system. However, the model needs to be perfectly known at the receiver. Therefore, the channel coding of the model has to be chosen such that the model can be decoded with high reliability. Additionally, error detection has to be included such that appropriate packet loss concealment measures can be taken by the FlexCode source decoder if the model (and thus also the transform coefficients) has not been received correctly.

5.7 Outlook

In this chapter, the FlexCode baseline channel coder, which realizes a joint source-channel coding approach with iterative decoding, has been introduced. However, the concept is still at an early phase of development due to the ongoing work in the source coder workpackage. Several open issues do exist and shall be addressed in the future. In this section, some of these open issues are introduced and possible solutions are presented.

One major issue of the present implementation is the strong interaction between source and channel coder introduced in Sec. 5.3. If one of the model parameters is decoded wrong, no valid *a priori* knowledge is generated thus leading to a decoding failure. Possible remedies include the already introduced error detection followed by a standard block code decoding without epxloiting the *a priori* knowledge or an unequal error protection scheme: a strong protection will be applied for the model part whereas a weaker protection may be applied for the transform coefficients. If the model is decoded correctly, the stepsizes are known and can be exploited for decoding the transform coefficients. An unequal error protection scheme can also be beneficial for the reference channel coding system deploying conventional channel decoding and arithmetic decoding. If a bit error occurs in the beginning of the block, the arithmetic decoder will most likely fail to decode the rest of the block. Therefore it is advantageous to better protect the beginning of the block and have a degrading performance towards the end of the block. Several wrongly decoded transform coefficients might have less impact on the perceived audio quality than wrongly decoded model parameters such as LSFs or gains.
A second possibility to overcome the problem of wrongly decoded model parameters might be to use a twofold channel coding scheme. The model is encoded using a strong conventional channel coding scheme such as Reed-Solomon codes, strong convolutional codes or even Turbo codes optimized for small block lengths. This is mainly a joint optimization problem of finding a suitable good together with a good interleaver [WK00].
One major issue is the complexity of the proposed scheme. Due to the iterative behavior of the decoder, the decoding complexity is scalable within certain limits. However, complexity measures have to be carried out in order to measure the complexity of the current channel coding and decoding scheme and to compare the complexity with known channel coding schemes.
Finally, if the concept of multiple description coding shall be used in the source coder, the concepts presented in Sec. 4.2 can be extended for working with the baseline channel coder.

Chapter 6

Practical Realization

In this chapter, the realization of the baseline channel coder is briefly outlined. The FlexCode channel coder is a more or less independent software modul which transforms the output of the FlexCode source encoder [Fle08] in such a way that a successful transmission can be performed using the FlexCode channel model [Fle07a].

Figure 6.1 depicts the realization of the transmission chain for the reference transmission system introduced in Sec. 5.4 using constrained-entropy quantization. The source encoder delivers quantized indices, stepsizes needed for determining the probabilities of occurence as well as *a priori* information such as the GMM. After generating a bit stream by the arithmetic coder and channel coding, the channel coded bitstream is transmitted over the channel model. The output of the channel model is decoded in order to reconstruct a bitstream which is then decoded using the arithmetic decoder. As the arithmetic decoder needs the identical stepsizes as the respective encoder in order to invert the encoding operation, the source decoder has to continuously interact with the arithmetic decoder to deliver new stepsizes. This interaction has been described in Sec. 5.3.

On the other hand, if constrained resolution quantization is utilized, the arithmetic coding and decoding depicted in Fig. 6.1 is no longer needed. The source encoder delivers the indices as well as information about the bit allocation to the channel encoder which can then generate a channel encoded bit stream. At

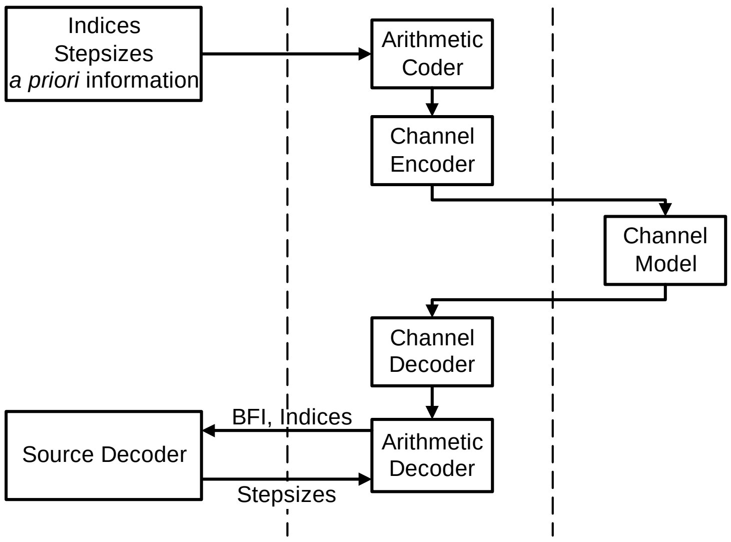


Figure 6.1: Realization of the interface between WP1 and WP2 for the reference transmission system described in Sec. 5.4

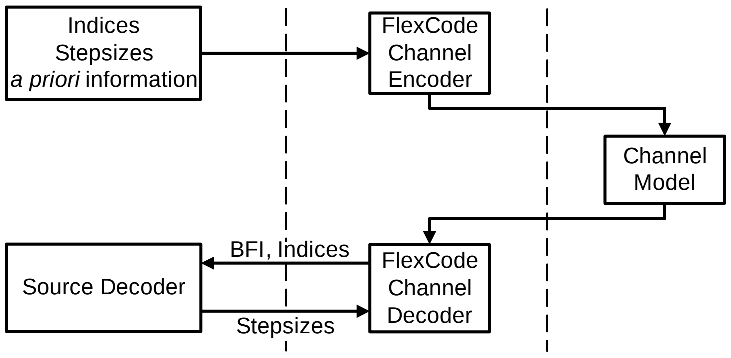


Figure 6.2: Realization of the interface between WP1 and WP2 for the FlexCode baseline channel coder described in Sec. 5.5

the receiver, the arithmetic decoder is also no longer needed. The source encoder can also be configured such some parameters are quantized with constrained resolution (e.g., the model parameters) and the other with constrained entropy (e.g., the transform coefficients).
Figure 6.2 depicts the transmission chain of the baseline channel coder introduced in Sec. 5.6. Via the interface, the channel coder receiver the quantized indices of the model parameters and the transform coefficients as well as the information needed for the index assignment: stepsizes or bit allocation, according to the chosen quantization. At the receiver, the channel decoder interacts with the model as described in Secs. 5.3 and 5.6. After having decoded the model, the channel decoder receives asll necessary information to decode the transform coefficients.

Chapter 7

Conclusion

In this report the baseline FlexCode channel coder has been introduced. After a small initiation to modern, iterative channel coding and decoding techniques, the concepts of soft-decision source decoding and iterative source-channel decoding, which are an integral part of the FlexCode channel coder have been introduced. Several advancements helping to improve the flexibility as well as the computational complexity requirements of iterative source-channel decoders, have been shown in order to introduce the baseline channel coder. The baseline channel coder is compared to a reference channel coder which does already feature an inherent flexibility. It has been shown that the proposed approach can compete with the reference approach, deploying state-of-the-art Turbo codes jointly with arithmetic coding. This report however only outlines the concepts of the FlexCode channel coder as the source encoder still is in development. However, the final concept will be based upon the concepts presented in Chapter 5, with the respective modifications in order to fulfill the requirements of the source encoder. Finally, after a brief explanation of the practical realization aspects, the report concludes with Appendix A where some theoretical results are shown. These results were obtained during the FlexCode project and are of a certain utility during the design and the optimization of a channel encoder with a decoder based on the Turbo principle.

Appendix A

EXIT Characteristics of Feed Forward Convolutional Codes

Introduced in [tB99], EXIT charts have become an important tool for the convergence analysis of concatenated systems with iterative evaluation of extrinsic information at the receiver. EXIT characteristics plot the mutual information $I^{\text{[ext]}}$ of the extrinsic output of a decoder as a function of the mutual information $I^{\text{[apri]}}$ of the *a priori* input.

To clarify notations, Fig. A.1 shows the block diagram of an EXIT chart measurement circuit [AKtB04] for a feed forward convolutional code employed in a parallel concatenated (PC) iterative decoding scheme or as inner component in a serially concatenated (SC) iterative decoding scheme. A binary source S generates a vector $\mathbf{x}$ of binary data bits. The vector $\mathbf{x}$ is encoded to a vector $\mathbf{y}$ using a convolutional encoder. After transmission over a communication channel the MAP SISO decoder [BMDP98] receives a possibly noisy vector $\mathbf{z}$. Here, the communication channel consists of BPSK modulation with symbol energy $E_\text{s} = 1$, AWGN with noise variance $\sigma_n^2 = N_0/2$, BPSK demodulation, and conversion to L-values [HOP96]. The MAP SISO decoder receives additional *a priori* information on the data bits $\mathbf{x}$ from the extrinsic output of the (notional) second constituent decoder of the iterative decoding scheme. This information is modelled by the *extrinsic channel* which adds Gaussian noise with defined mean and variance [tB01a], [AKtB04] to the bipolar representation of those bits. Note that the inputs and the outputs of the MAP SISO decoder are represented as L-values. The MAP SISO decoder outputs extrinsic information on the data bits $L^{\text{[ext]}}(\hat{\mathbf{x}})$ and extrinsic information on the encoded bits $L^{\text{[ext]}}(\hat{\mathbf{y}})$. In our case, the mutual information of interest to compute the EXIT charts is the *a priori* mutual information $I^{\text{[apri]}} = I(X; L_{\hat{X}}^{\text{[apr]}})$ and the extrinsic mutual information $I^{\text{[ext]}} = I(X; L_{\hat{X}}^{\text{[ext]}})$. Note that $\mathbf{x}$ is a realization of the random process X and accordingly, $L^{\text{[ext]}}(\hat{\mathbf{x}})$ and $L^{\text{[apr]}}(\hat{\mathbf{x}})$ are realizations of the corresponding random processes $L_{\hat{X}}^{\text{[ext]}}$ and $L_{\hat{X}}^{\text{[apr]}}$.

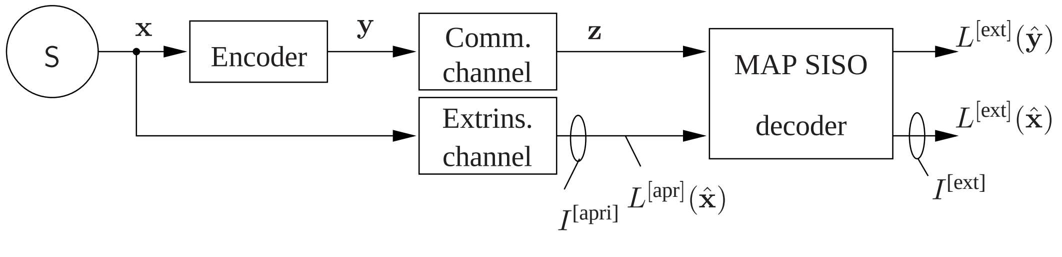


Figure A.1: Block diagram showing the measurement of the EXIT chart of a MAP SISO convolutional decoder [AKtB04]

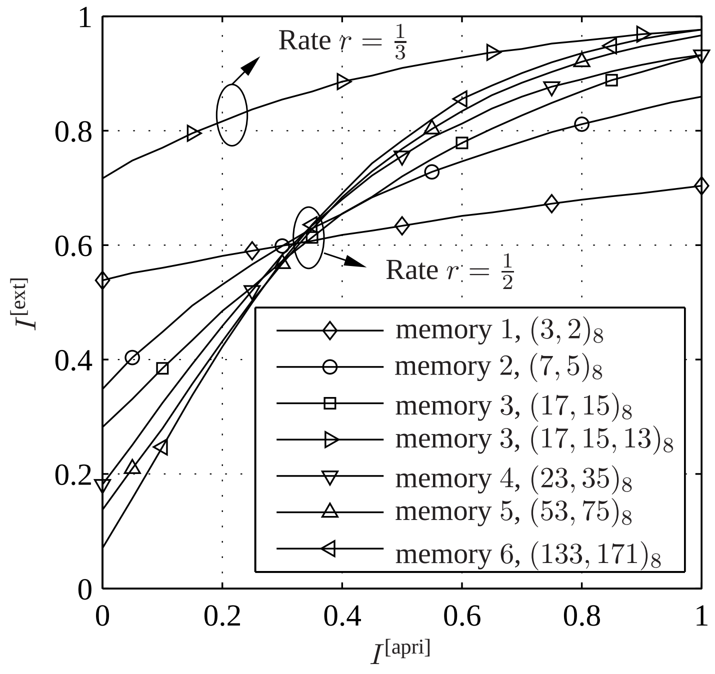


Figure A.2: EXIT characteristics of different feed forward convolutional codes for $E_\mathrm{s}/N_0 = 1/(2\sigma_n^2) = -5\,\text{dB}$

Figure A.2 shows the EXIT characteristics of different rate $1/2$ and rate $1/3$ feed forward convolutional codes acting in a parallel concatenated system or as inner code in a serially concatenated system, i.e., the input to the decoder consists of the *a priori* knowledge on the data bits and of the received channel values of the encoded bits; the decoder generates extrinsic information $L^{[\text{ext}]}(\hat{\mathbf{x}})$ for the data bits $\mathbf{u}$. As visible in Fig. A.2, the characteristics do not reach the point $(I^{[\text{apri}]} = 1\,\text{bit}\,, I^{[\text{ext}]} = 1\,\text{bit})$ which is needed for perfect decoding. Table A.1 lists the maximum mutual information for perfect *a priori* knowledge (i.e., $I^{[\text{apri}]} = 1\,\text{bit}$) for the codes of Fig. A.2. An easy to understand explanation of this behavior and an expression to analytically compute the mutual information $I^{[\text{ext}]}$ for $I^{[\text{apri}]} = 1\,\text{bit}$ is given in Section A.1.

Table A.1: Measured maximum mutual information for the codes in Fig. A.2 and $E_\mathrm{s}/N_0 = 1/(2\sigma_n^2) = -5\,\text{dB}$

Generator polynomials	$I^{[\text{ext}]}\Big\vert_{I^{[\text{apri}]}=1}$
$(3,2)_8$	0.7038
$(7,5)_8$	0.8595
$(17,15)_8$	0.9316
$(17,15,13)_8$	0.9764
$(23,35)_8$	0.9324
$(53,75)_8$	0.9665
$(133,171)_8$	0.9762

A.1 Maximum Attainable Mutual Information

It has already been observed in [tB01b] that the EXIT characteristics of feed forward convolutional codes do not reach $I^{[\text{ext}]} = 1$ bit for $I^{[\text{apri}]} = 1$ bit. The explanation given is based on the fact that the coupling of the bits is limited by the constraint length of the code. We give a more detailed explanation of this behavior using the Trellis representation of the convolutional code. In Section A.1.2, we consider the problem theoretically and provide an analytical solution for the maximum attainable mutual information.

A.1.1 Illustrative Explanation

The behavior of imperfect mutual information if perfect *a priori* knowledge is available can best be visualized using the Trellis representation of the convolutional code. Figure A.3 depicts parts of the Trellis diagram of a memory $J = 2$ feed forward convolutional code. Without loss of generality (due of the linearity of the code), it can be assumed that the all-zero path has been encoded and transmitted. Determining the extrinsic information for $\hat{u}_k$ at time instant k means determining the probability that the estimated data bit $\hat{u}_k$ at time instant k is either 0 or 1 if *a priori* information on the data bits x_k is available at all time instants except for k. If we take a look at Fig. A.3, we see immediately that, if it is perfectly known that the all-zero data sequence has been sent ($I^{[\text{apri}]} = 1$ bit), the decision at instant k cannot be determined by purely considering the *a priori* knowledge at all time instants except k. However, in order to compute the extrinsic output at instant k, the decision on $\hat{u}$ has to be made using the channel output only. This decision is not influenced by the *a priori* knowledge as the (perfectly known) inputs at instants $k+1$, $k+2$, etc. lead to the same inner state of the convolutional encoder after J inputs (due to the non-recursive structure of the code).

In the case of recursive convolutional codes, however, a different data input x_k at time instant k and identical, perfectly known inputs at subsequent instants do not lead to the same state after J inputs, resulting from the recursiveness of the encoder as shown in Fig. A.4. A different decision at instant k, followed by perfectly known data bits leads to a different Trellis path which does not end in the same state. Thus, if the encoder is terminated (i.e., the encoding stops in a defined state) and the input vector $\mathbf{u}$ is of finite length, perfect *a priori* knowledge leads to a non-ambiguous decision on the extrinsic output at instant k. If the recursive code is not terminated, similar effects as in the case of feed forward codes are observed, i.e., no perfect extrinsic information can be generated even if perfect *a priori* knowledge is available. Nevertheless, the remainder of this appendix will only focus on feed forward convolutional codes.

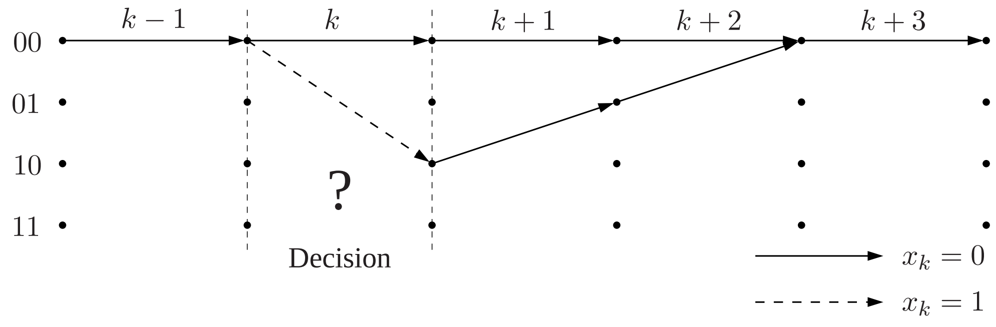


Figure A.3: Extrinsic Decision if perfect *a priori* knowledge is available in the case of a memory $J = 2$ feed forward convolutional code

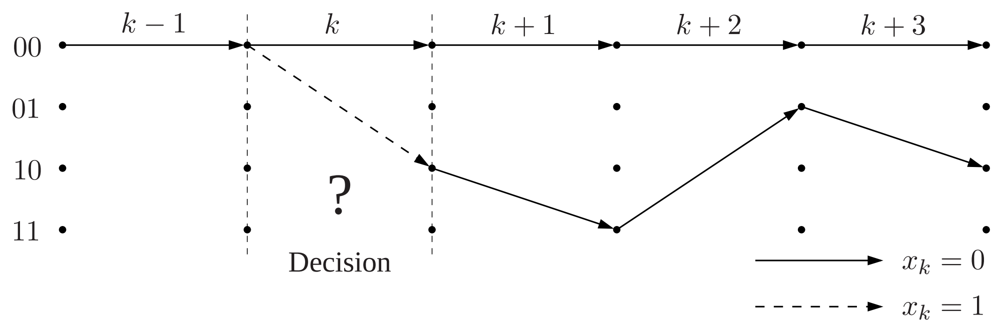


Figure A.4: Extrinsic Decision if perfect *a priori* knowledge is available in the case of a memory $J = 2$ recursive convolutional code

A.1.2 Theoretical Results

Definition A.1.1 *Let* C *be a rate* $r = 1/N$ *feed forward convolutional encoder with time-domain generator matrix* $\mathbf{G}$*. Let* $\mathbf{x}^{(1)} = (1, 0, \ldots, 0)$ *be a weight one input vector of length* $J + 1$*. The vector* $\mathbf{x}^{(1)}$ *is encoded by* C *to the code vector* $\mathbf{y}^{(1)} = \mathbf{x}^{(1)}\mathbf{G} = \left(y_1^{(1)}, y_2^{(1)}, \ldots\right)$ *with* $y_k^{(1)} \in \{0; 1\}$*. The vector* $\mathbf{y}^{(1)}$ *is also denoted as the impulse response of the convolutional code. The Hamming weight of the impulse response vector* $\mathbf{y}^{(1)}$ *is defined as* $\bar{D}_{\mathsf{C}}$ *and it holds*

$$\bar{D}_{\mathsf{C}} = \sum_{k=0}^{N(J+1)} y_k^{(1)} . \tag{A.1}$$

Theorem A.1.2 *Given a rate* $r = 1/N$ *feed forward convolutional code* C *with memory* J *and transmission over an AWGN channel with noise variance* σ_n^2*, the* L*-values of the extrinsic MAP SISO decoder output* $L^{[ext]}(\hat{\mathbf{x}})$ *show a Gaussian distribution with mean* $\mu_E = 2\frac{\bar{D}_{\mathsf{C}}}{\sigma_n^2}$ *and variance* $\sigma_E^2 = 2\mu_E = 4\frac{\bar{D}_{\mathsf{C}}}{\sigma_n^2}$ *if perfect* a priori *knowledge on the equiprobable data bits* $\mathbf{x}$ *is available (i.e.,* $I^{[apri]} = 1$ *bit).*

Proof (*This proof uses intermediate results from [KHC06]*)
Let $\mathbf{x}^{(0)} = (x_1^{(0)}, \ldots, x_{J+1}^{(0)}) = (0, \ldots, 0)$ be the all zero vector and $\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)} \ldots, x_{J+1}^{(1)}) = (1, 0, \ldots, 0)$ a weight one input vector. The length of the input vectors can be restricted to $J + 1$, as after J identical inputs, the feed forward convolutional encoder will have the same inner state. Let $\mathbf{G}$ be the time domain generator matrix of the feed forward convolutional code C. Encoding both vectors $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ with C produces the outputs

$$\begin{aligned}
\mathbf{y}^{(0)} &= \mathbf{x}^{(0)}\mathbf{G} = \left(y_1^{(0)}, \ldots, y_{N(J+1)}^{(0)}\right) = (0, 0, \ldots, 0) \\
\mathbf{y}^{(1)} &= \mathbf{x}^{(1)}\mathbf{G} = \left(y_1^{(1)}, \ldots, y_{N(J+1)}^{(1)}\right) .
\end{aligned}$$

Let $\mathbf{x}_{\backslash 1}^{(i)} = \left(x_2^{(i)}, \ldots, x_{J+1}^{(i)}\right) = (0, 0, \ldots, 0)$, $i \in \{0; 1\}$, denote the vector of length J which does not contain the first element of either $\mathbf{x}^{(0)}$ or $\mathbf{x}^{(1)}$. Due to the linearity of the convolutional code, it is sufficient to perform the proof for the all-zero vector only.
The encoded vector $\mathbf{y}^{(0)}$ of length $N(J + 1)$ is BPSK modulated onto the vector $\ddot{\mathbf{y}}^{(0)}$ with elements $\ddot{y}_k^{(0)} = 1 - 2y_k^{(0)} = +1$, $k = 1, 2, \ldots, N(J + 1)$ and $\mathbf{y}^{(1)}$ is modulated onto $\ddot{\mathbf{y}}^{(1)}$. After transmission

over a channel with additive white Gaussian noise of zero mean and variance $\sigma_n^2 = N_0/2$, the vector $\ddot{\mathbf{z}}^{(0)}$ is received with $\ddot{\mathbf{z}}^{(0)} = \ddot{\mathbf{y}}^{(0)} + \mathbf{n}$, and $\mathbf{n} = \left(n_1, \ldots, n_{N(J+1)}\right)$ denoting the noise vector. The extrinsic outputs of the MAP SISO decoder are the probabilities that the decoded data bit $\hat{x}_k$ is either 0 or 1 under the condition that *a priori* knowledge on all data bits except the one at position k and the entire received sequence $\ddot{\mathbf{z}}^{(0)}$ are available. Without loss of generality, it is sufficient to consider only the first bit position of the vectors, as a consequence of the linearity of convolutional codes. Using the Bayes theorem and the assumption that the data bits x_k are equiprobable, the extrinsic probabilities can be expressed as (with $\ell \in \{0; 1\}$) [KHC06]

$$P(\hat{x}_1 = \ell | \ddot{\mathbf{z}}^{(0)}, \mathbf{x}_{\setminus 1}^{(0)}) = \frac{p\left(\ddot{\mathbf{z}}^{(0)} | \mathbf{x}^{(\ell)}\right)}{p\left(\ddot{\mathbf{z}}^{(0)} | \mathbf{x}^{(0)}\right) + p\left(\ddot{\mathbf{z}}^{(0)} | \mathbf{x}^{(1)}\right)}$$

with

$$\begin{aligned} p\left(\ddot{\mathbf{z}}^{(0)} | \mathbf{x}^{(0)}\right) &= \left(\frac{1}{\sqrt{2\pi}\sigma_n}\right)^{N(J+1)} \cdot \prod_{\kappa=1}^{N(J+1)} \mathrm{e}^{-\frac{n_\kappa^2}{2\sigma_n^2}} \\ p\left(\ddot{\mathbf{z}}^{(0)} | \mathbf{x}^{(1)}\right) &= \left(\frac{1}{\sqrt{2\pi}\sigma_n}\right)^{N(J+1)} \cdot \prod_{\kappa=1}^{N(J+1)} \mathrm{e}^{-\frac{(n_\kappa + d_\kappa)^2}{2\sigma_n^2}} \end{aligned}$$

and d_k elements of the vector $\mathbf{d} = \ddot{\mathbf{y}}^{(0)} - \ddot{\mathbf{y}}^{(1)}$, i.e., $d_k \in \{0; +2\}$. Using vector notation, the extrinsic probabilities can be expressed as

$$P(\hat{x}_1 = 0 | \ddot{\mathbf{z}}^{(0)}, \mathbf{x}_{\setminus 1}^{(0)}) = \frac{\exp\left(\frac{|\mathbf{n}+\mathbf{d}|^2 - |\mathbf{n}|^2}{2\sigma_n^2}\right)}{1 + \exp\left(\frac{|\mathbf{n}+\mathbf{d}|^2 - |\mathbf{n}|^2}{2\sigma_n^2}\right)} \tag{A.2}$$

$$P(\hat{x}_1 = 1 | \ddot{\mathbf{z}}^{(0)}, \mathbf{x}_{\setminus 1}^{(0)}) = \frac{1}{1 + \exp\left(\frac{|\mathbf{n}+\mathbf{d}|^2 - |\mathbf{n}|^2}{2\sigma_n^2}\right)} \tag{A.3}$$

with $|\mathbf{n}|^2 = \sum_{\kappa=1}^{N(J+1)} n_k^2$ and $|\mathbf{n} + \mathbf{d}|^2$ defined similarly. By using (A.2) and (A.3) the extrinsic L-values $L^{[\text{ext}]}(\hat{\mathbf{x}})$ can be determined as

$$\begin{aligned} L^{[\text{ext}]}(\hat{x}_1) &= L(\hat{x}_1 | \ddot{\mathbf{z}}^{(0)}, \mathbf{x}_{\setminus 1}^{(0)}) \\ &= \ln\left(\frac{P(\hat{x}_1 = 0 | \ddot{\mathbf{z}}^{(0)}, \mathbf{x}_{\setminus 1}^{(0)})}{P(\hat{x}_1 = 1 | \ddot{\mathbf{z}}^{(0)}, \mathbf{x}_{\setminus 1}^{(0)})}\right) \\ &= \frac{|\mathbf{n}+\mathbf{d}|^2 - |\mathbf{n}|^2}{2\sigma_n^2}\,. \end{aligned} \tag{A.4}$$

The factor $|\mathbf{n} + \mathbf{d}|^2 - |\mathbf{n}|^2$ can be further simplified

$$\begin{aligned} |\mathbf{n}+\mathbf{d}|^2 - |\mathbf{n}|^2 &= \sum_{\kappa=1}^{N(J+1)} (n_\kappa + d_\kappa)^2 - \sum_{\kappa=1}^{N(J+1)} n_\kappa^2 \\ &= 2 \sum_{\kappa=1}^{N(J+1)} n_\kappa d_\kappa + \sum_{\kappa=1}^{N(J+1)} d_\kappa^2 \\ &= 2 \sum_{\kappa=1}^{N(J+1)} n_\kappa d_\kappa + 4\bar{D}_{\mathsf{C}}\,, \end{aligned}$$

using the fact that $\sum_{\kappa=0}^{N(J+1)} d_\kappa^2 = 4\bar{D}_\mathsf{C}$ (see definition of $\bar{D}_\mathsf{C}$). Therefore, we obtain

$$L(\hat{x}_1|\ddot{\mathbf{z}}^{(0)}, \mathbf{x}_{\backslash 1}^{(0)}) = 2\frac{\bar{D}_\mathsf{C}}{\sigma_n^2} + \sum_{\kappa=1}^{N(J+1)} \frac{d_\kappa}{\sigma_n^2} n_\kappa \,. \tag{A.5}$$

Equation (A.5) states that the L-values of the extrinsic output are composed by the sum of weighted Gaussian distributed noise values and an offset $2\bar{D}_\mathsf{C}/\sigma_n^2$. A random process composed by sum of Gaussian distributed processes is again a Gaussian process [PU02]. The mean of the resulting process is the sum of the means of the sub-processes and the resulting variance is the sum of the sub-processes' variances. As the noise samples n_k have zero mean, the process resulting from the sum of all noise samples has zero mean. As a consequence, the mean of the L-values is only determined by the offset in (A.5) and the mean μ_E of the L-value distribution is thus

$$\mu_E = 2\frac{\bar{D}_\mathsf{C}}{\sigma_n^2} \,.$$

The variance of the noise samples n_κ is σ_n^2, but as the noise samples n_κ are scaled by d_κ/σ_n^2, the variances of the scaled samples in the sum (A.5) amount to $\sigma_n^2 \cdot \frac{d_\kappa^2}{\sigma_n^4} = d_\kappa^2/\sigma_n^2$. Therefore, the total variance σ_E^2 of the L-value distribution is

$$\sigma_E^2 = \sum_{\kappa=1}^{N(J+1)} \frac{d_\kappa^2}{\sigma_n^2} = \frac{1}{\sigma_n^2} \sum_{\kappa=1}^{N(J+1)} d_\kappa^2 = 4\frac{\bar{D}_\mathsf{C}}{\sigma_n^2} = 2\mu_E \,. \quad \square$$

Corollary A.1.3 *Given a feed forward convolutional code* C*, the hard decision bit error probability of the extrinsic output after transmission over an AWGN channel with noise variance* σ_n^2 *and after MAP SISO decoding is given by* $P_b^{[\text{ext}]}\big|_{I^{[apri]}=1} = \frac{1}{2}\text{erfc}\left(\frac{\sqrt{\bar{D}_\mathsf{C}}}{\sqrt{2}\sigma_n}\right)$ *under the condition that perfect* a priori *knowledge on the data bits* $\mathbf{x}$ *is available (*$I^{[apri]} = 1$ *bit).*

Proof Due to the linearity of the convolutional code C, it can be assumed, without loss of generality, that the all-zero codeword has been sent. It is known from Theorem A.1.2 that the pdf of the extrinsic information, given that the all-zero codeword $\mathbf{x}^{(0)} = (0, 0, \ldots, 0)$ has been encoded resulting in the transmitted BPSK modulated vector $\ddot{\mathbf{y}}^{(0)} = (+1, +1, \ldots, +1)$, is Gaussian distributed with mean $\mu_E = 2\frac{\bar{D}_\mathsf{C}}{\sigma_n^2}$ and variance $\sigma_E^2 = 4\frac{\bar{D}_\mathsf{C}}{\sigma_n^2}$. Thus, the bit error probability of the extrinsic output can be determined as

$$\begin{aligned} P_b^{[\text{ext}]}\bigg|_{I^{[\text{apri}]}=1} &= \int_{-\infty}^{0} p_E(\xi)\mathrm{d}\xi = \frac{1}{\sqrt{2\pi}\sigma_E} \int_{-\infty}^{0} \mathrm{e}^{-\frac{(\xi-\mu_E)^2}{2\sigma_E^2}} \mathrm{d}\xi \\ &= \frac{1}{2}\text{erfc}\left(\frac{\mu_E}{\sqrt{2}\sigma_E}\right) = \frac{1}{2}\text{erfc}\left(\frac{\sqrt{\bar{D}_\mathsf{C}}}{\sqrt{2}\sigma_n}\right) \,. \quad \square \end{aligned}$$

Corollary A.1.4 *The mutual information* $I^{[ext]}$ *between the data bits* $\mathbf{u}$ *and the extrinsic output of the decoded bits* $\hat{u}$ *(*L*-value representation) of a MAP SISO decoder for a feed forward convolutional code* C*, given the conditions that perfect* a priori *knowledge on the data bits* $\mathbf{x}$ *is available (*$I_A = 1$ *bit), that the data bits* $\mathbf{u}$ *are equiprobable, and that the transmission is performed on an AWGN channel, depends solely on the noise variance* σ_n^2 *and on the Hamming weight* $\bar{D}_\mathsf{C}$ *of the impulse response. This mutual information can be expressed as*

$$I^{[ext]}\bigg|_{I^{[apri]}=1} = 1 - \frac{\sigma_n}{\sqrt{8\pi\bar{D}_\mathsf{C}}} \int_{-\infty}^{+\infty} \mathrm{e}^{-\frac{\left(\xi\sigma_n^2 - 2\bar{D}_\mathsf{C}\right)^2}{8\bar{D}_\mathsf{C}\sigma_n^2}}\, ld\left(1 + \mathrm{e}^{-\xi}\right) \mathrm{d}\xi \,.$$

Proof Due to the linearity of convolutional codes, we can assume, without loss of generality, that the all zero sequence has been encoded and thus the $(+1, +1 \ldots, +1)$ sequence has been transmitted. Therefore, according to Theorem A.1.2, the L-values of the extrinsic decoder output show a Gaussian distribution with mean $\mu_E = 2\frac{\bar{D}_{\mathsf{C}}}{\sigma_n^2}$ and variance $\sigma_E^2 = 4\frac{\bar{D}_{\mathsf{C}}}{\sigma_n^2}$. According to [tB01a], the mutual information $I^{[\text{ext}]}$ can then be expressed as

$$I^{[\text{ext}]}(\sigma_E) = 1 - \int_{-\infty}^{+\infty} \frac{\mathrm{e}^{-((\xi-\mu_E)^2/(2\sigma_E^2))}}{\sqrt{2\pi}\sigma_E} \mathrm{ld}\left(1 + \mathrm{e}^{-\xi}\right) \mathrm{d}\xi. \tag{A.6}$$

Substituting $\mu_E = 2\frac{\bar{D}_{\mathsf{C}}}{\sigma_n^2}$ and $\sigma_E^2 = 4\frac{\bar{D}_{\mathsf{C}}}{\sigma_n^2}$ into (A.6) proves the corollary. $\square$

A.2 Numerical Simulation Results

Figure A.5 shows the result of Corollary A.1.4, i.e., the mutual information of the extrinsic output of a MAP SISO decoder of a feed forward convolutional code C if perfect *a priori* knowledge is available. It can easily be seen that the upper right point of the EXIT characteristic (i.e., $I^{[\text{ext}]} = 1\,\text{bit}$, given $I^{[\text{apri}]} = 1\,\text{bit}$) can closely be reached only for large $\bar{D}_{\mathsf{C}}$ and in good channel conditions. For terminated (or tailbiting) recursive convolutional codes, this plot would be a flat surface, as $\bar{D}_{\mathsf{C}}$ tends to infinity in the case of recursive codes [KHC06].
Figure A.6 shows the hard decision bit error probability of the extrinsic output of the MAP SISO decoder if perfect *a priori* knowledge is available ($I^{[\text{apri}]} = 1\,\text{bit}$). The maximum mutual information of the EXIT characteristics in Fig. A.2 (see also Table A.1) can be read off in Fig. A.5 using the information in Table A.2. Table A.2 also contains the calculated values of $P_b^{[\text{ext}]}$ and $I^{[\text{ext}]}\big|_{I^{[\text{apri}]}=1}$ for a channel quality of

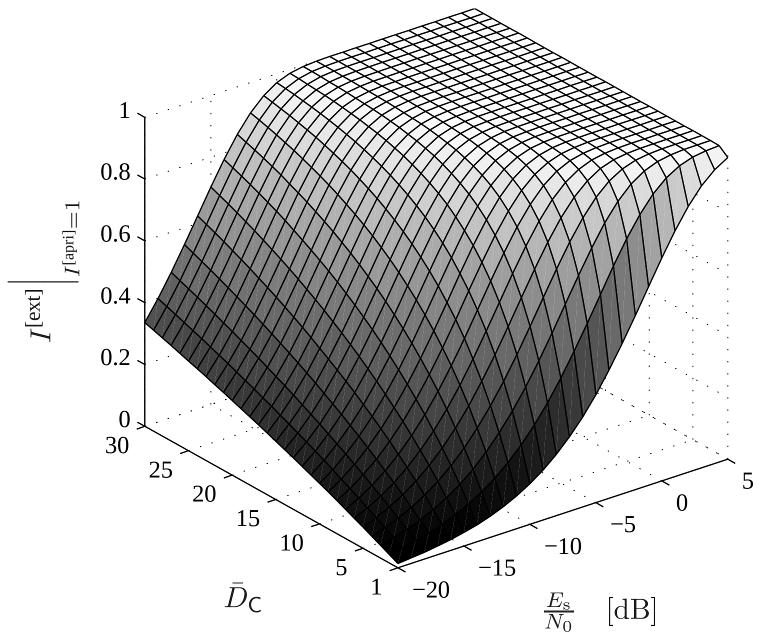


Figure A.5: Mutual information of the extrinsic output as a function of channel quality and $\bar{D}_{\mathsf{C}}$

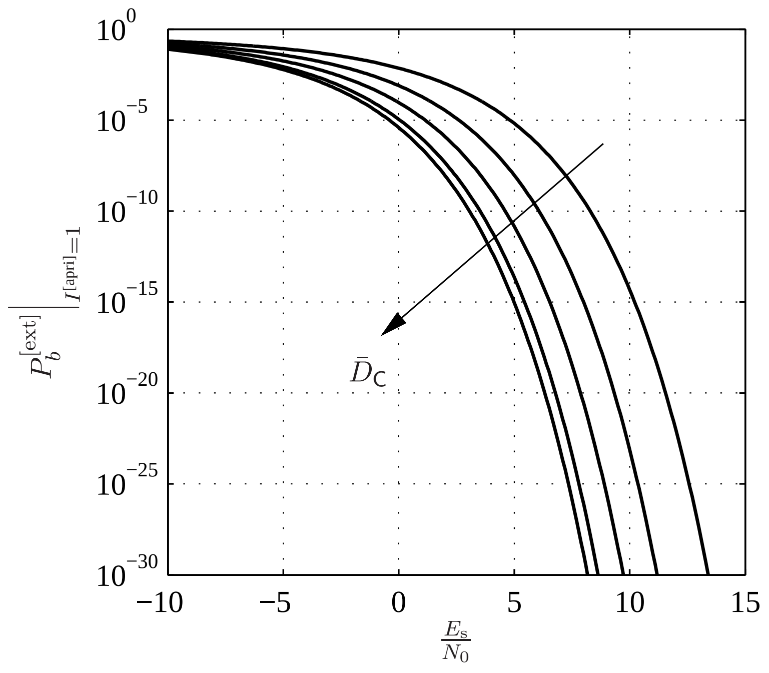


Figure A.6: Hard decision bit error probability of the extrinsic output for $I^{[\text{apri}]} = 1\,\text{bit}$ $(\bar{D}_{\mathsf{C}} = 3, 5, 7, 9, 10)$

$\frac{E_\text{s}}{N_0} = -5\,\text{dB}$. It can be seen that the calculated maximum mutual information almost perfectly matches the measured values of Table A.1. The differences can be explained by numerical inaccuracies during the measurement and/or by the finite histogram resolution.

To conclude, in this appendix we analyzed the behavior of the EXIT characteristics of feed forward convolutional codes. Simulations have shown that the mutual information at the extrinsic output of a MAP SISO decoder for feed forward and non-terminated recursive convolutional codes does not reach $I^{[\text{ext}]} = 1\,\text{bit}$ if perfect *a priori* information is available – a fact that has already been noted in the literature. We give an analytical expression of the attainable mutual information if perfect *a priori* knowledge is available for the case of feed forward convolutional codes. This maximum attainable mutual information solely depends on the channel noise variance and the Hamming weight of the code impulse response. We give an easy explanation of this property using the Trellis representation of the convolutional code.

Table A.2: Impulse response Hamming weights for some selected convolutional codes and numerical results for $E_\text{s}/N_0 = 1/(2\sigma_n^2) = -5\,\text{dB}$

Generator polynomials	memory J	$\bar{D}_{\mathsf{C}}$	$P_b^{[\text{ext}]}$	$I^{[\text{ext}]}\Big\vert_{I^{[\text{apri}]}=1}$
$(3,2)_8$	1	3	0.0842	0.7038
$(7,5)_8$	2	5	0.0377	0.8592
$(17,15)_8$	3	7	0.0177	0.9315
$(17,15,13)_8$	3	10	0.0060	0.9762
$(23,35)_8$	4	7	0.0177	0.9315
$(53,75)_8$	5	9	0.0085	0.9662
$(133,171)_8$	6	10	0.0060	0.9762

Furthermore, we have shown that the extrinsic output of a MAP SISO decoder is Gaussian distributed if the channel noise is Gaussian, the data bits are equiprobable, and the *a priori* information on the data bits is considered to be perfect. Additionally, we have derived an analytical expression of the mutual information attainable if perfect *a priori* knowledge is available as well as an expression of the hard decision bit error rate of the extrinsic output in this case. Finally, an evaluation of different codes and the verification of the theoretical results has been presented.

Bibliography

[3GP] 3GPP. *3GPP TS 25.222 V3.4.0 (2000-09)*. available at http://www.3gpp.org.

[ABCV05] M. Adrat, J. Brauers, T. Clevorn, and P. Vary. “The EXIT-Characteristic of Softbit-Source Decoders”. *IEEE Communications Letters*, 9(6):540–542, June 2005.

[ACBV06] M. Adrat, T. Clevorn, J. Brauers, and P. Vary. “Minimum Terms of Residual Redundancy for Successful Iterative Source-Channel Decoding”. *IEEE Communications Letters*, 10(11):778 – 780, November 2006.

[AKtB02] A. Ashikhmin, G. Kramer, and S. ten Brink. “Extrinsic Information Transfer Functions: A Model and Two Properties”. *Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, USA, March 2002.

[AKtB04] A. Ashikhmin, G. Kramer, and S. ten Brink. “Extrinsic Information Transfer Functions: Model and Erasure Channel Properties”. *IEEE Trans. Inform. Theory*, November 2004.

[AV05] M. Adrat and P. Vary. “Iterative Source-Channel Decoding: Improved System Design Using EXIT Charts”. *EURASIP Jour. Appl. Sig. Proc.*, 205(6):928–941, May 2005.

[AVC05] M. Adrat, P. Vary, and T. Clevorn. “Optimized Bit Rate Allocation for Iterative Source-Channel Decoding and its Extension towards Multi-Mode Transmission”. *14th IST Mob. & Wirel. Comm. Summit*, Dresden, June 2005.

[AVS01] M. Adrat, P. Vary, and J. Spittka. “Iterative Source-Channel Decoder Using Extrinsic Information from Softbit-Source Decoding”. *IEEE ICASSP*, Salt Lake City, USA, May 2001.

[BCJR74] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. “Optimal Decoding of Linear Codes for Minimizing Symol Error Rate”. *IEEE Trans. Inform. Theory*, 20:284–287, March 1974.

[BCK07] E. Bodden, M. Clasen, and J. Kneis. “Arithmetic Coding revealed - A guided tour from theory to praxis”. Technical Report SABLE-TR-2007-5, Sable Research Group, School of Computer Science, McGill University, Montréal, Québec, Canada, May 2007.

[BCW90] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice Hall, Inc., 1990.

[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.*, February 1998.

[BDMP98b] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. “Serial Concatention of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding”. *IEEE Trans. Inform. Theory*, pages 909–921, May 1998.

[BG96] C. Berrou and A. Glavieux. “Near optimum error correcting coding and decoding: Turbo codes”. *IEEE Trans. Commun.*, 44(10):1261–1271, October 1996.

[BH00] R. Bauer and J. Hagenauer. “Symbol by Symbol MAP Decoding of Variable Length Codes”. *Proc. of 3rd International ITG Conference on Source and Channel Coding*, Munich, Germany, January 2000.

[BHG02] J. Barros, J. Hagenauer, and N. Goertz. “Turbo cross decoding of multiple descriptions”. *Proc. of the International Conference on Communications (ICC)*, 2002.

[BM02] S. Benedetto and G. Montorsi. “The new frontier of channel coding: concatenated codes with interleaver and iterative decoding”. *4th International ITG Conference on Source and Channel Coding (SCC)*, Berlin, Germany, January 2002.

[BMDP98] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara. “Soft-Input Soft-Output Modules for the Construction and Distributed Iterative Decoding of Code Networks”. *Europ. Trans. Telecommun. (ETT)*, March 1998.

[CAV06] T. Clevorn, M. Adrat, and P. Vary. “Turbo DeCodulation using Highly Redundant Index Assignments and Multi-Dimensional Mappings”. *Intl. Symposium on Turbo Codes & Related Topics*, Munich, April 2006.

[CBAV05] T. Clevorn, J. Brauers, M. Adrat, and P. Vary. “Turbo DeCodulation: Iterative Combined Demodulation and Source-Channel Decoding”. *IEEE Communications Letters*, 9(9), September 2005.

[CE81] Y. K. Chan and R. W. Edsinger. “A Correlated Random Numbers Generator and Its Use to Estimate False Alarm Rates of Airplane Sensor Failure Detection Algorithms”. *IEEE Trans. Automat. Contr.*, AC-26(3):676–680, June 1981.

[CF02] J. Chen and M. P. C. Fossorier. “Near Optimum Universal Belief Propagation Based Decoding of Low-Density Parity Check Codes”. *IEEE Trans. Commun.*, 50(3), March 2002.

[CHIW98] D. J. Costello, Jr., J. Hagenauer, H. Imai, and S. B. Wicker. “Applications of Error-Control Coding”. *IEEE Trans. Inform. Theory*, 44(6):2531–2560, October 1998.

[Cle06] T. Clevorn. *Turbo DeCodulation: Iterative Joint Source-Channel Decoding and Demodulation*. PhD thesis, RWTH Aachen University, December 2006.

[CSVA06] T. Clevorn, L. Schmalen, P. Vary, and M. Adrat. “On The Optimum Performance Theoretically Attainable for Scalarly Quantized Correlated Sources”. *Proc. of ISITA*, Seoul, Korea, November 2006.

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

[CTB98] G. Caire, G. Taricco, and E. Biglieri. “Bit-Interleaved Coded Modulation”. *IEEE Trans. Inform. Theory*, pages 927–946, May 1998.

[CVA06] T. Clevorn, P. Vary, and M. Adrat. “Iterative Source-Channel Decoding using Short Block Codes”. *IEEE ICASSP*, Toulouse, France, May 2006.

[DB05a] L. Dinoi and S. Benedetto. “Design of Fast Prunable S-Random Interleavers”. *IEEE Trans. Wireless Commun.*, 4(5), September 2005.

[DB05b] L. Dinoi and S. Benedetto. “Variable Size Interleaver Design for Parallel Turbo Decoder Architectures”. *IEEE GLOBECOM*, Dallas, TX, USA, December 2005.

[DB05c] L. Dinoi and S. Benedetto. “Variable Size Interleaver Design for Parallel Turbo Decoder Architectures”. *IEEE Trans. Commun.*, 53(11), November 2005.

[DDP98a] S. Dolinar, D. Divsalar, and F. Pollara. “Code Performance as a Function of Block Size”. *JPL TMO Progress Report 42-133*, May 1998.

[DDP98b] S. Dolinar, D. Divsalar, and F. Pollara. “Turbo Code Performance as a Function of Code Block Size”. *IEEE International Symposium on Information Theory (ISIT)*, Cambridge, MA, USA, August 1998.

[DP95a] D. Divsalar and F. Pollara. “Multiple Turbo Codes for Deep-Space Communications”. *JPL TDA Progress Report 42-121*, pages 66–77, May 1995.

[DP95b] D. Divsalar and F. Pollara. “Turbo Codes for PCS Applications”. *IEEE International Conference on Communications (ICC)*, Seattle, WA, USA, June 1995.

[DPD$^+$95] C. Douillard, A. Picart, P. Didier, M. Jézéquel, C. Berrou, and A. Glavieux. “Iterative Correction of Intersymbol Interference: Turbo-Equalization”. *Europ. Trans. Telecommun. (ETT)*, pages 507–512, September 1995.

[EH99] M. Eroz and A. R. Hammons Jr. “On the Design of Prunable Interleavers for Turbo Codes”. *Proc. of IEEE Vehicular Technology Conference (VTC)*, Houston, TX, USA, July 1999.

[FHV99] T. Fingscheidt, S. Heinen, and P. Vary. “Joint Speech Codec Parameter And Channel Decoding of Parameter Individual Block Codes (PIBC)”. *Speech Coding Proceedings*, 1999.

[Fin98] T. Fingscheidt. *Softbit-Sprachdecodierung in digitalen Mobilfunksystemen*. PhD thesis, RWTH Aachen University, July 1998. (in German).

[Fle07a] FlexCode. “Deliverable D-2.1: Generic Binary Input Soft Output (BISO) Channel Model”. Technical report, European Union, July 2007.

[Fle07b] FlexCode. “Deliverable D-3.1: Ordered List of Real World Service Scenarios”. Technical report, European Union, March 2007.

[Fle08] FlexCode. “Deliverable D-1.1: Baseline Source Coder”. Technical report, European Union, February 2008.

[FSB02] M. Ferrari, F. Scalise, and S. Bellini. “Prunable S-Random Interleavers”. *Proc. of International Conference on Communications (ICC)*, New York City, NY, USA, April 2002.

[FV01] T. Fingscheidt and P. Vary. “Softbit Speech Decoding: A New Approach to Error Concealment”. *IEEE Trans. Speech Audio Processing*, 9(3):240–251, March 2001.

[Gal62] R. G. Gallager. “Low-Density Parity-Check Codes”. *IRE Transactions on Information Theory*, pages 21–28, January 1962.

[Gal63] R. G. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, 1963.

[GFGR01] A. Guyader, E. Fabre, C. Guillemot, and M. Robert. “Joint Source-Channel Turbo Decoding of Entropy-Coded Sources”. *IEEE J. Select. Areas Commun.*, 19(9), September 2001.

[GG04] T. Guionnet and C. Guillemot. “Soft and Joint Source-Channel Decoding of Quasi-Arithmetic Codes”. *EURASIP Journal on Applied Signal Processing*, 2004(3):393–411, March 2004.

[Gor01] N. Gortz. “On the Iterative Approximation of Optimal Joint Source-Channel Decoding”. *IEEE J. Select. Areas Commun.*, 19(9):1662–1670, September 2001.

[Goy01] V. K. Goyal. “Multiple Description Coding: Compression Meets the Network”. *IEEE Signal Processing Mag.*, September 2001.

[GS04] N. Görtz and A. Schaefer. “The Use of EXIT Charts in Iterative Source-Channel Decoding”. *Proc. of EUSIPCO*, Vienna, Austria, September 2004.

[Gör98] N. Görtz. “Joint Source Channel Decoding Using Bit-Reliability Information and Source Statistics”. *Proc. of IEEE International Symposium on Information Theory*, August 1998.

[Hag94] J. Hagenauer. “Soft is Better than Hard”. D. Blahut and D.Costello, editors, *Proc. in Communications, Coding and Cryptology*. Kluwer Publication, May 1994.

[Hag97] J. Hagenauer. “The Turbo Principle - Tutorial Introduction and State of the Art”. *1st International Symposium on Turbo Codes & Related Topics*, Brest, France, September 1997.

[Hag02] J. Hagenauer. “The Turbo Principle in Mobile Communications”. *International Symposium on Information Theory and its Applications (ISITA)*, Xi’an, China, October 2002.

[HLY02] L. Hanzo, T. H. Liew, and B. L. Yeap. *Turbo Coding, Turbo Equalisation asn Space-Time Coding for Transmission over Fading Channels*. Wiley, 2002.

[HOP96] J. Hagenauer, E. Offer, and L. Papke. “Iterative Decoding of Binary Block and Convolutional Codes”. *IEEE Trans. Inform. Theory*, 42(2):429–445, March 1996.

[HV05] S. Heinen and P. Vary. “Source-Optimized Channel Coding for Digital Transmission Channels”. *IEEE Trans. Commun.*, 53(4):592–600, April 2005.

[KHC06] J. Kliewer, A. Huebner, and D. J. Costello, Jr. “On the Achievable Extrinsic Information of Inner Decoders in Serial Concatenation”. *Proc. of ISIT*, Seattle, WA, USA, July 2006.

[KO07] W. B. Kleijn and A. Ozerov. “Rate Distribution between Model and Signal”. *Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 243–246, November 2007.

[LR97] X. Li and J. A. Ritcey. “Bit-Interleaved Coded Modulation with Iterative Decoding”. *IEEE Comm. Lett.*, pages 169–171, November 1997.

[Mac99] D. J. C. MacKay. “Good Error-Correcting Codes Based on Very Sparse Matrices”. *IEEE Trans. Inform. Theory*, pages 399–431, March 1999.

[MN96] D. J. C. MacKay and R. M. Neal. “Near Shannon Limit Performance of Low Density Parity Check Codes”. *Electr. Lett.*, pages 1645–1646, August 1996.

[PKH01] R. Perkert, M. Kaindl, and T. Hindelang. “Iterative Source and Channel Decoding for GSM”. *Proc. of IEEE ICASSP*, May 2001.

[Poo00] H. V. Poor. “Turbo Multiuser Detection: An Overview”. *IEEE 6th International Symposium on Spread-Spectrum Techniques and Applications (ISSSTA)*, Parsippany, NJ, USA, September 2000.

[Pro01] J. G. Proakis. *Digital Communications*, chapter 10. McGraw-Hill, New York, 4th edition, 2001.

[PTVF92] W. Press, S. Teukolsky, W. Vetterding, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[PU02] A. Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw Hill, 2002.

[RU08] T. Richardson and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008. unpublished.

[RVH95] P. Robertson, E. Villebrun, and P. Hoeher. “A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain”. *Proc. IEEE Int. Conf. Commun. (ICC)*, pages 1009–1013, Seattle, USA, 1995.

[SCV07] L. Schmalen, T. Clevorn, and P. Vary. “Iterative Source-Coded Equalization: Turbo Error Concealment for ISI Channels”. *Proc. of IEEE SPAWC*, Helsinki, June 2007.

[SVAC07] L. Schmalen, P. Vary, M. Adrat, and T. Clevorn. “On the EXIT Characteristics of Feed Forward Convolutional Codes”. *Proc. of International Symposium on Information Theory (ISIT)*, Nice, France, June 2007.

[SVCS08] L. Schmalen, P. Vary, T. Clevorn, and B. Schotsch. “Efficient Iterative Source-Channel Decoding Using Irregular Index Assignments”. *Proc. of International ITG Conference on Source and Channel Coding (SCC)*, Ulm, Germany, January 2008.

[tB99] S. ten Brink. “Convergence of Iterative Decoding”. *IEE Electronics Letters*, 35(10):806–808, May 1999.

[tB00a] S. ten Brink. “Design of Serially Concatenated Codes based on Iterative Decoding Convergence”. *3rd International Symposium on Turbo Codes & Related Topics*, Brest, France, September 2000.

[tB00b] S. ten Brink. “Designing Iterative Decoding Schemes with the Extrinsic Information Transfer Chart”. *AEÜ Int. J. Electron. Commun.*, pages 389–398, December 2000.

[tB01a] S. ten Brink. “Convergence Behaviour of Iteratively Decoded Parallel Concatenated Codes”. *IEEE Trans. Commun.*, 49(10):1727–1737, October 2001.

[tB01b] S. ten Brink. “Exploiting the Chain Rule of Mutual Information for the Design of Iterative Decoding Schemes”. *Proc. Allerton Conf. on Commun., Control, and Computing*, Monticello, IL, USA, October 2001.

[TDB07] A. Tarable, L. Dinoi, and S. Benedetto. “Design of Prunable Interleavers for Parallel Turbo Decoder Architectures”. *IEEE Commun. Lett.*, 11(2), February 2007.

[TH02] M. Tüchler and J. Hagenauer. “EXIT Charts of Irregular Codes”. *Proc. of Conf. on Information Sciences and Systems (CISS)*, Princeton University, March 2002.

[Tho07a] R. Thobaben. “A New Transmitter Concept for Iteratively-Decoded Source-Channel Coding Schemes”. *Proc. of IEEE Workshop on Signal Processing Advances in Wireless Communications*, Helsinki, Finland, June 2007.

[Tho07b] R. Thobaben. *Iterative Quellen- und Kanaldecodierung für Codes variabler Länge*. PhD thesis, Christian-Albrechts-Universität zu Kiel, 2007. (in German).

[TK05] R. Thobaben and J. Kliewer. “Low-Complexity Iterative Joint Source-Channel Decoding for Variable-Length Encoded markov Sources”. *IEEE Trans. Commun.*, 53(12), December 2005.

[TKS02] M. Tüchler, R. Koetter, and A. C. Singer. “Turbo Equalization: Principles and New Results”. *IEEE Trans. Commun.*, 50(5):754–767, May 2002.

[TtBH02] M. Tüchler, S. ten Brink, and J. Hagenauer. “Measures for Tracing Convergence of Iterative Decoding Algorithms”. *4th International ITG Conference on Source and Channel Coding (SCC)*, Berlin, Germany, January 2002.

[Tüc04] M. Tüchler. “Design of Serially Concatenated Systems Depending on the Block Length”. *IEEE Trans. Commun.*, 52(2):209–218, February 2004.

[Ung82] G. Ungerboeck. “Channel coding with multilevel/phase signals”. *IEEE Trans. Inform. Theory*, pages 56–67, January 1982.

[Vai93] V. A. Vaishampayan. “Design of Multiple Description Scalar Quantizers”. *IEEE Trans. Inform. Theory*, 39(3), May 1993.

[Vas07] A. Vasilache. “Indexing of Lattice Codevectors Applied to Error Resilient Audio Coding”. *Proc. of 30th AES International Conference*, Saariselkä, Findland, March 2007.

[VM06] P. Vary and R. Martin. *Digital Speech Transmission - Enhancement, Coding and Error Concealment*. John Wiley & Sons, Ltd., 2006.

[VT03] A. Vasilache and I. Tabus. “Robust Indexing of Lattices and Permutation Codes over Binary Symmetric Channels”. *Signal Processing*, 83(7):1467–1486, July 2003.

[VY00] B. Vucetic and J. Yuan. *Turbo Codes: Principles and Applications*. Kluwer Academic Publishers, 2000.

[WK00] D. Wang and H. Kobayashi. “On Design of Interleavers with Practical Size for Turbo Codes”. *Proc. of IEEE International Conference on Communications (ICC)*, New Orleans, LA, USA, June 2000.