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

Project title: FlexCode

Instrument: STREP

Thematic Priority: Information Society Technologies

D2.3: Final Channel Coder

Due date of deliverable: 2008-10-01

Actual submission date: 2008-10-01

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

Organisation name of lead contractor for this deliverable: RWTH Aachen

Revision: 1.2

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

Dipl.-Ing. Thomas Schlien:
Institute of Communication Systems and Data Processing
RWTH Aachen University
Aachen, Germany
schlien@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

Table of Contents

Chapter 1

Overview

1.1 Introduction

The main objective of the *FlexCode* project is to develop *generic* coding technologies that can adapt to the situation at hand instantaneously. The methods avoid application-specific solutions and provide a coding efficiency that is similar to current state-of-the-art algorithms. The technologies address a rapidly increasing diversity of services and an increasingly heterogeneous telecommunications network.
It is useful to briefly review the organizational structure and technical approach of the *FlexCode* project to provide a context for this report. The algorithm development work of *FlexCode* is separated into source coding and channel coding. Work package 1 (WP1) deals with source coding and WP2 with channel coding. WP2 has as objective to develop a generic channel model (described in [Fle07a]) as well as a generic channel coder which will adapt to the source encoder for optimal combined/joint source-channel coding as well as to the different channel types. The specific objectives of WP2 are

- to develop a new generic binary input soft output channel model;
- to develop a flexible channel encoder and a corresponding powerful and efficient channel decoder;
- to create a practical implementation.

This report is deliverable D2.3, which is to describe the final implementation of the *FlexCode* channel coder. In addition, it describes the main specific source-channel coding techniques developed under WP2. Not all of the new techniques are actually used in the implemented channel coder. Some simply provide insight and some turned out not to be applicable for different reasons.
In this chapter we first provide an outline of the *FlexCode* channel coding approach. We then give an overview of the remaining chapters of this report.

1.2 Channel Coding in *FlexCode*

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 bit stream 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 bit stream 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 decoders [RU08]. Therefore, channel codes with iterative receivers have been chosen as a candidate for the *FlexCode* source coder.
On the other hand, if a fixed bit rate is required on the transmission link, *constrained resolution* quantization is employed by the source coder. In this case, uniform quantizers with a limited number of quantization levels are utilized [Fle08a], [Fle08b]. This scheme is perfectly suited for the application of *Iterative Source-Channel Decoding* (ISCD) and therefore, an iterative receiver has been developed during the project.

1.2.1 Architecture

As already outlined during the development of the baseline channel coder [Fle08c], a twofold channel coding scheme is used for the *FlexCode* channel coder. The interface between source and channel coding is not on the bit stream level but immediately after quantization, i.e., the channel coder receives the quantizer indices and side information on the quantization (e.g., type of quantization used, assumed probability density function of the signal, number of quantization levels) and is responsible for generating a bit stream itself. The model parameters are encoded separately by a strong conventional channel code and the transform coefficient are encoded using a joint source-channel coding approach which allows iterative source-channel decoding at the receiver. This twofold scheme is necessary as in order to decode the transform coefficients, the information extracted by the model parameters is required. This was already outlined in [Fle08c]. Therefore a strong interaction between source and channel coding is required at the receiver, where the decoding is performed in several steps:

1. channel decoding of the model parameters;
2. extraction of spectral envelope and transform information from the decoded model parameters;
3. using this information to perform channel decode and, if needed, entropy decoding of the transform coefficients;
4. reconstruct the audio signal using the transform coefficients by performing the inverse transform.

Due to this strong interaction between source and channel coding, both WP1 and WP2 had to work closely together during the development of the source and channel coder.

1.3 Report Overview

In Deliverable D-2.2 [Fle08c], the *FlexCode* baseline channel coder has been introduced. Furthermore, the basics of modern iterative channel coding techniques such as Turbo codes as well as the EXIT chart analysis tools and basic concepts of interleaver design have been introduced. The concept of soft decision source decoding (SDSD) has been depicted and extended for the deployment in iterative Turbo-like decoders leading to *Iterative Source-Channel Decoding* (ISCD). First 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 have been presented. Finally the adaptation of the ISCD scheme to the *FlexCode* source coding concept was given along with a brief outline of how to practically realize the channel encoder and the interface between source and channel coder in *FlexCode*.
In this report, we first describe the channel coder in detail in Chapter 2. The remaining chapters describe the main source-channel coding technologies developed within the *FlexCode* project since Deliverable D2.2 [Fle08c]. In Chapter 3 we present several optimizations dealing with the complexity of the iterative source-channel decoder. These optimizations aim at the reduction of computational complexity mainly at the receiver, however some of them also required modifications of the transmitter. In Chapter 4 the concept of irregular index assignments, already introduced in Deliverable D2.2 [Fle08c], is extended and it is shown how unequal error protection can be easily realized by utilizing irregular index assignments. Furthermore it is shown how the *FlexCode* channel coder can be used as an error-resilient entropy coding scheme by a simple modification of the optimization problem.

Chapter 2

The Final *FlexCode* Channel Coder

2.1 General Channel Coder Description

The channel coding concept has to be adapted to the basic structure of the source encoder. The baseline *FlexCode* source coding concept is described for instance in [BGK$^+$08] and [Fle08a]. For each frame, the source encoder provides a set of parameters which can be grouped into two main parts: model parameters and transform coefficients. The model parameters include for example the LP coefficients and gain factors. Using the model parameters the source encoder determines the quantizer setup for the transform coefficients with the possibility to select between two different quantization modes:

- In the case of constrained resolution (CR) quantization, resulting in a bit stream of fixed rate, the source encoder determines the bit allocation of the transform coefficients, i.e., the number of quantization levels to be used for the considered parameter.
- In the case of constrained entropy (CE) quantization, resulting in a bit stream of variable rate, the source encoder uses the model parameters to determine the distribution of the transform coefficients and the step size of the uniform quantizer. Using this information, an entropy coder (for example an arithmetic coder) can efficiently generate a compressed bit stream.

As a result of this source coding concept, it has been found that it is not feasible to perform 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 step sizes. Therefore, we have proposed to utilize a separate transmission of the model parameters and the transform coefficients in [Fle08c]. The resulting split transmission structure which defines the *FlexCode* baseline channel coder is depicted in Fig. 2.1.
The *FlexCode* source encoder outputs model parameters and transform coefficients which are independently coded in two branches. In order to encode the transform coefficients, side information resulting from the model parameters is required. This side information can be for instance probability density function approximations of the different transform coefficients or quantizer reproduction levels used during quantization. After separately encoding both model parameters and transform coefficients, the resulting bit streams are multiplexed to form a single packet and then transmitted over the *FlexCode* channel model [Fle07a]. At the receiver, this bit stream is demultiplexed and then first the model parameters are channel decoded. Additionally a *Bad Frame Indicator* (BFI) is computed by means of error detection. Such a BFI can be used for instance at the source decoder to perform *frame erasure concealment*. Using the model parameters the source decoder computes the side information required by the transform coefficient channel decoder. This interaction between source and channel coder is detailed in Fig. 2.2. Without this side information it is not possible to reconstruct the transform coefficients from the

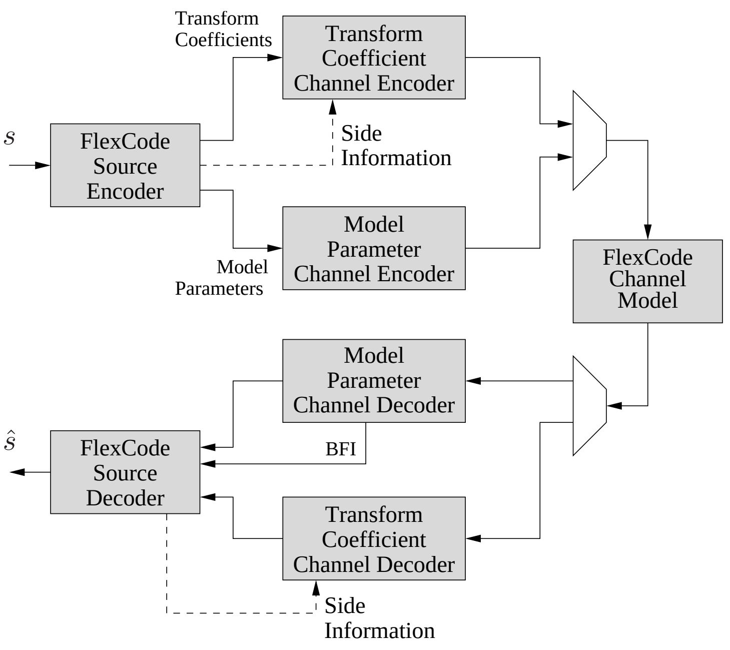


Figure 2.1: Block diagram of the general *FlexCode* channel coding approach

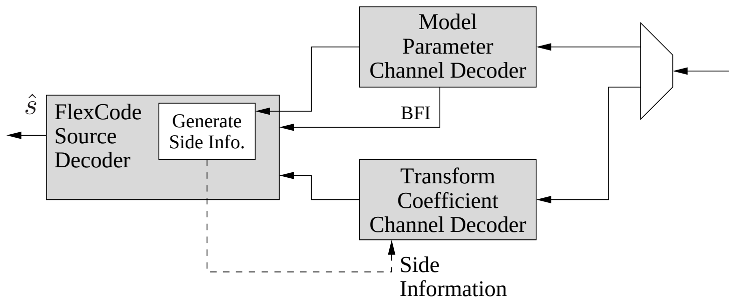


Figure 2.2: Generation of side information for decoding the transform coefficients in the *FlexCode* decoder

bit stream as the number of bits utilized for each coefficient (in the case of constrained resolution quantization) or the probability density functions required by the arithmetic decoder (in the case of constrained entropy quantization) are not available. Therefore the BFI is computed from the model parameters as an erroneously decoded model results in a complete decoding failure of the remainder of the packet. On the other hand, if the transform coefficients cannot be recovered, a signal with acceptable quality can be recovered as the spectral envelope and the gains are available. If the transform coefficients are wrong and the model is decoded correctly, the output of the source decoder is noise colored with the spectral envelope of the original signal.
In order to exploit the advantages of either quantization method, the *FlexCode* WP1 has decided that the source encoder can utilize either quantization method. However, this means that the source coding platform may differ in some elements depending on the utilized quantizer.
In WP1, it has furthermore been decided that two source coding versions are developed jointly [Fle08b]:

1. the version denoted as KLT, which uses a *Karhunen-Loeve transform* (KLT) and constrained entropy (CE, resulting in a bit stream of variable length) or constrained resolution (CR, resulting in a bit stream of fixed length) quantization. Additionally, multiple description coding can be used in

the case that packet losses can occur on the transmission link.

2. the low-complexity version denoted as MDCT, which uses an overlapping *modified discrete cosine transform* (MDCT) and constrained resolution (CR) quantization, resulting in a bit stream of fixed rate.

The model parameter channel encoder is identical for both versions of the source coder, however, the amount of model parameters might be subject to changes in the different versions. The model channel encoder is detailed in Section 2.2. The different quantization methods pose interesting problems for the design of a flexible, generic channel coder enabling iterative source-channel decoding. The channel coding for the constrained entropy version of the source coder will be described in Section 2.3.2, while the channel coding for the constrained resolution quantizer will be outlined in Section 2.3.3.

2.2 Channel Encoding of the Model

As mentioned above, the *FlexCode* channel coder uses a separate transmission chain for the model parameters. This channel encoding and decoding chain for the model parameters is depicted in Fig. 2.3. The model parameters are quantized in all operating modes of the source coder using constrained resolution quantization. After grouping a bit stream is generated using a simple natural binary bit mapping.
The number of model parameters and their bit allocation can vary with the source codec setup. The model parameters utilized in the two different setups are summarized in Table 2.1. For details we refer the reader to [Fle08b]. The model parameter output of the source encoder consists of two informations: the vector of quantizer indices $\mathbf{i}^{\text{[Model]}} = (i_1^{\text{[Model]}}, i_2^{\text{[Model]}}, \ldots, i_{N_M}^{\text{[Model]}})^T$ as well as the number of quantizer reproduction levels $L_\ell^{\text{[Model]}}$, $\ell = 1, \ldots, N_M$ utilized. The bit stream generator then generates a bit stream consisting of $N_B^{\text{[Model]}} = \sum_{\ell=1}^{N_M} \log_2 L_\ell^{\text{[Model]}}$ bits. Note that the number of bits remains constant as constrained resolution quantized is used. The generation of the bit stream is then performed using either a natural binary or a gray bit mapping. The natural binary mapping simply maps the binary representation of the quantizer index $i^{\text{[Model]}}$ to the bit pattern $\mathbf{x}^{\text{[Model]}}$, for instance an index $i_1^{\text{[Model]}} = 5$ (with $L_i^{\text{[Model]}} = 16$) is mapped to $\mathbf{x}_1^{\text{[Model]}} = (0, 1, 0, 1)^T$. By definition, the MSB is the first entry of the bit pattern vector. On the other hand, a gray mapping [JN84] can be utilized. If a gray mapping is employed, the bit mappings of neighboring quantizer reproduction levels only differ by a single bit. Therefore, if a single bit error occurs, the gray mapping has the advantage that the distortion in the reconstructed signal is small.

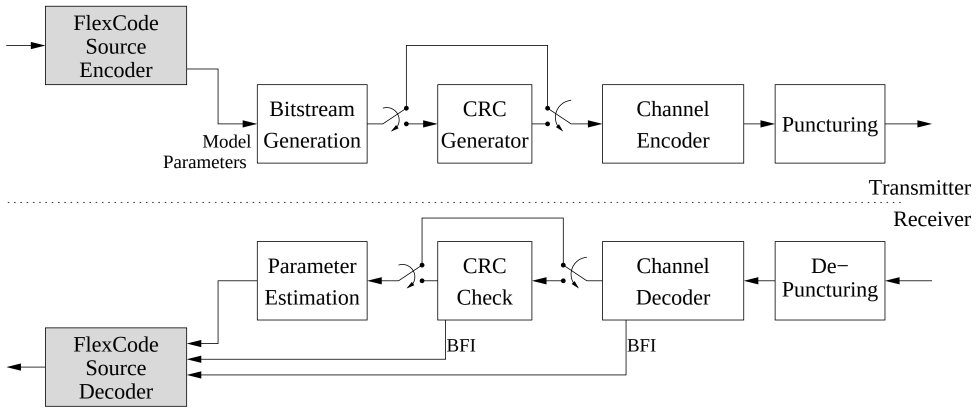


Figure 2.3: Block diagram of the *FlexCode* model parameter channel coder

KLT version	MDCT version
GMM index	GMM index
16 LSF indices	16 LSF indices
5 gains (every subframe)	1 gain (every second subframe)
Pitch index	2 BWE modes for multi-mode quantization (every subframe)
5 pitch refine (every subframe)	
5 pitch decay (every subframe)	

Table 2.1: Model parameters to be transmitted in the KLT and MDCT codec setup

After grouping, the bit stream can optionally be protected by a CRC code [LC03]. The generator polynomial of the CRC can be freely chosen, however, in the standard *FlexCode* setup, we selected one of the following generator polynomials

$$G_{\text{CRC},1}(z) = 1 + z + z^3 \tag{2.1}$$

$$G_{\text{CRC},2}(z) = 1 + z^4 + z^5 + z^6 + z^8 \, . \tag{2.2}$$

The generator polynomial $G_{\text{CRC},1}(z)$ adds 3 additional CRC bits to bit stream and offers only poor error detection capabilities. Therefore it is only chosen if the channel quality is such that after channel decoding only few bit errors are expected. On the other hand, if $G_{\text{CRC},2}(z)$ is used, 8 additional bits are used for error detection. This second code exhibits much better error detection capabilities.
After the optional CRC check, the bit stream is encoded using a strong conventional channel code. This channel code could be for instance an iteratively decodable code such as a Turbo code or an LDPC code [Mac99], [MN96]. The bit rate for transmitting the model parameters is rather small and more or less fixed (around 5 kbit/s, see [KO07]). As LDPC codes and Turbo codes might show a considerably high error floor due to the small block size (and interleaver), 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 efficiently corrected by the Reed-Solomon decoder. By puncturing the convolutional code, the rate and the robustness requirements can be efficiently adjusted.
Although several possibilities have been studied throughout the project for encoding the model bit stream, in the final version we selected LDPC codes for several reasons:

- The complexity of the LDPC decoder, based on belief propagation[MN96] scales with the channel quality. In good channel conditions, a small number of iterations is sufficient to successfully decode the block while the number of necessary iterations increases if the channel quality becomes worse.

- The LDPC code offers very strong built in error detection capabilities. After a fixed maximum amount of allowed iterations, the hard decision output of the channel decoder is multiplied with the parity check matrix (i.e., the parity check equations are evaluated) and if all parity check equations are satisfied, the model parameters are declared error-free. If at least one parity check equation is not satisfied, the block is declared erroneous and the *Bad Frame Indicator* (BFI) is set. In this case, the source decoder can take appropriate measures for frame erasure concealment.

Due to these advantages, LDPC codes are selected as channel codes of choice for the model parameters. Due to the second error-detection properties, additional error detection by means of CRC codes is no longer necessary. Therefore, the CRC check in Fig. 2.3 can be bypassed. Three different LDPC codes can be utilized in the final *FlexCode* channel code. During operation, the coder can select either of these three depending on the assumed transmission quality. The 3 codes are all regular LDPC codes constructed using MacKay's algorithm [Mac99]

- a regular $(160, 110)$ LDPC code $\mathcal{C}_1$ generating 50 parity bits
- a regular $(210, 110)$ LDPC code $\mathcal{C}_2$ generating 100 parity bits
- a regular $(410, 110)$ LDPC code $\mathcal{C}_3$ generating 300 parity bits

As the number of input bits may vary from one codec configuration to the other, the code has to be shortened. Therefore, the input bit stream of length $N_B^{\text{[Model]}}$ (with $N_B^{\text{[Model]}} \leq 110\,\text{bit}$) is padded with zeros, encoded, and after encoding the padded zeros are removed again prior to transmission. At the receiver, the knowledge that part of the input bit stream have been zeros can be used as *a priori* information in the belief propagation decoder. Therefore, the rates of the three LDPC codes are $\frac{N_B^{\text{[Model]}}}{N_B^{\text{[Model]}}+50}$, $\frac{N_B^{\text{[Model]}}}{N_B^{\text{[Model]}}+100}$, or $\frac{N_B^{\text{[Model]}}}{N_B^{\text{[Model]}}+300}$, respectively. Additionally, the codes can be punctured for an code rate adaptation with finer granularity. This is achieved by not transmitting part of the parity bits. At the receiver these parity bits are then considered to be erasures. The puncturing is performed such that at least two non-erased bit nodes are connected to each check node in the Tanner graph.
The decoder of the model parameter transmission chain performs the inverse operation of the encoder and reconstructs the model parameters which are then used to generate the necessary side information for the transform coefficients (see Fig. 2.2).

2.3 Channel Encoding of the Transform Coefficients

The transform coefficients on the other hand are encoded using an iterative source-channel coding system. For a detailed description and implementational details of this joint source-channel coding approach with iterative decoding, we refer the reader to the literature, e.g., [ACS08]. As the approach depends on the type of quantization (constrained resolution or constrained entropy) two cases and two different coding schemes have to be considered. The details for both cases will be given in Sections 2.3.2 and 2.3.3. During the development of both coding systems, the compatibility and interoperability of both platforms was a required constraint. Therefore, most modules can be be used in both versions and the concept remains more or less the same.
The basic concept is identical for both quantization modes: a bit stream is generated and a block encoder adds a certain amount of artificial redundancy (depending on the overall coding rate) to the bit stream. This bit stream is interleaved using the interleaver presented in Sec. 2.3.1 and then encoded by a convolutional encoder. At the receiver, a MAP decoder and an SDSD (which may exploit the residual redundancy of the transform coefficients, if available) iteratively exchange extrinsic information. After a certain number of iterations have been carried out, the transform coefficients are estimated using the MAP rule.
The information about the bit allocation (in the case of constrained resolution quantization) or the quantizer step sizes and parameter distribution (in the case of constrained entropy quantization) is derived by the *FlexCode* source decoder from the model parameters which are decoded first. This information is then used by the soft decision source decoder in the iterative source-channel decoding process.

2.3.1 Flexible Interleavers

Due to the fact that the size of the bit stream to be encoded and transmitted can vary from frame to frame, flexible interleavers that can automatically interleave and deinterleave frames of different lengths

are required. Furthermore, the interleavers have to fulfill several properties, like the S-property, in order to be suitable for iterative decoding. For details, we refer the reader to Chapter 2.5 in [Fle08c] where a detailed description of S-random interleavers is given.
The flexibility of the interleaver is achieved by pruning an S-random interleaver according to the technique presented in [FSB02]. The generation of such an interleaver is also highlighted in [Fle08c]. If the size of the smallest possible interleaver and the largest possible interleaver differ by a factor larger than two, it can be beneficial to generate several prunable interleavers and to select the according one. This has to be done as the S parameter has to be fixed for the minimum size of the interleaver and the performance degrades with small interleaver size, respectively with small S.

2.3.2 Constrained Entropy (CE) case

In the case of CE quantization, an arithmetic encoder generates a variable-length bit stream using the statistical information on the transform coefficients. Usually it is assumed that the transform coefficients are Gaussian distributed. The source encoder then estimates the variance of the Gaussian distribution. Using this variance and the quantizer step size, probabilities of occurrence of the different quantized transform coefficient indices can be determined. These are used in the arithmetic coder to perform the entropy coding. Details have already been given in [Fle08c].
The basic block diagram of the *FlexCode* channel coder with constrained entropy quantization is given in Fig. 2.4. For the transmission of the model parameters, an LDPC code is selected. The description of the model parameter transmission chain has already been outlined in Section 2.2. The transform coefficients are arithmetically encoded using the side information from the source encoder and the resulting bit stream (of variable length) is then encoded by a serially concatenated product code: after a first encoding by the *outer channel encoder*, puncturing, and interleaving, the *inner channel encoder* performs a second channel coding step. This inner channel code is a recursive convolutional code. Two operation modes can be selected:

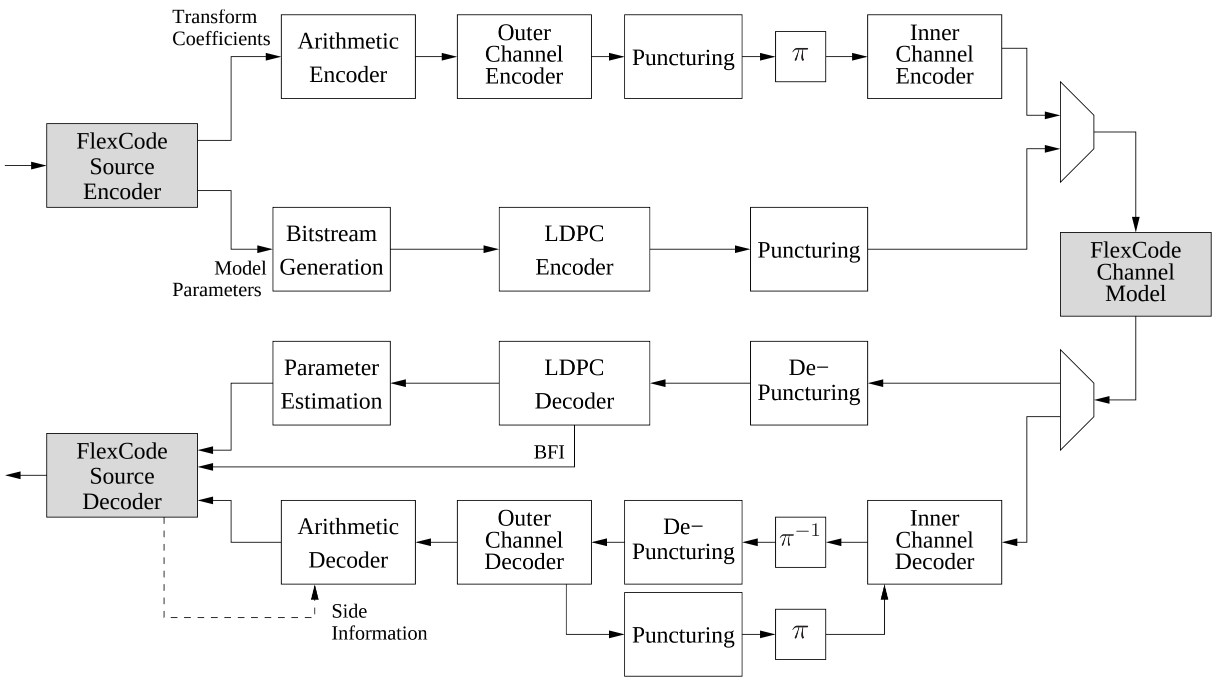


Figure 2.4: Block diagram of the *FlexCode* channel coder with constrained entropy quantization

- The inner code consists of a recursive convolutional code code of constraint length $L + 1 = 4$, rate $r_{CI} = 1$ and generator polynomial $G_I(z) = \frac{1}{1+z^{-1}+z^{-2}+z^{-3}}$. The outer code is a code which partitions the arithmetically coded bit stream into groups of several bits. A small block code which adds one or several parity bits depending on the number of available redundancy bits, is then assigned to these groups. The assignment of block codes to the groups of bits can be optimized using the concept of irregular codes and index assignments [TH02a], [SVCS08], [Fle08c] (see also Section 4). At the receiver, the outer channel decoder consists then of an SDSD which does not exploit any statistical properties as the bits after arithmetic coding are assumed to be equiprobable. The SDSD reduces in this case to a MAP decoding of the single block codes. For an optimization of the selection of the outer block codes and details of the implementation, we refer the reader to Section 4.4.

- In the second setup, the inner code is a convolutional code of constraint length $L + 1 = 2$, rate $r_{CI} = 1$ and generator polynomial $G_I(z) = \frac{1}{1+z^{-1}}$. The outer code is a rate < 1 convolutional code. The overall channel coding scheme then reduces to a convolutional product code, also commonly referred to as *serial Turbo code* [BM02], [BDMP98a], [BDMP98b]. Both channel decoders at the receiver are MAP decoders according to [BCJR74], [RVH95].

The interleaver is a flexible interleaver as described in Section 2.3.1. The reason for placing the puncturer in between both channel encoders and not prior to the transmission over the channel (as would be the conventional way to place it) will be detailed in Section 2.3.2.4.
The decoder structure is given in the bottom part of Fig. 2.4. First the model parameters are decoded and using these model parameters, the source decoder extracts the side information necessary for arithmetic decoding. On the other hand, the transform coefficients are decoded using the iterative decoder based on the Turbo principle. Both model parameter and transform coefficient channel decoding can be executed in parallel as they do net rely upon each other. After a certain number of iterations, the bit stream generated by the transform coefficient decoder is fed to the arithmetic decoder which reconstructs the transform coefficients. The arithmetic encoder and decoder are described in detail in [Fle08c] and in Section 2.3.2.1.
It shall be noted that by a proper selection of the inner and outer codes, the channel coding part can be completely disabled and the arithmetically coded bit stream is directly fed to the channel. This is especially important if no bit errors are expected on the transmission link, e.g., in pure packet oriented networks.
In the following Sections, several components of the *FlexCode* source- and channel coding system are described in detail.

2.3.2.1 Arithmetic Coding for *FlexCode*

The *FlexCode* source encoder outputs quantized indices i_ℓ which are entropy coded using an arithmetic coder [BCK07], [BCW90]. The arithmetic coder generates the bit stream and is considered to be part of the channel coder in *FlexCode*. 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 (after KLT) to be Gaussian distributed. The variances of the distribution can be computed using the model parameters (e.g., spectral envelope) and the knowledge of the utilized transform.

The quantizer that is used in the *FlexCode* constrained entropy source encoder is a uniform scalar quantizer. The quantizer outputs indices $i_\ell \in \mathbb{Z}$, $\ell \in \{1, \dots, N_T\}$ with N_T being the amount of transform coefficients per frame. The quantizer maps all elements in the interval $[s_\ell(i_\ell - \frac{1}{2}); s_\ell(i_\ell + \frac{1}{2}))$ to the index i_ℓ. The quality of the quantizer (and thus also the overall source coding rate) is controlled by the quantizer step size s_ℓ. The step size is determined by the source encoder (and decoder) and then passed to the channel encoder, i.e., to the arithmetic encoder. In order to simplify the calculation of the probability intervals, we define the effective step size $s_{\ell,\text{eff}} \doteq \frac{s_\ell}{\sigma_\ell}$ with σ_ℓ being the standard deviation of the ℓ-th transform coefficient. This effective step size is the step size normalized to a unit-variance Gaussian probability density function (pdf). Using the effective step size we can calculate the probabilities of occurrence of the different transform coefficients by evaluating

$$P(i_\ell|s_{\ell,\text{eff}}) = \frac{1}{2}\left(\text{erfc}\left(\frac{s_{\ell,\text{eff}}\left(i_\ell - \frac{1}{2}\right)}{\sqrt{2}}\right) - \text{erfc}\left(\frac{s_{\ell,\text{eff}}\left(i_\ell + \frac{1}{2}\right)}{\sqrt{2}}\right)\right). \tag{2.3}$$

The arithmetic encoder, 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 bit stream 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]. In constrained entropy quantization, theoretically an infinite number of quantized indices can occur. For this reason, the cumulative distribution function (cdf) is used for the computation of the intervals. For a Gaussian distribution with unit variance and zero mean, the cdf is given by

$$F_n(a) = \int_{-\infty}^{a} p_n(\zeta)\mathrm{d}\zeta = 1 - \frac{1}{2}\,\text{erfc}\left(\frac{a}{\sqrt{2}}\right). \tag{2.4}$$

This cdf can be stored in a lookup table (LUT) in order to facilitate the computation. During encoding, the arithmetic encoder selects the lower bound of the selected probability interval to be $F_n\left(s_{\ell,\text{eff}}(k)\left(i_\ell - \frac{1}{2}\right)\right)$ and the upper bound to be $F_n\left(s_{\ell,\text{eff}}(k)\left(i_\ell + \frac{1}{2}\right)\right)$. Using these bounds the probability intervals can be refined during arithmetic encoding. At the decoder, the encoding procedure is reverted and from the bit stream, the indices $\hat{i}_\ell$ are recovered. The arithmetic decoder has knowledge of the variances (from the decoded model parameters) and can then search for the appropriate indices. The arithmetic decoder starts the search by trying the intervals that have the larges probability, i.e., $0, \pm 1, \pm 2, \dots$. This reduces the decoding complexity.

2.3.2.2 Error Detection in Arithmetic Coding

As the arithmetic is very sensitive to transmission errors, and even a single bit error may cause the complete frame to be erroneous (see also Section 2.3.2.1 and 2.3.2.3), error detection for the transform coefficients may also be desirable. In the case an error has been detected, the *FlexCode* source decoder can take appropriate error concealment measures. An elegant way to incorporate error detection in arithmetic decoding is the use of a forbidden interval, presented in [KCR98]. Instead of using the complete probability interval $[0; 1)$ for encoding the symbols, only a portion of length $1 - \epsilon$, with $0 \leq \epsilon < 1$ is used for encoding. The other portion, of length ϵ is called *forbidden interval* and is never used during encoding. If the decoder happens to operate in the forbidden interval, a transmission error must have occurred previously and the complete block of data can be marked as erroneous. However, the use of a forbidden interval reduces the utilizable probability interval which results in a frame overhead. According to [KCR98], this rate overhead is $-\log_2(1 - \epsilon)$ per encoded symbol. It has been found that acceptable error detection capabilities in the *FlexCode* scenario are achieved with $\epsilon > 0.1$. If for example 320 transform coefficients (a typical value) shall be encoded in a frame, additional 75 bits (i.e, 3.75 kbps) have to be used for error detection with $\epsilon = 0.15$. In the final version of *FlexCode* however, ϵ has been set to zero, as it is advantageous to use the additional available bit rate for error correction.

2.3.2.3 Reordering of Transform Coefficients

It is well known that a single bit error can cause the arithmetic decoder to generate a completely wrong output sequence (error propagation). However, as the arithmetic decoder, implemented according to [BCK07], operates sequentially, the output sequence is correct up to the point where the bit error occurs. Therefore, if for example after channel decoding, there is a single bit error in the middle of the (decoded) bit stream, the first half of the transform coefficients is most likely decoded correctly while the second half is erroneous. As iterative decoders have a noticeable error-floor, single bit errors may occur after channel decoding in bad channel conditions. Therefore it is advantageous to place the most important transform coefficients at the beginning of the frame resulting in *unequal error protection* (UEP).
The reordering strategy in *FlexCode* is to order the transform coefficients according to the variances of their pdf in descending order, i.e., transform coefficients having a pdf with a high variance are placed at the beginning of the block. The reason for this is that if the variance is large, the probability distribution approaches a uniform distribution and the probability intervals in the arithmetic decoder are small and of similar sizes. Therefore, it is essential that the arithmetic decoder picks the right interval. On the other hand, if the variance is very small, the pdf becomes Dirac-like and the quantization index at zero has a large probability interval. Those coefficients are placed at the end of the block because the probability that the arithmetic decoder picks accidentally this interval (if a bit error has occurred beforehand) is quite high and therefore the coefficient is likely decoded correctly.
A simulation example shall prove the advantages of the transform coefficient reordering. It is assumed in the example that all the model parameters are decoded correctly (they are transmitted over an error-free side channel). The transform coefficients are transmitted using the *FlexCode* channel coder over a simple AWGN channel using a simple decoder setup with only 2 iterations. The MOS scores of the system with and without reordering are depicted in Fig. 2.5 over the channel quality E_s/N_0. It can be seen that by

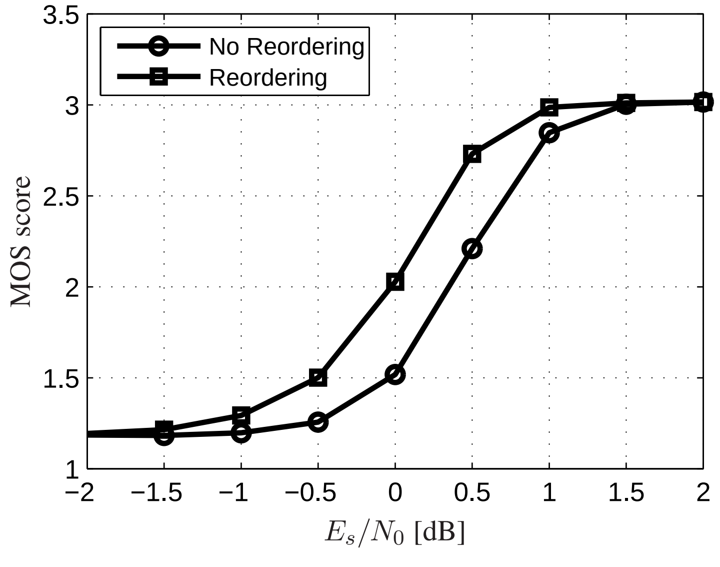


Figure 2.5: Influence of the reordering of transform coefficients on the subjective quality

the simple measure of reordering the transform coefficients, the perceived quality can be dramatically increased if the channel starts to become bad. Note that a preliminary codec version with low overall quality has been used for recoding this plot, however, the effect of reordering will also be visible in the higher quality versions as the reordering does not modify the quality in error-free conditions.

2.3.2.4 Rate Adaptation

The rate adaptation is achieved by puncturing, i.e., not transmitting some of the bits and assuming at the receiver that those bits have been erased during transmission [Hag]. There exist basically two possibilities for placing the puncturing unit for rate adaptation. The first straightforward possibility would be to place the unit after the channel coding, prior to modulation and transmission (i.e., immediately behind the block *Inner Channel Encoder* in Fig. 2.4). However, this causes problems in the present setup where the length of the arithmetically coded bit stream may vary from frame to frame and is not known beforehand at the receiver. The receiver only gets the size of the complete frame after channel coding. However, in order to apply puncturing and depuncturing (filling up the received bit stream with erasures at the receiver) the size has to be known beforehand. A small example shows the problems that can occur due to now knowing the size of the original input frame. Imagine a rate $1/2$ convolutional code that is punctured with the following puncturing matrix

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} . \tag{2.5}$$

Each line of the the puncturing matrix corresponds to one output of the convolutional encoder and each column corresponds to a time step. The puncturing matrix will be periodically repeated of more bits than the width of the puncturing matrix are encoded. The convolutional encoder of rate $1/2$ encodes a vector $\mathbf{x} = (x_1, x_2, x_3, x_4)$ into a vector $\mathbf{y} = (y_1^{(1)}, y_1^{(2)}, y_2^{(1)}, y_2^{(2)}, y_3^{(1)}, y_3^{(2)}, y_4^{(1)}, y_4^{(2)})$. After puncturing with $\mathbf{P}$, the vector $\mathbf{y}_P = (y_1^{(1)}, y_3^{(2)})$ is obtained. At the receiver, erasures are placed at the punctured positions priori to decoding. If the length of the vector $\mathbf{y}$ is unknown it is impossible to construct a vector with the same length, as all vector $\mathbf{y}$ of length $6, 7, 8$, or 9 are punctured to a length 2 vector. As the output of the arithmetic encoder differs in length from frame to frame, conventional puncturing cannot be applied as the channel decoder will be unable to reconstruct the original length of the input bit stream to the arithmetic decoder. Therefore, an unambiguous puncturing algorithm is required that allows to reconstruct the bit stream of correct length at the receiver. Instead of transmitting and encoding the length of the arithmetically coded bit stream, we exploit a property of the arithmetic decoder and do not aim at recovering a bit stream of the same length but a bit stream that will yield the same result after arithmetic decoding.
As the output of the arithmetic encoder is the binary representation of a number between 0 and 1, additional zeros can be added to the arithmetically encoded bit stream without changing the information. These additional zeros have no influence on the decoder and can thus be used to realize an unambiguous puncturing scheme. Therefore the channel encoder adds additional zeros to the output of the arithmetic encoder until after encoding and puncturing the last entry utilized in the puncturing matrix is a "1". In the example presented above, we have to add a single zero to $\mathbf{x}$ resulting in $\mathbf{x}' = ((x_1, x_2, x_3, x_4, 0)$ which is encoded to $\mathbf{y}' = (y_1^{(1)}, y_1^{(2)}, y_2^{(1)}, y_2^{(2)}, y_3^{(1)}, y_3^{(2)}, y_4^{(1)}, y_4^{(2)}, y_5^{(1)}, y_5^{(2)})$. The decoder can now unambiguously reconstruct the length of the original bit stream as the puncturing matrix is known and the rule that the last utilized entry is a "1" has been applied at the receiver.
The puncturer has been placed after the outer channel encoder as the addition of zeros at the input bit stream is then simplified. This somewhat unconventional placement of the punturer has to be considered by the decoder to by placing puncturer and depuncturer within the iterative loop. However, there is no considerable performance loss by placing the puncturer at this position as shown within the project.

2.3.2.5 Multiple Description Coding

In packet oriented transmission scenarios, packet erasures occur frequently. The main tool for combating packet erasures in *FlexCode* are multiple description codes [Goy01], [Vai93]. Multiple description codes split the signal into several description which are transmitted separately over the transmission link. If all the descriptions are received the signal can be reconstructed with high quality. If some of the descriptions are missing but if at least one description is received, the signal can be reconstructed with inferior quality. Note that in contrast to hierarchical coding (as used, e.g., in the ITU-T G729.1 [IT07]) any of the descriptions suffices to reconstruct the signal.

Due to the special structure of the *FlexCode* source coder, the model is absolutely necessary to reconstruct the information for decoding the transform coefficients. The model also needs to be available in full quality at the receiver. Therefore, as the model is absolutely necessary, it has to be transmitted in each description such that the model can be fully reconstructed even if only one description is received. Due to this fact, it has been decided to use only two descriptions in *FlexCode* in order to minimize the rate overhead by transmitting the model in each description [Fle08b], [ZKK09], [KK09].

The block diagram of the channel encoder with *multiple description coding* (MDC) is depicted in Fig. 2.6. The *FlexCode* source encoder emits the model which is encoded using the same blocks as in Fig. 2.3 and two descriptions of the model parameters. Both descriptions are independently encoded using an arithmetic encoder followed by the serially concatenated channel coder known from Fig. 2.4. The channel encoded model bits are attached to each descriptions and both descriptions are transmitted independently over the *FlexCode* channel model.

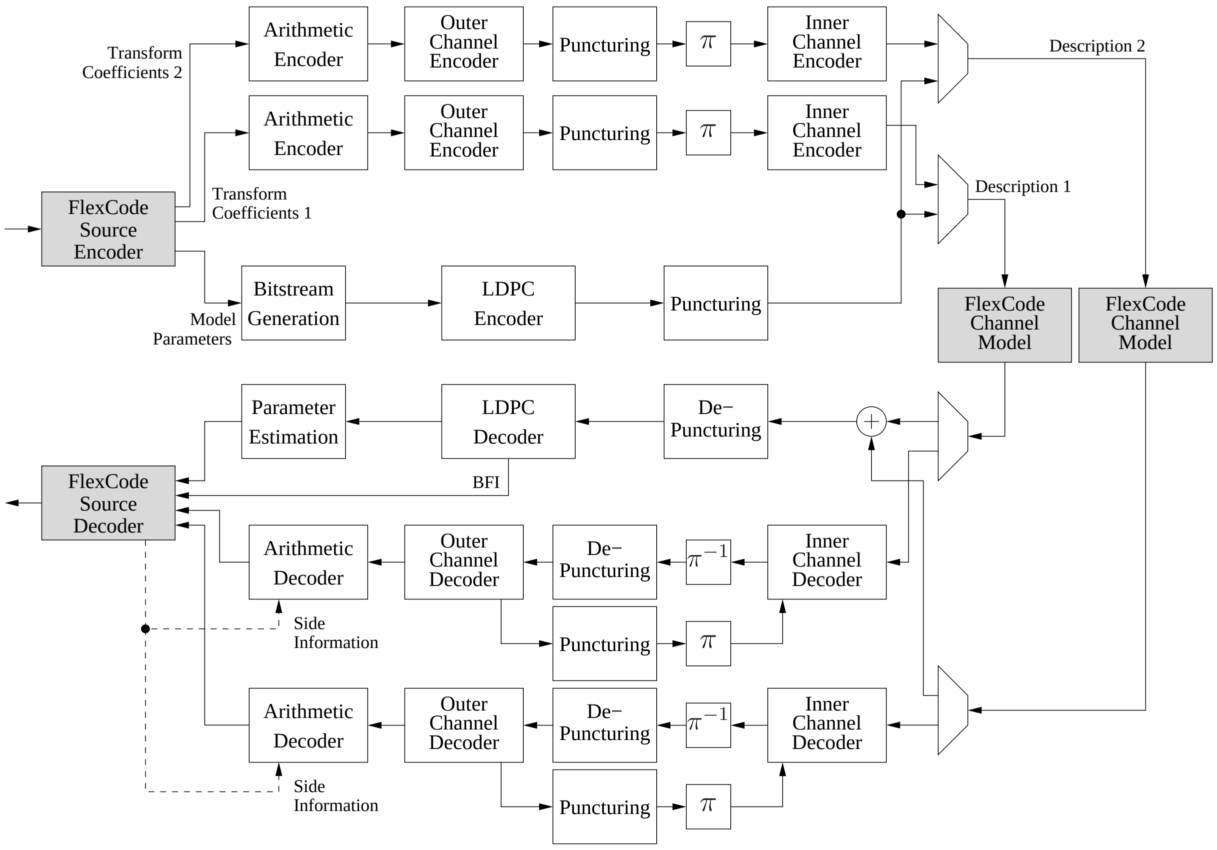


Figure 2.6: Block diagram of the *FlexCode* channel coder with constrained entropy quantization, multiple description case

As the model is included in both descriptions, the principle of information combining [LHHH05], or diversity transmission [LC03], respectively, can be applied. In the L-value domain, information combining is achieved by adding the received L-values. This can be seen in the bottom part of Fig. 2.6 where the received L-values (if available) of the model parameters are summed up prior to channel decoding. If the description has been received, the transform coefficients of the respective description are channel decoded and using the information from the model parameters, arithmetic decoding can be applied. The *FlexCode* source decoder reconstructs the signal using either the decoded indices from one or both descriptions. If both descriptions are lost on the transmission link, frame erasure concealment has to be performed in the source decoder.

2.3.2.6 Turbo Source Coding

It has been shown in [Tho07], [TSV08] that fixed length coding can achieve similar performance as variable length coding by reducing the computational complexity if iterative decoding is employed at the receiver. In [SV09] it has been shown how the parameter individual block codes can be optimized in an ISCD system that yields error-resilient near-lossless source compression. High compression ratios can be achieved with the system proposed in [SV09]. For this reason, it has been studied if the Turbo source coding scheme can also be employed in the final *FlexCode* channel coder, as arithmetic coding is sensitive to transmission errors and requires strong channel coding whereas the Turbo source coding approach can inherently cope with errors on the transmission channel. For details, we refer to Section 4.5 which gives an application example and optimization guidelines for Turbo source coding based on ISCD. The block diagram for the *FlexCode* channel coder for constrained entropy quantization using Turbo source coding is depicted in Fig. 2.7. The model parameters are transmitted using the same channel coding chain as already given in Fig. 2.4. The only modification is the transmission of the transform coefficients. The arithmetic decoder is not used in this setup, and instead, the quantized indices are truncated and a parameter individual block code is assigned to each transform coefficients. For details,

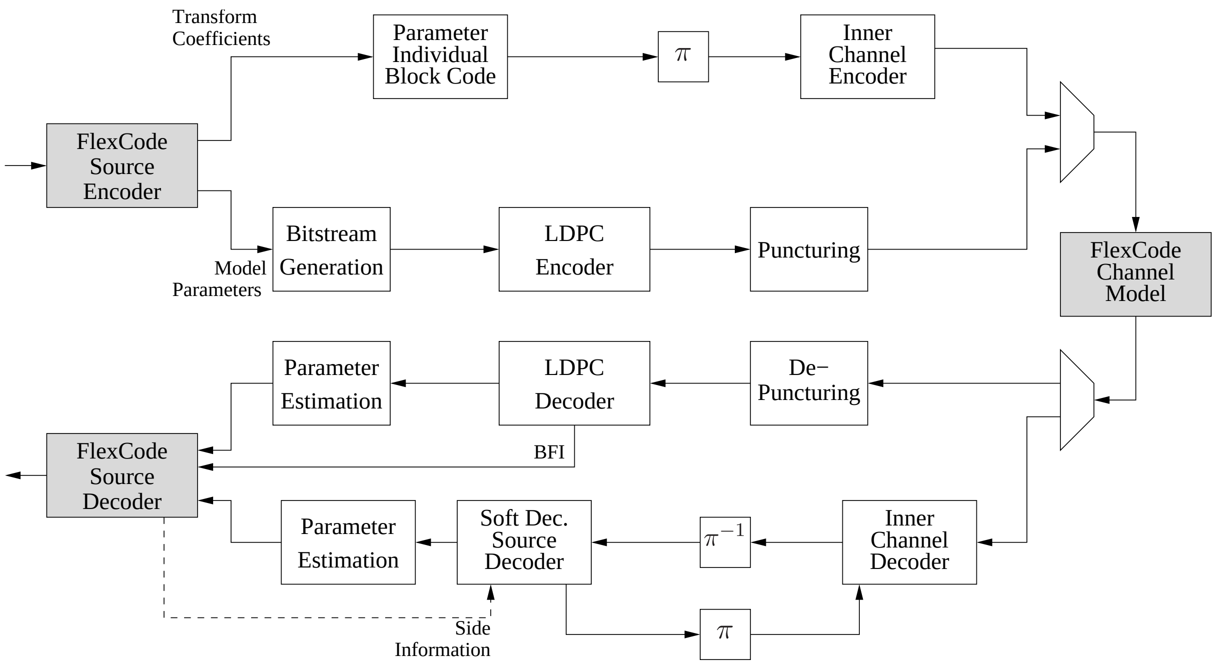


Figure 2.7: Block diagram of the *FlexCode* channel coder with constrained entropy quantization and Turbo ISCD compression

we refer the reader to Section 5.5.1 in [Fle08c] which gives an example of a selection of the parameter individual block codes in dependency of the effective quantizer step size (which takes into account the variance of the transform coefficient distribution). The inner channel coder is a rate > 1 convolutional encoder, i.e., the actual compression is performed by the inner channel encoder. At the receiver, iterative source-channel decoding is performed and the soft decision source decoder makes use of the distribution of the signal and the knowledge of quantizer step size in order to generate *a priori* knowledge of zeroth order which is then exploited by the SDSD [ACS08].
However, the Turbo source coding approach for joint source compression and channel coding has not been considered in the final version of the *FlexCode* channel coder which is used by the demonstrator, mainly for complexity reasons. If an error-free transmission channel is present, the complexity of the arithmetic encoding and decoding is very small and their is almost no additional channel decoding complexity (a single iteration suffices in this case). On the other hand, if Turbo source coding is utilized, a high computational complexity is required even in error-free conditions as the ISCD loop has to be executed with a large number of iterations until convergence is observed.

2.3.3 Constrained Resolution (CR) Case

In the case of CR quantization, the source encoder determines the number of quantization levels L_ℓ (and thus also the number of required bits $N_{B,\ell}$) for each transform coefficient. Unlike in the CE case, where the arithmetic encoder generates a bit stream, this is different in the CR case. As the model parameters are also quantized using constrained resolution, the bit stream generation is similar to the case described in Section 2.2. The transform coefficient quantization index $i_\ell^{[\mathrm{TC}]}$ has been quantized using $L_\ell = 2^{N_{B,\ell}}$ levels and the natural binary representation of $i_\ell^{[\mathrm{TC}]}$ gives the bit pattern $\mathbf{x}_\ell^{[\mathrm{TC}]} = (x_{\ell,1}^{[\mathrm{TC}]}, \ldots, x_{\ell,N_{B,\ell}}^{[\mathrm{TC}]})$. This operation is performed in the block *Bit mapping* in the block diagram of the constrained resolution (CR) channel coder given in Fig. 2.8. If, due to adverse channel conditions, additional channel coding redundancy is required, one or several parity check bits are added to each transform coefficient. This is performed by the block *Parameter Individual Block Code* in Fig. 2.8. The *Bit Distribution Algorithm* (BDA) determines the selection of the parameter individual block codes. More details on the BDA can be found in Section 2.3.3.1. The selection of appropriate parameter individual block codes is described below.
After parameter individual block coding, the resulting bits of a complete frame are interleaved and encoded by the inner channel encoder, which is a recursive systematic rate $1/2$ convolutional encoder with generator polynomials $G_{I,1}(z) = 1$ and $G_{I,2}(z) = \frac{1}{1+z^{-1}+z^{-2}+z^{-3}}$ with puncturing. By puncturing the systematic bits, this code can be rendered into a rate-1 code which has been proven to give good results for iterative source-channel decoding. If the transmission channel is expected to be good, rate < 1 inner coding is utilized as the number of decoding iterations can then be reduced, thus reducing the overall decoding complexity. If the channel is expected to become worse, the inner code is punctured to rate 1 as in this case the highest gains by iterative decoding are expected. In that case, the redundancy is only added by the outer code component, i.e., the parameter individual block code. After inner channel coding, the bit stream of the transform coefficients is concatenated with the bit stream of the model parameters and the frame is transmitted over the *FlexCode* channel model. At the receiver, first the model parameters are decoded and fed together with the BFI (for frame erasure concealment purposes) to the *FlexCode* source decoder. For details, see Section 2.3.2. The transform coefficients are decoded using iterative source-channel decoding (ISCD). The basic ISCD principle has already been presented in [Fle08c]. For a thorough introduction to ISCD, we refer the reader to [ACS08]. The soft decision source decoder gets the information from the BDA and can decode the parameter individual block code using soft decision source decoding. The soft decision source decoder (SDSD) can exploit all available statistical side information on the transform coefficients such as an unequal distribution or correlation as well as the artificial

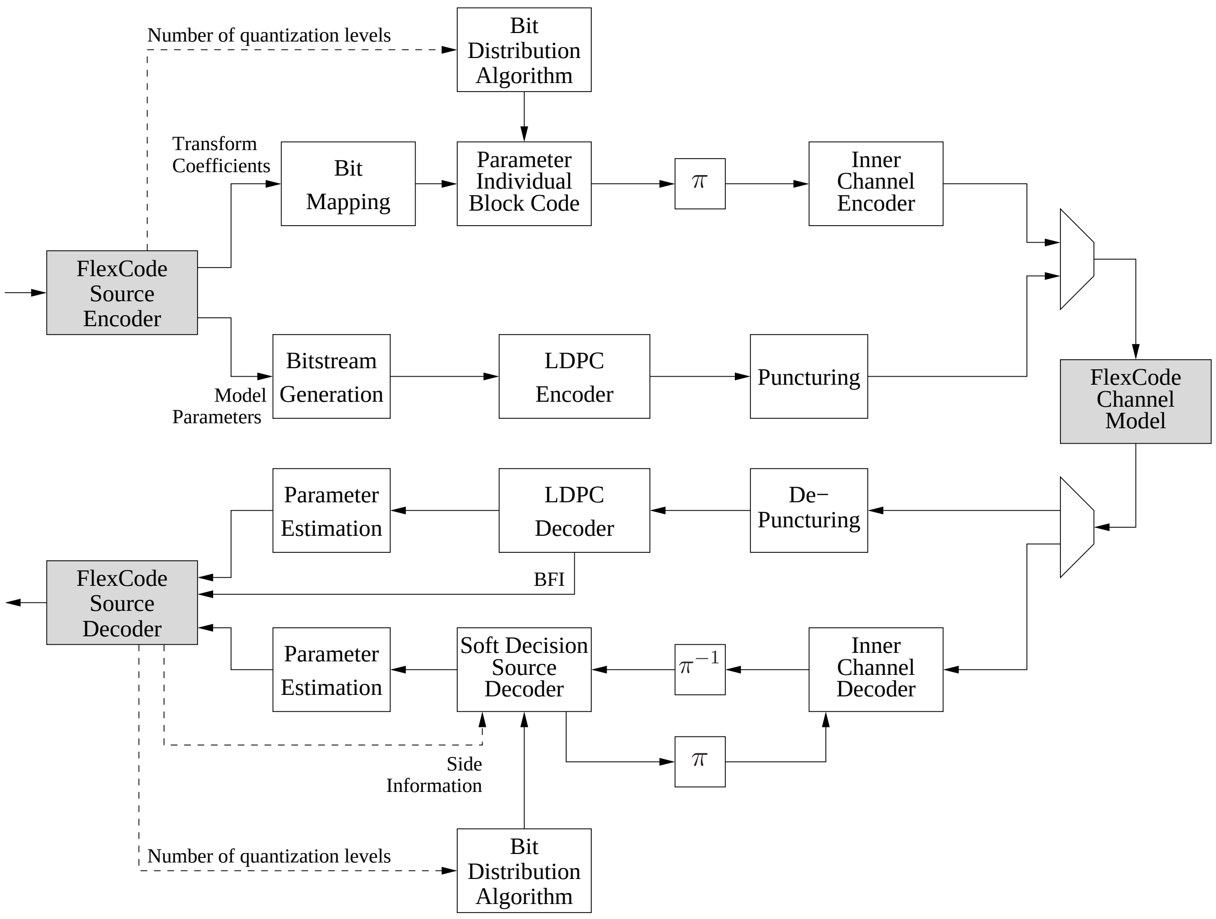


Figure 2.8: Block diagram of the *FlexCode* channel coder with constrained resolution quantization

redundancy added by the parameter individual block code. Due to complexity reasons and in order to maintain the real time requirements for the demonstrator, we only consider unequal distribution of the transform coefficient indices (*a priori* knowledge of order zero).

The first simulation example shows a snapshot of the *FlexCode* coding system: The *FlexCode* source coder operates at a coding rate of 24 kbit/s and the channel coder at a coding rate of $1/2$. Model parameters and transform coefficients are encoded with rate $1/2$ such that the total amount of data to be transmitted on the channel amounts to 48 kbit/s. Figure 2.9 shows a snapshot of the EXIT chart [ten01] analysis of the system. It can be seen that a (narrow) decoding tunnel exists between the characteristic of the channel code $\mathcal{C}_{\mathrm{CC}}$ (code punctured to rate-1) and the characteristic of the SDSD $\mathcal{C}_{\mathrm{SDSD}}$. Note that the SDSD in this example does not exploit any residual redundancy in the quantized parameters. Better performance is obtained if redundancy such as unequal parameter distribution is exploited.

The effect of achieving higher performance by exploiting unequal transform coefficient quantizer index distribution is shown by the next simulation example. A preliminary version of the MDCT source encoder with CR quantization has been used with the *FlexCode* channel coder and transmission over an AWGN channel with varying E_s/N_0. The overall coding rate has been set to $1/2$. The model has been protected with an LDPC code. The unequal distribution of the *FlexCode* source encoder has been measured offline by means of oversampled histograms using the *FlexCode* speech database. MOS scores for SDSD with and without the exploitation of *a priori* knowledge over varying channel quality are depicted in Fig. 2.10 for 2 and 4 receiver iterations, respectively. It can be seen that the utilization of zeroth order

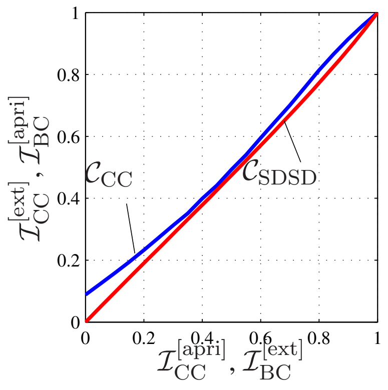


Figure 2.9: EXIT chart analysis (snapshot, CR case) at $E_s/N_0 = -2.6\,\text{dB}$

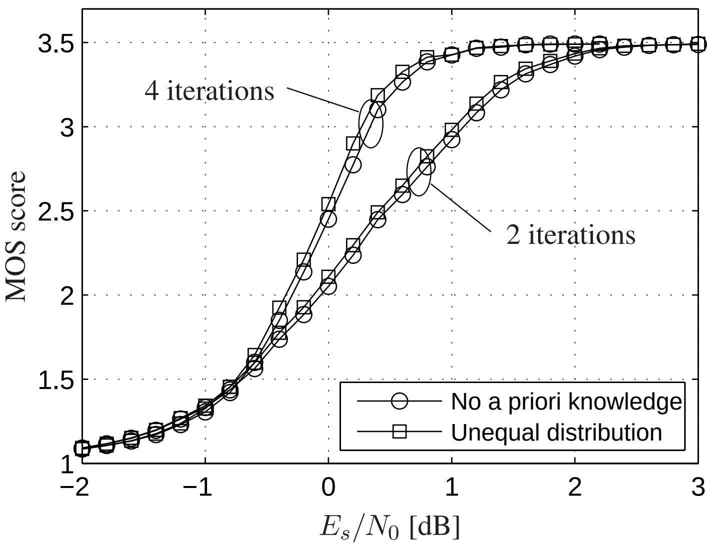


Figure 2.10: Influence of the exploitation of *a priori* knowledge on the decoding performance

a priori knowledge slightly improves the decoding quality as expected [FV01], [Fin98]. Further gains are expected by exploiting the inter-frame correlation of the transform coefficients.

The overall performance of the system can basically be controlled by a careful selection of the parameter individual block codes (PIBC). In order to show how the PIBCs can be carefully selected, we use the following notation: the ℓ'th transform coefficient is quantized using L_ℓ quantization levels and therefore, after natural binary bit mapping, a bit vector $\mathbf{x}_\ell^{[\text{TC}]} = (x_{\ell,1}^{[\text{TC}]}, \ldots, x_{\ell,N_{B,\ell}}^{[\text{TC}]})$ is obtained with $N_{B,\ell} = \log_2 L_\ell$. The parameter individual block code determines a bit vector $\mathbf{y}_\ell^{[\text{TC}]}$ consisting of $N_{B,\ell}+N_{P,\ell}$ bits, i.e., additional $N_{P,\ell}$ parity bits are generated by the code. Therefore, $\mathbf{y}_\ell^{[\text{TC}]} = (y_{\ell,1}^{[\text{TC}]}, \ldots, y_{\ell,N_{B,\ell}+N_{P,\ell}}^{[\text{TC}]})$. As each transform coefficient can be quantized using a different number of bits due to the flexibility of the source encoder and the Fox algorithm used for constrained resolution quantization and as the number of additional parity bits depends on the gross bit rate available on the channel (the number of parity bits per parameter is determined by the bit distribution algorithm, the codes that are utilized should have a certain structure and the parity bits should be determinable by simple operations. Due to these constraints, it is not feasible to store a large number of optimized codes but is useful to store only a certain structure of the code. Two different code families have been studied in the *FlexCode* project:

- Multiple parity check (MPC) codes. It has been shown in, e.g., [CAV06], that a single parity check code shows good performance in Iterative Source-Channel Decoding. Therefore, the first family of codes simply computes the parity check of all bits of $\mathbf{x}_\ell^{[\text{TC}]}$ and then repeats it. For instance, if the bit distributor assigns $N_{P,\ell}$ parity bits to a transform coefficients, all $N_{P,\ell}$ parity bits are obtained by
$$y_{\ell,N_{B,\ell}+1}^{[\text{TC}]} = y_{\ell,N_{B,\ell}+2}^{[\text{TC}]} = \ldots = y_{\ell,N_{B,\ell}+N_{P,\ell}}^{[\text{TC}]} = x_{\ell,1}^{[\text{TC}]} \oplus x_{\ell,2}^{[\text{TC}]} \oplus \cdots \oplus x_{\ell,N_{B,\ell}}^{[\text{TC}]}\,. \tag{2.6}$$
As the code is assumed to be systematic, the first $N_{B,\ell}$ entries of $\mathbf{y}_\ell^{[\text{TC}]}$ are given by
$$y_{\ell,i}^{[\text{TC}]} = x_{\ell,i}^{[\text{TC}]}, \quad \forall i \in \{1,\ldots,N_{B,\ell}\}\,. \tag{2.7}$$
This general generator matrix of such a code is
$$\mathbf{G}_{\text{MPC}} = \left(\mathbf{I}_{N_{B,\ell}}\ \mathbf{1}_{N_{B,\ell},N_{P,\ell}}\right) \tag{2.8}$$
with $\mathbf{I}_{N_{B,\ell}}$ denoting the $N_{B,\ell} \times N_{B,\ell}$ identity matrix and $\mathbf{1}_{N_{B,\ell},N_{P,\ell}}$ denoting the $N_{B,\ell} \times N_{P,\ell}$ all-one matrix (i.e., a matrix containing only 1's).

- Following [Bre09], a different class of codes is studied. They have shown to perform better than multiple parity check if $N_{B,\ell} > 5$. As this may occur frequently in the *FlexCode* source coder due to the multi-mode quantization, these codes have also been studied. The code is again systematic, i.e., $y_{\ell,i}^{[\text{TC}]} = x_{\ell,i}^{[\text{TC}]}, \forall i \in \{1,\ldots,N_{B,\ell}\}$. The first parity bit is obtained again by performing a parity check over all bits of $\mathbf{x}_\ell^{[\text{TC}]}$, i.e.,
$$\begin{aligned} y_{\ell,N_{B,\ell}+1}^{[\text{TC}]} &= x_{\ell,1}^{[\text{TC}]} \oplus x_{\ell,2}^{[\text{TC}]} \oplus \cdots \oplus x_{\ell,N_{B,\ell}}^{[\text{TC}]} && (2.9)\\ &= \bigoplus_{j=1}^{N_{B,\ell}} x_{\ell,j}^{[\text{TC}]} && (2.10)\end{aligned}$$
The following parity check bits then exclude one of the systematic bits, i.e., they are obtained by
$$y_{\ell,N_{B,\ell}+1+i}^{[\text{TC}]} = \bigoplus_{\substack{j=1\\ j\neq 1+((i-1) \bmod N_{B,\ell})}}^{N_{B,\ell}} x_{\ell,i}^{[\text{TC}]}\,. \tag{2.11}$$
For the example of $N_{B,\ell} = 4$ and $N_{P,\ell} = 7$ the generator matrix looks like follows:
$$\mathbf{G}_{\text{OPT},\ell}^{[4,7]} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1\\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0\\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1\\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}. \tag{2.12}$$
Due to the special structure of this code the parity bits are also easy to generate using simple binary operations and the generator matrices do not need to be explicitly stored.

Both code families have been compared in a simulation setup similar to the one in Section 2.3.2.3. A preliminary version of the MDCT source encoder with CR quantization has been used with the *FlexCode* channel coder and transmission over an AWGN channel with varying E_s/N_0. The source coding rate has been 16kbps and the gross transmission rate 32kbps leading to an overall coding rate of $1/2$. The model has been protected with an LDPC code. MOS scores for both codes and a varying channel quality are depicted in Fig. 2.11 for 2 and 5 receiver iterations, respectively. It can be seen that the repeated multiple parity check codes (first code presented above) has a significantly lower quality than the optimized algorithm according to [Bre09]. Therefore, the optimized parameter individual block code algorithm is chosen for the final *FlexCode* channel coder setup.

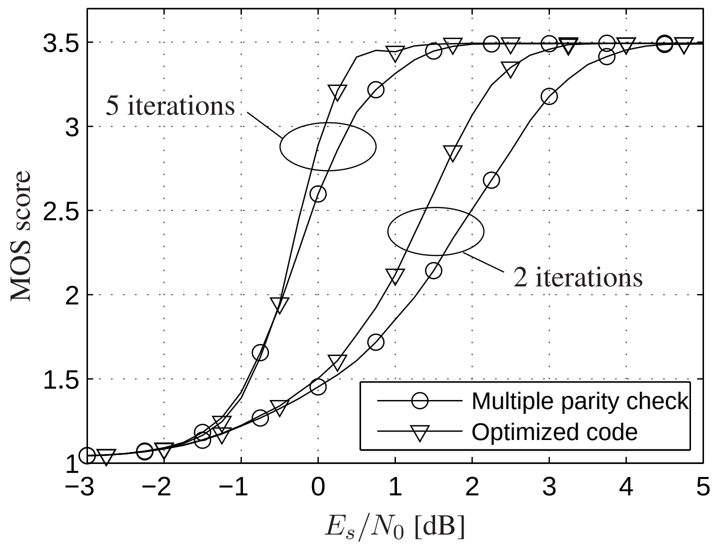


Figure 2.11: Comparison of different parameter individual block codes

2.3.3.1 Rate Adaptation

The rate adaptation in the constrained resolution channel coder is performed by using different parameter individual block codes for each transform coefficient. The channel coder knows the source encoder bit rate and the gross bit rate allowed on the channel. Therefore, it can compute the number of bits that the channel encoder is allowed to add to each frame. After subtraction of the number of bits which will be added by the inner channel encoder (in the case of rate 1 inner channel encoding, only the zero termination bits will be added by the inner convolutional code), the number of bits $N_{\text{PIBC}}^{[\text{tot}]}$ that are going to be added by the parameter individual block code can be determined. Furthermore, let $N_{\text{TC,source}}^{[\text{tot}]}$ the number of bits utilized by the source encoder for the transform coefficients (TC) and let $N_{\text{TC,coded}}^{[\text{tot}]}$ be the number of bits for the transform coefficients after parameter individual block coding. Therefore $N_{\text{TC,coded}}^{[\text{tot}]} = N_{\text{TC,source}}^{[\text{tot}]} + N_{\text{PIBC}}^{[\text{tot}]}$. Furthermore, N_{TC} denotes the number of transform coefficients per frame. The *bit distribution algorithm* (BDA) takes care of selecting the appropriate number of parity bits for each transform coefficient. Two different bits distribution algorithms have been studied during the *FlexCode* project:

- **Algorithm 1**: First, $\left\lfloor \frac{N_{\text{PIBC}}^{[\text{tot}]}}{N_{\text{TC}}} \right\rfloor$ bits are distributed to each transform coefficient and then 1 additional bit is distributed to the first $N_{\text{PIBC}}^{[\text{tot}]} - N_{\text{TC}} \cdot \left\lfloor \frac{N_{\text{PIBC}}^{[\text{tot}]}}{N_{\text{TC}}} \right\rfloor$ transform coefficients.

- **Algorithm 2**: while the first algorithm tries to achieve a more or less constant number of parity bits per parameter, the second algorithm tries to achieve a more or less constant coding rate for each transform coefficient. The total coding rate of the parameter individual block coding is $\frac{N_{\text{TC,source}}^{[\text{tot}]}}{N_{\text{TC,coded}}^{[\text{tot}]}}$. The bit distribution algorithm aims at distributing the bits such that this rate is obtained for each transform coefficient. If the source coder determines that $N_i^{[\text{TC}]}$ bits (or $L_i = 2^{N_i^{[\text{TC}]}}$ quantization levels) are employed for each transform coefficient, then the bit distributor assigns $\left\lfloor \left(\frac{N_{\text{TC,coded}}^{[\text{tot}]}}{N_{\text{TC,source}}^{[\text{tot}]}} - 1 \right) N_i^{[\text{TC}]} \right\rfloor$ additional parity bits to each transform coefficient. One additional parity

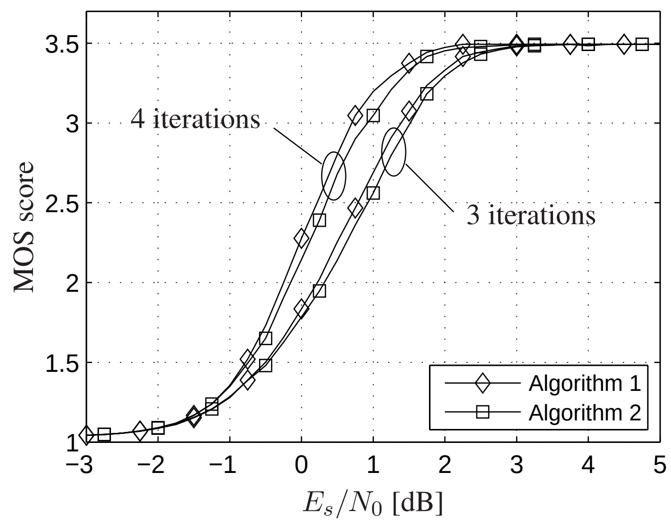


Figure 2.12: Comparison of both bit distribution algorithms

bit is then assigned to the remaining $N_{\text{PIBC}}^{[\text{tot}]} - \sum_{i=1}^{N_{\text{TC}}} \left\lfloor \left(\frac{N_{\text{TC,coded}}^{[\text{tot}]}}{N_{\text{TC,source}}^{[\text{tot}]}} - 1 \right) N_i^{[\text{TC}]} \right\rfloor$ transform coefficients.

Both algorithms have been compared in a simulation setup similar to the one in Section 2.3.2.3. A preliminary version of the MDCT source encoder with CR quantization has been used with the *FlexCode* channel coder and transmission over an AWGN channel with varying E_s/N_0. The source coding rate has been 16kbps and the gross transmission rate 32kbps leading to an overall coding rate of $1/2$. The model has been protected with an LDPC code. MOS scores for both setups and a varying channel quality are depicted in Fig. 2.12 for 3 and 4 receiver iterations, respectively. It can be seen that the first, simpler algorithm 1 outperforms algorithm 2, which tries to maintain a constant coding rate for each transform coefficient. Therefore, the first algorithm is chosen for the final *FlexCode* channel coder setup.

2.3.3.2 Multiple Description Coding

Multiple description coding can also be applied to constrained resolution quantization. The principle is the same as in the constrained entropy case leading to a similar block diagram. The block diagram of the constrained resolution multiple descriptions channel codec is given in Fig. 2.13. Again the *FlexCode* source encoder outputs the model coefficients, which are encoded using the system given in Section 2.2 and two descriptions for the transform coefficients, which are encoded using the serial concatenation of a parameter individual block code and a convolutional encoder (see Section 2.3.3, Fig. 2.8). According to [KK09], [ZKK09], only two descriptions are used and each description contains a copy of the model parameters. Both descriptions are transmitted independently over the *FlexCode* channel model and at the receiver first the L-values model parameters are combined (diversity transmission, see also Section 2.3.2.5), decoded and using the side information generated by the *FlexCode* source decoder, both descriptions of the transform coefficients can be decoded using the ISCD system introduced beforehand. The source encoder then reconstructs the audio signal using the model parameters and either both or a single description of the transform coefficients. If both descriptions are lost, frame erasure concealment has to be performed. Throughout the project, it has been decided that the multiple description operation for constrained resolution quantization is not used by the source encoder.

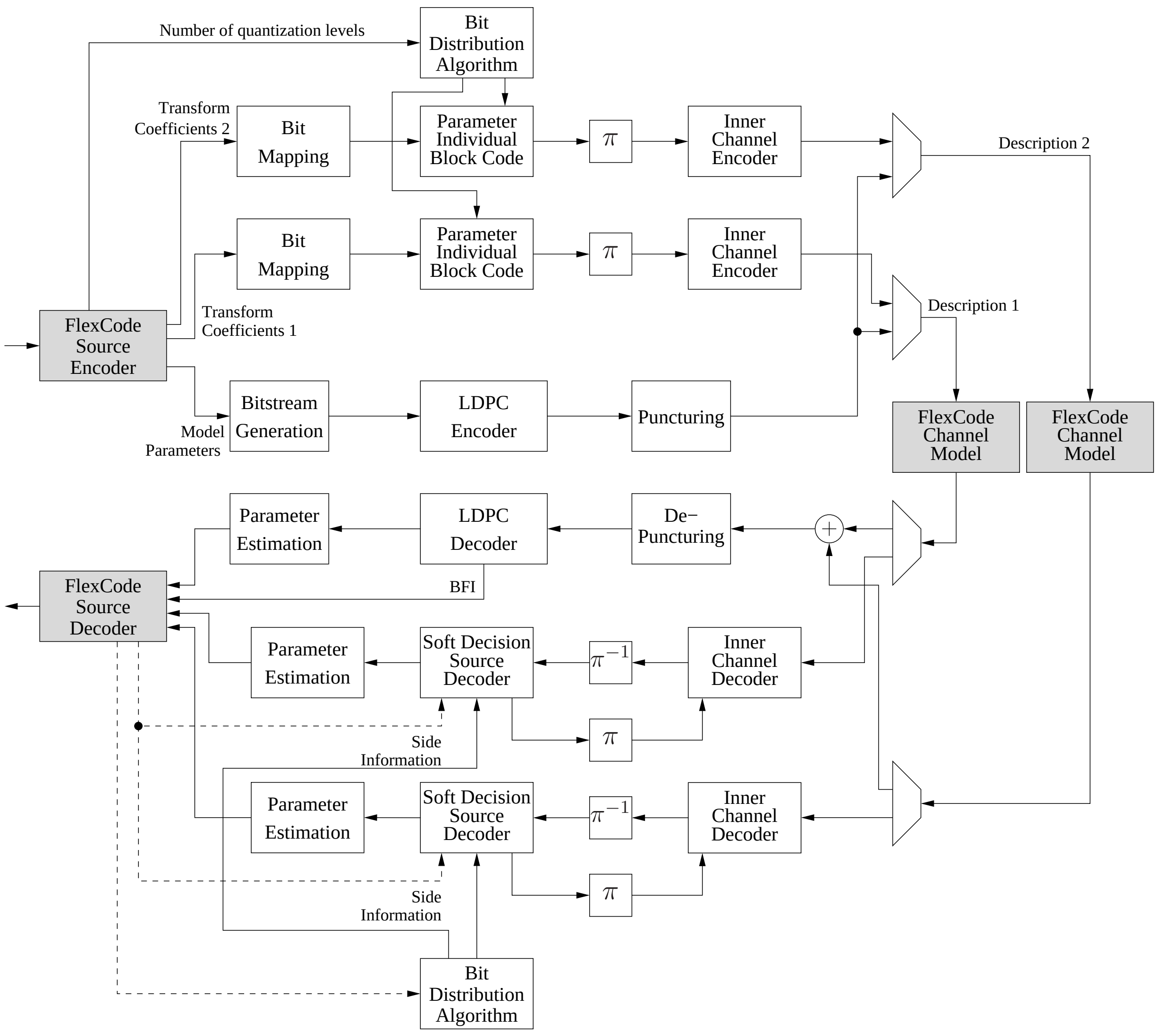


Figure 2.13: Block diagram of the *FlexCode* channel coder with constrained resolution quantization and multiple description coding

Chapter 3

Complexity-Reduced Iterative Source-Channel Decoding

In this Chapter, we give hints how the complexity of Iterative Source-Channel Decoding can be reduced. We present two strategies for complexity reduction, which are

- conditional quantization
- reduced-search source decoders.

The concept of conditional quantization, presented in [SVAC08] is a concept to exploit the auto-correlation of the source parameters during the quantization process. This results however in a modified quantizer and therefore also in a quality loss. The second principle reduces the amount of operations in the source decoder by reducing the search space of the source decoder [SVA08]. The advantage of this method is that the maximum achievable reproduction quality is not altered. Both algorithms can also be combined if a quality loss can be tolerated.
The execution of the SDSD, however, can be computationally quite demanding, especially if large quantizer codebooks are employed. In non-iterative transmission systems, it is possible to execute the SDSD only for the most significant bits, as proposed in [LK03]. However, if such a source decoder is utilized in an ISCD transmission scheme, the source decoder can only generate extrinsic information for the most significant bits, leading to a sub-optimal performance.
Therefore, we propose a different approach for a complexity-reduced ISCD receiver. It has been observed that quite a high number of certain pairs of consecutive quantized values $(\bar{u}_\kappa, \bar{u}_{\kappa-1})$ occur with small probabilities if the sequence is correlated. If the transmitter is modified in a way that these transitions are not allowed, the SDSD does not need anymore a fully developed trellis, but a trellis with a reduced number of state transitions. Note that only the number of state transitions is reduced while the number of states remains the same. This differs from the M-algorithm [FA98], which works with a reduced number of states.
However, the conditional quantizer also affects the quality of the reconstructed signal. Therefore, we propose a receiver-only approach called M-SDSD. This approach is similar to the well-known M-algorithm [WM04], [FA98], known from channel decoding. A similar approach has also been introduced in [Adr03]. We show that the number of operations can be considerably reduced by only slightly affecting the overall system performance. Furthermore, by combining the M-SDSD with the *conditional quantization* we show that the complexity can even further be reduced.
First, in Section 3.1 the basic abstract system model is presented. This system model is utilized throughout this chapter in order to simply demonstrate the concepts of the complexity reduction algorithms. Adapting the findings of this chapter to the *FlexCode* channel decoder is a more or less trivial task. In

Section 3.2 a first concept for complexity reduction is presented, called *conditional quantization*. The complexity can be further reduced by modifying the search-space of the source decoder in Section 3.3. It is also shown how both concepts can be combined in Section 3.3.

3.1 System Model

In the following, we will briefly review the *iterative source-channel decoding* (ISCD) system. In Fig. 3.1 the baseband model of ISCD is depicted. At time instant t a source encoder generates a frame $\underline{u}_t = (u_1, \dots u_{K_S})$ of K_S unquantized source codec parameters u_κ, with $\kappa \in \{1, \dots, K_S\}$ denoting the position within the frame. Each value u_κ is individually mapped to a quantizer reproduction level $\bar{u}_\kappa$, with $\bar{u}_\kappa \in \mathbb{U} = \{\bar{u}^{(1)}, \dots, \bar{u}^{(Q)}\}$. The set $\mathbb{U}$ denotes the quantizer codebook with a total number of $|\mathbb{U}| = Q$ codebook entries. A unique (bipolar) bit pattern $\mathbf{x}_\kappa \in \mathbb{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(Q)}\}$ of w^* bits (i.e., $\mathbb{X} \subseteq \{\pm 1\}^{w^*}$), with $w^* \geq \lceil \log_2 Q \rceil \doteq w$, is assigned to each quantizer level $\bar{u}_\kappa$ according to the index assignment

$$\begin{aligned} \Gamma : \quad & \mathbb{U} \rightarrow \mathbb{F}_2^{w^*} \\ & \bar{u}^{(i)} \mapsto \Gamma(\bar{u}^{(i)}) = \mathbf{x}^{(i)} \end{aligned}$$

with $\mathbb{F}_2 = \{0, 1\}$.

The single bits of a bit pattern $\mathbf{x}_\kappa$ are indicated by $x_\kappa(m)$, $m \in \{1, \dots, w^*\}$. If $w^* > \log_2 Q$, the index assignment Γ is called *redundant index assignment* [CAV06] and can be considered to be the composite function $\Gamma = \Gamma_\mathrm{R} \circ \Gamma_\mathrm{NB}$ (i.e., $\Gamma(\bar{u}) = (\Gamma_\mathrm{R} \circ \Gamma_\mathrm{NB})(\bar{u}) = \Gamma_\mathrm{R}(\Gamma_\mathrm{NB}(\bar{u}))$). The function Γ_NB performs a non-redundant *natural binary* index assignment, i.e., the binary representation of the codebook index of $\bar{u}$ is assigned to $\Gamma_\mathrm{NB}(\bar{u})$. The function Γ_R, which is the redundancy introducing part, can be regarded as being a (linear or non-linear) block code of rate $r^\mathsf{IA} = w/w^*$. The concept of non-linear block codes employed as redundant index assignments has been successfully utilized in, e.g., [HV05]. After index assignment, K_S bit patterns are grouped to a frame of bit patterns $\underline{\mathbf{x}} = (\mathbf{x}_1, \dots, \mathbf{x}_{K_S})$ consisting of $K_S \cdot w^*$ bits. The frame $\underline{\mathbf{x}}$ of bits is then re-arranged by a bit interleaver π in a deterministic, pseudo-random like manner. The interleaved frame with $K_S \cdot w^*$ bits is denoted as $\underline{\breve{\mathbf{x}}}$.

For channel encoding of a frame $\underline{\breve{\mathbf{x}}}$, we use a convolutional code of constraint length $J+1$ and of rate r^C. In general, any channel code can be used as long as the respective decoder is able to provide the required

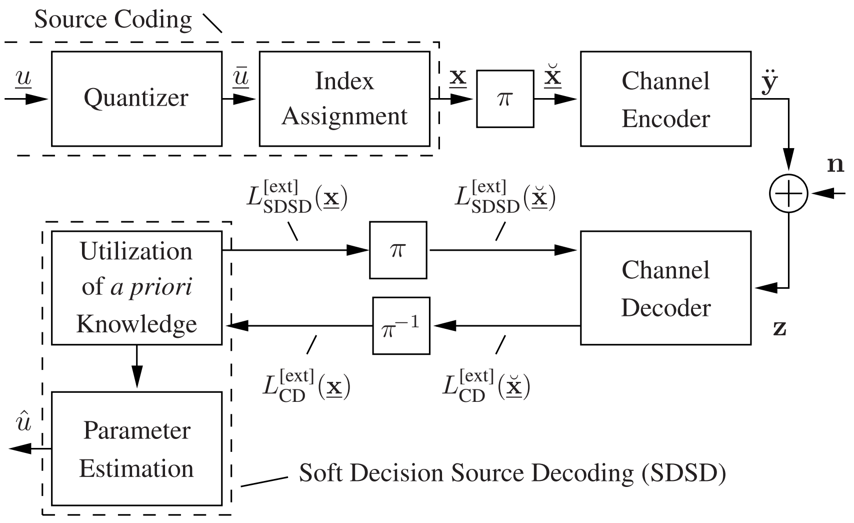

Source Coding
u
Quantizer
$\bar{u}$
Index Assignment
$\underline{\mathbf{x}}$
π
$\underline{\breve{\mathbf{x}}}$
Channel Encoder
$\ddot{\mathbf{y}}$
$\mathbf{n}$
$L_\mathrm{SDSD}^\mathrm{[ext]}(\underline{\mathbf{x}})$
$L_\mathrm{SDSD}^\mathrm{[ext]}(\underline{\breve{\mathbf{x}}})$
Utilization of *a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

Figure 3.1: Baseband model of the utilized ISCD system

extrinsic reliabilities. In this paper, we restrict ourselves to rate $r^{\mathsf{C}} = 1$, recursive, non-systematic convolutional codes as it has been shown [KHC06] that the inner code of a serially concatenated system should be recursive in order to be capacity achieving. For the termination of the code, J tail bits are appended to $\breve{\underline{\mathbf{x}}}$. The encoded frame of length $K_S \cdot w^* + J$ is denoted by $\underline{\mathbf{y}}$. The bits y_k of $\underline{\mathbf{y}}$ are indexed by $k \in \{1, \dots, K_S \cdot w^* + J\}$. Prior to transmission over the channel, the encoded bits y_k are mapped to bipolar values $\ddot{y}_k$ forming a sequence $\ddot{\mathbf{y}} \in \{\pm 1\}^{K_S \cdot w^* + J}$. We only consider BPSK modulation in this paper in order to demonstrate the concept, which can easily be extended to include higher order modulation schemes [CBAV05], [CAV06] or channel equalization [SCV07]. Note that in Fig. 3.1 the baseband model is considered.
On the channel, the modulation symbols $\ddot{y}_k$ (with symbol energy $E_{\mathrm{s}} = 1$) are subject to additive white Gaussian noise (AWGN) with known variance $\sigma_n^2 = N_0/2$.
The received symbols $z_k \in \{\pm 1\}$ are transformed to L-values [HOP96] prior to being evaluated in a Turbo process which exchanges *extrinsic* reliabilities between the channel decoder (CD) and the soft decision source decoder (SDSD).
The channel decoder used in this paper is based on the LogMAP algorithm [BCJR74], [HOP96]. For the derivations of the equations for computing the *extrinsic* probabilities of the SDSD, we refer the reader to the literature, e.g., [Goe01], [FV01], [AV05], [ACS08]. In Section 3.2.2.1, we will briefly revise the SDSD equations and give expressions in the logarithmic domain in order to evaluate the complexity and the complexity savings of the proposed algorithms.

3.2 Complexity Reduction by Conditional Quantization

3.2.1 Conditional Scalar Quantization

In this section we present the concept of conditional scalar quantization, which enables a very efficient realization (in terms of computational complexity) of the soft decision source decoder (SDSD) [SVAC08]. Although we present the concept for scalar quantization and a first order Markov model only, the extension to vector quantization as well as higher order Markov models is straightforward.
If $\mathbb{U}$ denotes the original quantizer codebook, let $\mathcal{C} = \{\mathcal{C}^{(1)}, \dots, \mathcal{C}^{(Q)}\}$ denote the set of all quantization cells with

$$\mathcal{C}^{(q)} = \{u : |u - \bar{u}^{(q)}| < |u - \bar{u}^{(\ell)}|, \forall \bar{u}^{(\ell)} \in \mathbb{U}, \bar{u}^{(\ell)} \neq \bar{u}^{(q)}\}. \tag{3.1}$$

Conditional quantization exploits the correlation between successive samples in such a way that the quantization of the current value u_κ depends on the previously quantized value $\bar{u}_{\kappa-1}^{(i)}$. For quantizing the current sample u_κ, the conditional quantizer only considers codebook entries $\bar{u}^{(j)}$ where the conditional probability $P(\bar{u}_\kappa^{(j)} | \bar{u}_{\kappa-1}^{(i)})$ is above a certain threshold $\mathcal{T}$. We define a set of reduced codebooks $\mathbb{U}_{\text{red},i}$ with

$$\mathbb{U}_{\text{red},i} = \left\{ \bar{u}_\kappa^{(j)} : P\left(\bar{u}_\kappa^{(j)} | \bar{u}_{\kappa-1}^{(i)}\right) > \mathcal{T}, \forall\, \bar{u}_\kappa^{(j)} \in \mathbb{U} \right\}. \tag{3.2}$$

The conditional quantizer uses the reduced codebook $\mathbb{U}_{\text{red},i}$ to quantize the sample u_κ if the previous sample has been quantized to $\bar{u}_{\kappa-1}^{(i)}$. Let $|\mathbb{U}_{\text{red},i}|$ denote the number of entries in the reduced codebook $\mathbb{U}_{\text{red},i}$. The total number of allowed transitions $\bar{u}_{\kappa-1}^{(i)} \to \bar{u}_\kappa^{(j)}$ is thus reduced from $\mathcal{N}' = Q^2$ to

$$\mathcal{N} \doteq \sum_{i=1}^{Q} |\mathbb{U}_{\text{red},i}|\,. \tag{3.3}$$

This (reduced) number of transitions is directly linked to the complexity of the source decoder as shall be seen in Section 3.2.2. Let $\mathbb{X}_{\text{red},i}$ denote the set of all bit patterns assigned to the reduced codebook

$\mathbb{U}_{\text{red},i}$. Furthermore, we define

$$\mathbb{U}'_{\text{red},j} \doteq \left\{ \bar{u}^{(i)}_{\kappa-1} : \bar{u}^{(j)}_{\kappa} \in \mathbb{U}_{\text{red},i}, \forall\, i \in \{1, \dots, Q\} \right\} \tag{3.4}$$

to be the set of all codebook entries $\bar{u}^{(i)}_{\kappa-1}$ that allow a transition $\bar{u}^{(i)}_{\kappa-1} \rightarrow \bar{u}^{(j)}_{\kappa}$. Again, $\mathbb{X}'_{\text{red},j}$ denotes the set of assigned bit patterns to the entries of $\mathbb{U}'_{\text{red},j}$.
Note that the utilization of conditional quantization also modifies the *a priori* knowledge of first order which is exploited in the source decoder. We denote this modified conditional probability $P_{\text{red}}(\bar{u}^{(j)}|\bar{u}^{(i)})$ with $\bar{u}^{(j)} \in \mathbb{U}_{\text{red},i}$ and $\bar{u}^{(i)} \in \mathbb{U}$. Again, for the conditional quantizer, quantization cells $\mathcal{C}^{(q)}_{\text{red},i}$ with

$$\mathcal{C}^{(q)}_{\text{red},i} = \{ u : |u - \bar{u}^{(q)}| < |u - \bar{u}^{(\ell)}|, \forall \bar{u}^{(\ell)} \in \mathbb{U}_{\text{red},i}, \bar{u}^{(\ell)} \neq \bar{u}^{(q)} \}$$

can be defined for $q = 1, \dots, |\mathbb{U}_{\text{red},i}|$. For a given (stationary) source with (two-dimensional) joint probability function $p_U(u_\kappa, u_{\kappa-1}) = p_U(u_\kappa|u_{\kappa-1}) \cdot p_U(u_{\kappa-1})$ the quantization noise amounts to (e.g. [VM06], [JN84])

$$N = \sum_{\bar{u}^{(i)}_{\kappa-1} \in \mathbb{U}} \; \sum_{\bar{u}^{(j)}_{\kappa} \in \mathbb{U}_{\text{red},i}} \int_{\mathcal{C}^{(i)}} \int_{\mathcal{C}^{(j)}_{\text{red},i}} \left(\zeta - \bar{u}^{(j)}_{\kappa} \right)^2 p_U(\zeta, \nu) \, \mathrm{d}\zeta \, \mathrm{d}\nu. \tag{3.5}$$

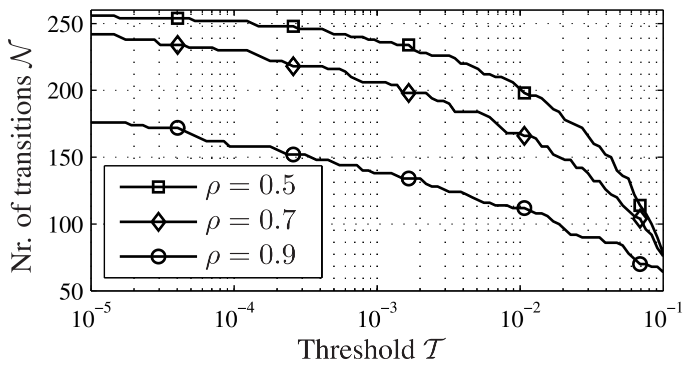
The quantization noise is determined by considering all possible previous samples $\bar{u}^{(i)}_{\kappa-1} \in \mathbb{U}$ and then calculating the quantization noise amount of the pair $(\bar{u}^{(i)}_{\kappa-1}, \bar{u}^{(j)}_{\kappa})$ with $\bar{u}^{(j)}_{\kappa} \in \mathbb{U}_{\text{red},i}$ by solving the double integral in (3.5). The total quantization noise is then obtained by summing over all combinations of $(\bar{u}^{(i)}_{\kappa-1}, \bar{u}^{(j)}_{\kappa})$
As an example, we assume that the source realizes a Gauss-Markov process of first order with correlation ρ, zero mean and variance $\sigma_u^2 = 1$. The two-dimensional joint distribution of the source amounts to (e.g., [VM06])

$$p_U(u_\kappa, u_{\kappa-1}) = \frac{1}{2\pi\sqrt{1-\rho^2}} \cdot \mathrm{e}^{-\frac{u_\kappa^2 + u_{\kappa-1}^2 - 2\rho u_\kappa u_{\kappa-1}}{2(1-\rho^2)}} . \tag{3.6}$$

Figure 3.2 depicts the number of transitions $\mathcal{N}$ as a function of the threshold $\mathcal{T}$ for $\rho \in \{0.5; 0.7; 0.9\}$ and $Q = 16$ quantizer levels (top subplot) as well as for $Q = 32$ (bottom subplot). As it is expected intuitively, a higher correlation ρ leads to a lower number of transitions $\mathcal{N}$ as transitions $u_{\kappa-1} \rightarrow u_\kappa$ with $|u_\kappa - u_{\kappa-1}| \gg 0$ occur less frequently.
The most interesting question however is how much the signal quality is affected by conditional quantization. Therefore, (3.5) is evaluated for the same source and the signal-to-noise ratio after quantization is determined as a function of the reduced number of transitions $\mathcal{N}$. The original codebook $\mathbb{U}$ is assumed to be the optimum Lloyd-Max codebook [JN84]. The results are depicted in Fig. 3.3, for $Q = 16$ (top subplot) and $Q = 32$ (bottom subplot). It can be seen that for $\rho \geq 0.7$ the number of transitions can be halved (e.g., from $Q^2 = 1024$ to $\mathcal{N} \approx 500$ for $Q = 32$) without affecting the SNR considerably.
Note that although we introduced the concept of conditional quantization for intra-frame correlation only, it is also easily applicable to inter-frame correlation.

3.2.2 Complexity Considerations

In Section 3.2.2.1 we first revise the SDSD equations [AVS01], [Goe01], [ACS08] and then modify the expressions such that the SDSD operates in the logarithmic domain in Section 3.2.2.2. The given expressions are already modified in such a way that conditional quantization is incorporated. The full expressions are needed in order to determine the complexity figures in Section 3.2.2.3.



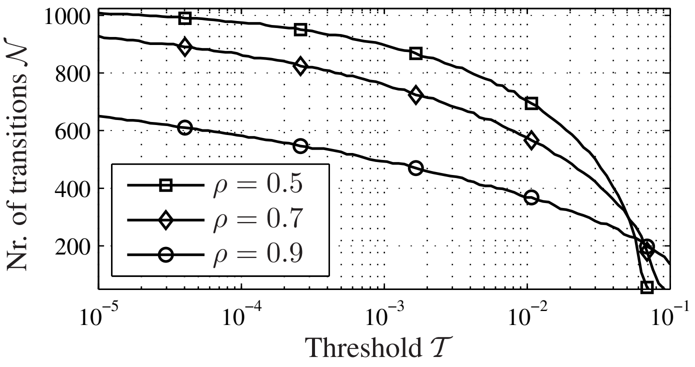


Figure 3.2: Number of transitions $\mathcal{N}$ as a function of the threshold $\mathcal{T}$ for $Q = 16$ (top) and $Q = 32$ (bottom).

3.2.2.1 Soft Decision Source Decoding (SDSD) for Conditional Quantization

The SDSD may be interpreted as a modification of the well-known BCJR algorithm [BCJR74], operating on a fully developed trellis diagram. In this section, we assume that only intra-frame correlation is exploited by the SDSD, the extension towards inter-frame correlation is straightforward by exchanging the position indices κ with time indices t.

The input to the soft decision source decoder (SDSD) are the extrinsic L-values generated by the channel decoder

$$\begin{aligned} L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m)) &= L_{\text{CD}}^{[\text{ext}]}(x_\kappa(m)) \\ &= \ln\left(\frac{P_{\text{CD}}^{[\text{ext}]}(x_\kappa(m)=+1)}{P_{\text{CD}}^{[\text{ext}]}(x_\kappa(m)=-1)}\right). \end{aligned} \tag{3.7}$$

The first step of the SDSD consists in determining the factors $\theta(\mathbf{x}_\kappa^{(j)})$ for each distinct bit pattern $\mathbf{x}_\kappa^{(j)}$ with

$$\theta(\mathbf{x}_\kappa^{(j)}) = \exp\left(\sum_{m=1}^{w^*} \frac{x_\kappa^{(j)}(m)}{2} L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m))\right). \tag{3.8}$$

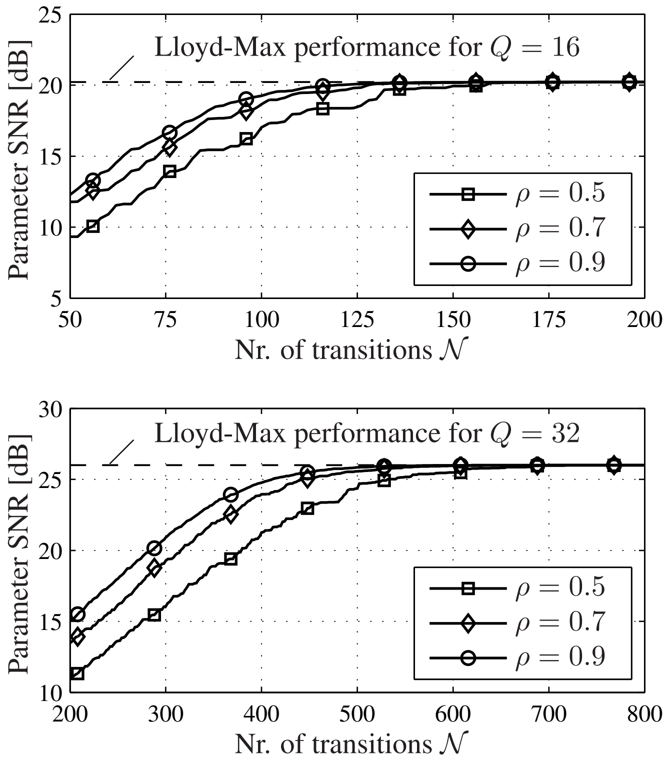


Figure 3.3: Parameter SNR as a function of the number of transitions for different values of auto-correlation ρ for $Q = 16$ (top) and $Q = 32$ (bottom).

Note that $x_\kappa^{(j)}(m) \in \{\pm 1\}$. The forward and backward recursion of the SDSD are given by

$$\alpha(\mathbf{x}_\kappa^{(j)}) = \theta(\mathbf{x}_\kappa^{(j)}) \sum_{\mathbf{x}_{\kappa-1}^{(i)} \in \mathbb{X}'_{\text{red},j}} \alpha(\mathbf{x}_{\kappa-1}^{(i)}) P_{\text{red}}(\mathbf{x}_\kappa^{(j)} | \mathbf{x}_{\kappa-1}^{(i)}) \tag{3.9}$$

$$= \theta(\mathbf{x}_\kappa^{(j)}) \cdot A(\mathbf{x}_\kappa^{(j)}) \tag{3.10}$$

$$\beta(\mathbf{x}_{\kappa-1}^{(i)}) = \sum_{\mathbf{x}_\kappa^{(j)} \in \mathbb{X}_{\text{red},i}} \beta(\mathbf{x}_\kappa^{(j)}) \theta(\mathbf{x}_\kappa^{(j)}) P_{\text{red}}(\mathbf{x}_\kappa^{(j)} | \mathbf{x}_{\kappa-1}^{(i)}) \tag{3.11}$$

with the initialization $\alpha(\mathbf{x}_0^{(\ell)}) = P(\mathbf{x}^{(\ell)})$, $\beta(\mathbf{x}_{K_S}^{(\ell)}) = 1$, $\forall \ell \in \{1, \ldots, Q\}$, (see, e.g., [ACS08]). We furthermore define $A(\mathbf{x}_\kappa^{(j)}) \doteq \sum_{\mathbf{x}_{\kappa-1}^{(i)} \in \mathbb{X}'_{\text{red},j}} \alpha(\mathbf{x}_{\kappa-1}^{(i)}) P_{\text{red}}(\mathbf{x}_\kappa^{(j)} | \mathbf{x}_{\kappa-1}^{(i)})$. Of course, the summations in (3.9) and (3.11) only have to be evaluated for those pairs of $(\mathbf{x}_{\kappa-1}^{(i)}, \mathbf{x}_\kappa^{(j)})$ with an existing transition between the corresponding $(\bar{u}_{\kappa-1}^{(i)}, \bar{u}_\kappa^{(j)})$ (i.e., $P(\bar{u}_\kappa^{(j)} | \bar{u}_{\kappa-1}^{(i)}) > \mathcal{T}$). Further note that due to conditional quantization the exploited *a priori* information changes. Therefore $P_{\text{red}}(\mathbf{x}_\kappa^{(j)} | \mathbf{x}_{\kappa-1}^{(i)})$ has to be utilized instead of $P(\mathbf{x}_\kappa^{(j)} | \mathbf{x}_{\kappa-1}^{(i)})$ (see also Section 3.2.1).

The final step of the SDSD consists in determining the extrinsic information for each bit $x_\kappa(m)$ which is given by [AVS01] (in terms of L-values)

$$L_{\text{SDSD}}^{[\text{ext}]}(x_\kappa(m)) = \ln \frac{\sum\limits_{\substack{j=1\\ x_\kappa^{(j)}(m)=+1}}^{Q} \beta(\mathbf{x}_\kappa^{(j)}) \theta_m^{[\text{ext}]}(\mathbf{x}_\kappa^{(j)}) A(\mathbf{x}_\kappa^{(j)})}{\sum\limits_{\substack{j=1\\ x_\kappa^{(j)}(m)=-1}}^{Q} \beta(\mathbf{x}_\kappa^{(j)}) \theta_m^{[\text{ext}]}(\mathbf{x}_\kappa^{(j)}) A(\mathbf{x}_\kappa^{(j)})} \tag{3.12}$$

with

$$\theta_m^{[\text{ext}]}(\mathbf{x}_\kappa^{(j)}) = \exp\left(\sum_{\substack{\ell=1\\ \ell \neq m}}^{w^*} \frac{x_\kappa^{(j)}(\ell)}{2} L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(\ell)) \right). \tag{3.13}$$

3.2.2.2 SDSD in the Logarithmic Domain

In a practical implementation the translation of the BCJR algorithm to the logarithmic domain offers several advantages such as, e.g., better numerical stability [RVH95]. In the following, we derive the equations for the SDSD in the logarithmic domain. Therefore, we define $\tilde{\theta}(\mathbf{x}_\kappa^{(j)}) \doteq \ln \theta(\mathbf{x}_\kappa^{(j)})$, $\tilde{\alpha}(\mathbf{x}_\kappa^{(j)}) \doteq \ln \alpha(\mathbf{x}_\kappa^{(j)})$, as well as $\tilde{\beta}(\mathbf{x}_\kappa^{(j)}) \doteq \ln \beta(\mathbf{x}_\kappa^{(j)})$. With (3.8), the expression of $\tilde{\theta}(\mathbf{x}_\kappa^{(j)})$ becomes

$$\tilde{\theta}(\mathbf{x}_\kappa^{(j)}) = \sum_{m=1}^{w^*} \frac{1}{2} x_\kappa^{(j)}(m) L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m)). \tag{3.14}$$

Taking the natural logarithm of (3.9) and using $\tilde{\alpha}(\mathbf{x}_\kappa^{(j)})$ as well as $\tilde{\theta}(\mathbf{x}_\kappa^{(j)})$ leads to (with $\tilde{P}_{\text{red}}(\mathbf{x}_\kappa^{(j)}|\mathbf{x}_{\kappa-1}^{(i)}) \doteq \ln P_{\text{red}}(\mathbf{x}_\kappa^{(j)}|\mathbf{x}_{\kappa-1}^{(i)})$) [RVH95]

$$\begin{aligned} \tilde{\alpha}(\mathbf{x}_\kappa^{(j)}) &= \tilde{\theta}(\mathbf{x}_\kappa^{(j)}) + \ln\left(\sum_{\mathbf{x}_{\kappa-1}^{(i)} \in \mathbb{X}'_{\text{red},j}} \mathrm{e}^{\ln\left(\alpha(\mathbf{x}_{\kappa-1}^{(i)}) \cdot P_{\text{red}}(\mathbf{x}_\kappa^{(j)}|\mathbf{x}_{\kappa-1}^{(i)})\right)} \right) \\ &= \tilde{\theta}(\mathbf{x}_\kappa^{(j)}) + \ln\left(\sum_{\mathbf{x}_{\kappa-1}^{(i)} \in \mathbb{X}'_{\text{red},j}} \mathrm{e}^{\tilde{\alpha}(\mathbf{x}_{\kappa-1}^{(i)}) + \tilde{P}_{\text{red}}(\mathbf{x}_\kappa^{(j)}|\mathbf{x}_{\kappa-1}^{(i)})} \right) \end{aligned} \tag{3.15}$$

The expression $\ln\left(\mathrm{e}^{\delta_1} + \mathrm{e}^{\delta_2}\right)$ which is part of (3.15) can be computed using the Jacobian logarithm [RVH95]:

$$\begin{aligned} \ln\left(\mathrm{e}^{\delta_1} + \mathrm{e}^{\delta_2}\right) &= \max(\delta_1, \delta_2) + f_c(|\delta_1 - \delta_2|) \\ &\doteq \max{}^*(\delta_1, \delta_2) \end{aligned} \tag{3.16}$$

with $f_c(\zeta) = \ln\left(1 + \mathrm{e}^{-\zeta}\right)$ and $\ln\left(\mathrm{e}^{\delta_1} + \mathrm{e}^{\delta_2} + \mathrm{e}^{\delta_3} + \ldots\right) = \max^*(\delta_1, \delta_2, \delta_2, \ldots) = \max^*(\delta_1, \max^*(\delta_2, \max^*(\delta_3, \ldots)))$. Furthermore, $\max^*(\delta_1, -\infty) = \max^*(-\infty, \delta_1) = \delta_1$. The $\max^*$ function can be efficiently implemented using, e.g., a lookup table.
Using the $\max^*$ function, (3.15) can then be rewritten as

$$\tilde{\alpha}(\mathbf{x}_\kappa^{(j)}) = \tilde{\theta}(\mathbf{x}_\kappa^{(j)}) + \max_{\mathbf{x}_{\kappa-1}^{(i)} \in \mathbb{X}'_{\text{red},j}}{}^{*} \left(\tilde{\alpha}(\mathbf{x}_{\kappa-1}^{(i)}) + \tilde{P}_{\text{red}}(\mathbf{x}_\kappa^{(j)}|\mathbf{x}_{\kappa-1}^{(i)}) \right). \tag{3.17}$$

Similarly, the backward recursion (3.11) can be rewritten as

$$\tilde{\beta}(\mathbf{x}_{\kappa-1}^{(i)}) = \max_{\mathbf{x}_{\kappa}^{(j)}\in\mathbb{X}_{\text{red},i}}{}^{*} \left(\tilde{\beta}(\mathbf{x}_{\kappa}^{(j)}) + \tilde{\theta}(\mathbf{x}_{\kappa}^{(j)}) + \tilde{P}_{\text{red}}(\mathbf{x}_{\kappa}^{(j)}|\mathbf{x}_{\kappa-1}^{(i)})\right). \tag{3.18}$$

The determination of the extrinsic information (3.12) can also be expressed using the $\max^*$ operator

$$\begin{aligned} L_{\text{SDSD}}^{[\text{ext}]}(x_\kappa(m)) = & \max_{\substack{j=1 \\ x_\kappa^{(j)}(m)=+1}}^{Q}{}^{*} \left(\tilde{\alpha}(\mathbf{x}_\kappa^{(j)}) + \tilde{\beta}(\mathbf{x}_\kappa^{(j)}) - \frac{1}{2} L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m))\right) \\ & - \max_{\substack{j=1 \\ x_\kappa^{(j)}(m)=-1}}^{Q}{}^{*} \left(\tilde{\alpha}(\mathbf{x}_\kappa^{(j)}) + \tilde{\beta}(\mathbf{x}_\kappa^{(j)}) + \frac{1}{2} L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m))\right) \end{aligned} \tag{3.19}$$

by consecutively exploiting the facts that $\ln \theta_m^{[\text{ext}]}(\mathbf{x}_\kappa^{(j)}) = \tilde{\theta}(\mathbf{x}_\kappa^{(j)}) - \frac{1}{2} x_\kappa^{(j)}(m) L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m))$ (compare (3.8) and (3.13)) and $\tilde{\theta}(\mathbf{x}_\kappa^{(j)}) + \ln A(\mathbf{x}_\kappa^{(j)}) = \tilde{\alpha}(\mathbf{x}_\kappa^{(j)})$ (see (3.10)).
The last step of the SDSD consists in estimating the parameters $\hat{u}$ which is done here using an MMSE estimation

$$\hat{u} = \sum_{i=1}^{Q} \bar{u}^{(i)} \exp\left(\tilde{\alpha}(\mathbf{x}_\kappa^{(i)}) + \tilde{\beta}(\mathbf{x}_\kappa^{(i)}) + \tilde{C}_1\right) \tag{3.20}$$

with the constant $\tilde{C}_1 \in \mathbb{R}$ which is chosen such that $\sum_{i=1}^{Q} \exp\left(\tilde{\alpha}(\mathbf{x}_\kappa^{(i)}) + \tilde{\beta}(\mathbf{x}_\kappa^{(i)}) + \tilde{C}_1\right) \overset{!}{=} 1$. Note that the parameter estimation only has to be performed once per frame and not for each iteration.
In some cases (e.g. if inter-frame correlation is exploited) only the forward recursion can be carried out due to delay constraints. In this case $\tilde{\beta}(\mathbf{x}_\kappa^{(j)})$ is set to zero in (3.19) as the backward recursion (3.18) does not need to be carried out. Equation (3.19) then reduces to

$$L_{\text{SDSD}}^{[\text{ext}]}(x_\kappa(m)) = \max_{\substack{j=1 \\ x_\kappa^{(j)}(m)=+1}}^{Q}{}^{*} \left(\tilde{\alpha}(\mathbf{x}_\kappa^{(j)}) - \frac{1}{2} L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m))\right) - \max_{\substack{j=1 \\ x_\kappa^{(j)}(m)=-1}}^{Q}{}^{*} \left(\tilde{\alpha}(\mathbf{x}_\kappa^{(j)}) + \frac{1}{2} L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m))\right). \tag{3.21}$$

3.2.2.3 Complexity of the SDSD

The evaluation of (3.14) requires $Q \cdot w^*$ additions per parameter as the $\tilde{\theta}(\mathbf{x}_\kappa^{(j)})$ have to be determined for each possible bit pattern $\mathbf{x}_\kappa^{(j)} \in \mathbb{X}$. The factors $\frac{1}{2} L_{\text{SDSD}}^{[\text{input}]}(x_\kappa(m))$ can be calculated and stored (as they are needed a second time in the run-time of the algorithm) using w^* multiplications per parameter. The multiplication by $x_\kappa(m)$ corresponds to a sign change only as $x_\kappa(m) \in \{\pm 1\}$. The forward and backward recursions also have to be calculated for each $\mathbf{x}_\kappa^{(j)} \in \mathbb{X}$. In the case of conventional SDSD (i.e., $|\mathbb{U}_{\text{red},i}| = |\mathbb{X}_{\text{red},i}| = |\mathbb{U}'_{\text{red},i}| = |\mathbb{X}'_{\text{red},i}| = Q$), the evaluation of (3.17) requires Q^2 $\max^*$ operations as well as $Q+Q^2$ additions per parameter while the evaluation of (3.18) requires Q^2 $\max^*$ operations and $2Q^2$ additions per parameter. If conditional quantization is utilized, the number of $\max^*$ operations each reduces to $\mathcal{N}$ and the number of additions to $Q + \mathcal{N}$ (forward recursion), respectively $2\mathcal{N}$ (backward recursion). Finally, the evaluation of (3.19) requires w^*Q $\max^*$ operations as well as $w^*(2Q+1)$ additions for the conventional as well as for the reduced-complexity SDSD.
Table 3.1 summarizes the number of operations required for both the standard SDSD and the complexity reduced SDSD by conditional quantization. Both cases are considered: the forward/backward algorithm and the forward-only variant which is used if inter-frame correlation is exploited. Note that we do not consider the complexity of the parameter estimation as this step, which is only performed once per frame, is required in any case. Thus the complexity figures given here only include the operations performed in the block denoted "Utilization of *a priori* knowledge" in Fig. 3.1.

Table 3.1: Operations per parameters needed by the SDSD

• Full algorithm:

	Standard	Conditional Quantization
$\max^*$	$2Q^2 + w^*Q$	$2\mathcal{N} + w^*Q$
ADD	$3Q^2 + (3w^* + 1)Q + w^*$	$3\mathcal{N} + (3w^* + 1)Q + w^*$
MUL	w^*	w^*

• Forward only algorithm:

	Standard	Conditional Quantization
$\max^*$	$Q^2 + w^*Q$	$\mathcal{N} + w^*Q$
ADD	$Q^2 + (2w^* + 1)Q + w^*$	$\mathcal{N} + (2w^* + 1)Q + w^*$
MUL	w^*	w^*

3.2.3 Simulation Results for Conditional Quantization

The capabilities of the proposed ISCD system with conditional quantization are demonstrated by two simulation examples. 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 ($E_u = w^* \cdot \frac{1}{r^{\mathsf{C}}} \cdot E_{\mathrm{s}}$). The source is realized by a Gauss-Markov (autoregressive) process with correlation coefficient ρ fixed to $\rho = 0.9$. This auto-correlation value can be observed in typical speech and audio codecs, e.g., for the scale factors in CELP codecs or MP3. The utilized channel code is a rate $r^{\mathsf{C}} = 1$ recursive non-systematic convolutional code of constraint length $J = 4$ with generator polynomial $G^{\mathsf{C}}(D) = \left(\frac{1}{1+D+D^2+D^3}\right)$. The non-iterative reference scheme uses optimized components for non-iterative systems, i.e., a natural binary index assignment with $w^* = w = \log_2\lceil Q\rceil$ and a rate $r^{\mathsf{C}} = \frac{1}{2}$ recursive, systematic convolutional code of constraint length $J = 4$ with $G^{\mathsf{C}}(D) = \left(\frac{1+D^2+D^3}{1+D+D^3}\right)$.

In a first experiment (denoted "experiment A"), we assume that the source exhibits intra-frame correlation, i.e., the single elements u_κ of $\underline{u}_t$ are modelled by a 1st order Gauss-Markov process and a frame consists of $K_S = 50000$ parameters. The quantization is performed using a $Q = 16$ level Lloyd-Max codebook $\mathbb{U}$ and the redundant index assignment Γ_{R} consists of the $(8, 4)$ block code with generator matrix

$$\mathbf{G}^{\Gamma}_{\mathrm{BC}(8,4)} = \begin{pmatrix} 1&0&0&0&1&1&1&0 \\ 0&1&0&0&1&1&0&1 \\ 0&0&1&0&1&0&1&1 \\ 0&0&0&1&0&1&1&1 \end{pmatrix} \tag{3.22}$$

proposed in [CVA06]. This index assignment results in $w^* = 8$ bit. The overall coding rate of the system amounts to $r^{\mathsf{IA}} \cdot r^{\mathsf{C}} = \frac{1}{2}$. At the receiver, 15 iterations are carried out. The simulation results are depicted in Fig. 3.4 for the standard SDSD with "regular" quantization as well as for the complexity-reduced system with conditional quantization for $\mathcal{T} \in \{0.01; 0.03; 0.05\}$. Note that the depicted non-iterative scheme uses SDSD while today's systems with hard-decision source decoding perform even worse. The resulting number of transitions $\mathcal{N}$ as well as the number of required $\max^*$ operations and additions are summarized in Table 3.2. As predicted by (3.5), the maximum attainable parameter SNR is reduced due to the influence of the conditional quantizer. Interestingly, the lower the number of transitions (i.e., for higher values of $\mathcal{T}$), the more the waterfall region is moved towards lower channel qualities.

This behavior can be explained by an EXIT chart analysis [ten01]: The EXIT characteristics for the conventional SDSD as well as for the complexity-reduced SDSD with conditional quantization are de-

Table 3.2: Operations per parameter and iteration needed by the SDSD for experiment A (top) and experiment B (bottom)

$Q = 16$	Standard	Conditional		
$w^* = 8$		$\mathcal{T} = 0.01$	$\mathcal{T} = 0.03$	$\mathcal{T} = 0.05$
Forw./Backw.		$\mathcal{N} = 112$	$\mathcal{N} = 90$	$\mathcal{N} = 84$
$\max^*$	640	352	308	296
ADD	1176	744	678	660

$Q = 32$	Standard	Conditional		
$w^* = 10$		$\mathcal{T} = 0.005$	$\mathcal{T} = 0.01$	$\mathcal{T} = 0.03$
Forw. only		$\mathcal{N} = 408$	$\mathcal{N} = 368$	$\mathcal{N} = 284$
$\max^*$	1344	728	688	604
ADD	1706	1090	1050	966

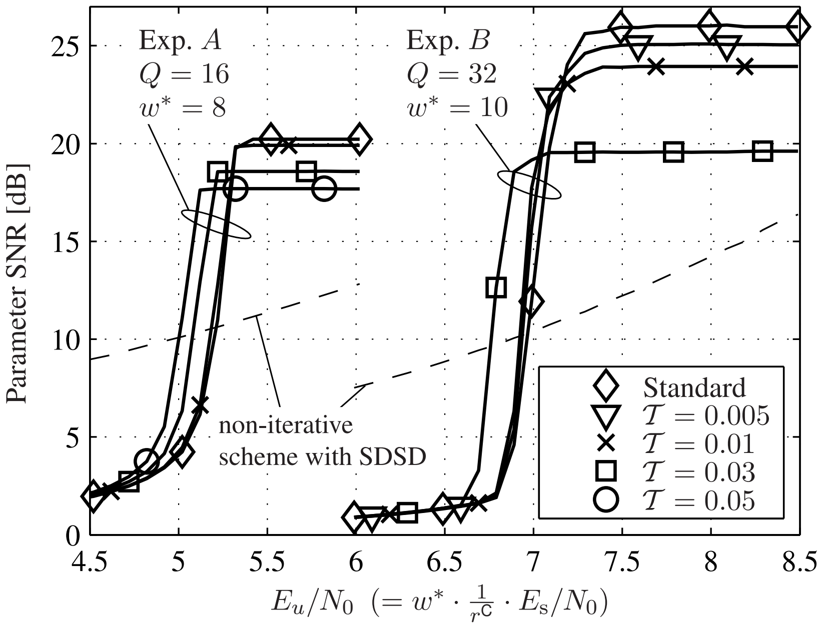


Figure 3.4: Simulation results for experiment A and experiment B

picted in Fig. 3.5-a). The area $\mathcal{A}_{\mathrm{SDSD}}$ under the EXIT curve of the complexity-reduced SDSD is larger than the area under the EXIT curve of the conventional SDSD. This leads to an earlier convergence (for lower channel qualities) as the area underneath the EXIT characteristic is linked to the channel quality where the waterfall region occurs [AKtB04]. The larger area is obvious as $1 - \mathcal{A}_{\mathrm{SDSD}}$ is a function of the conditional entropy $H(\bar{u}_\kappa|\bar{u}_{\kappa-1})$ [Tho07], which is reduced by the conditional quantizer. The reduction is caused by the fact that now $P_{\mathrm{red}}(\mathbf{x}_\kappa^{(j)}|\mathbf{x}_{\kappa-1}^{(i)})$ is exploited by the SDSD (instead of $P(\mathbf{x}_\kappa^{(j)}|\mathbf{x}_{\kappa-1}^{(i)})$). Figure 3.5-b) depicts $\mathcal{A}_{\mathrm{SDSD}}$ as a function of the number of transitions $\mathcal{N}$ for the given setup.

In a second experiment (denoted "experiment B") we assume that the source exhibits inter-frame correlation, i.e., all the single elements u_κ of $\underline{u}_t$ are assumed to be statistically independent from each other. The different samples u_κ are correlated with their counterpart from previous frames. In this experiment, a

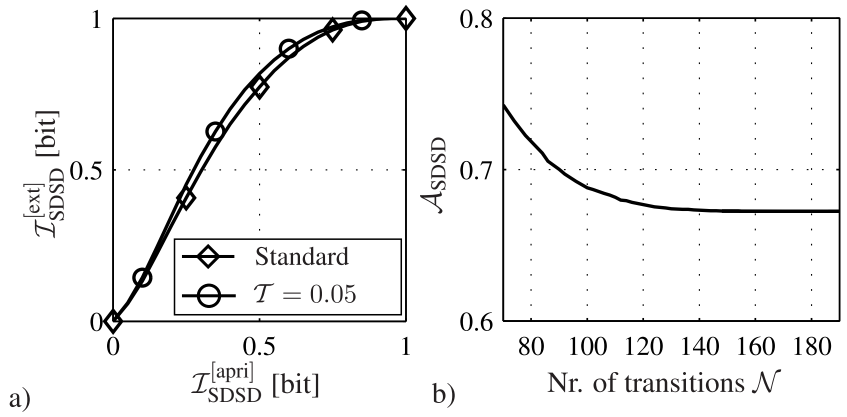


Figure 3.5: EXIT chart analysis for $Q = 16$ and $w^* = 8$ (index assignment generator matrix from (3.22)) for conventional SDSD ("Standard") and for conditional quantization SDSD with $\mathcal{T} = 0.05$, i.e., $\mathcal{N} = 84$. Area under EXIT characteristic as a function of $\mathcal{N}$.

frame consists of $K_S = 250$ parameters. In order not to introduce any additional delay, the forward-only SDSD has to be employed. The block coded index assignment Γ_R utilized for experiment B corresponds to a $(31, 26)$ BCH code shortened to $(10, 5)$ [LC03]. The generator matrix of the shortened systematic code is given by

$$\mathbf{G}^{\Gamma}_{\mathrm{BCH}(10,5)} = \begin{pmatrix} 1&0&0&0&0&1&0&0&1&0 \\ 0&1&0&0&0&0&1&0&0&1 \\ 0&0&1&0&0&1&0&1&1&0 \\ 0&0&0&1&0&0&1&0&1&1 \\ 0&0&0&0&1&1&0&1&1&1 \end{pmatrix} . \tag{3.23}$$

Note that we do not use a specially optimized index assignment but a standard small block code for demonstrating the concept. One possibility to optimize the index assignments are irregular index assignments introduced in [SVCS08].

At the receiver 20 iterations are carried out. The simulation results are depicted in Fig. 3.4 and the number of utilized operations are summarized in Table 3.2. The results behave as expected: again a shift of the waterfall region towards lower channel qualities is observed.

3.3 Complexity Reduction by Reduced Search Soft Decision Source Decoding

3.3.1 M-SDSD

In channel decoding of convolutional codes, the M-algorithm [FA98] can be successfully applied in order to reduce the complexity of the decoder. Another successful field of application is channel equalization of ISI-channels [WM04].

The SDSD in fact is a variant of the BCJR algorithm [BCJR74] operating on a fully developed trellis [ACS08]. Each state corresponds to a quantizer reproduction level (or a bit pattern, respectively). The state transitions correspond to the possible transitions $\mathbf{x}^{(i)}_{\kappa,t-1} \rightarrow \mathbf{x}^{(j)}_{\kappa,t}$. At each trellis transition the M-SDSD determines the M states with the highest probability and only considers these states for computing the state transitions.

Equation (3.17) is performed for all states $\mathbf{x}^{(j)}_{\kappa,t}$ but only the M (saved) best states from the previous time instant $t-1$ are considered. Therefore, the complexity of (3.17) reduces to $Q + MQ$ (with $M < Q$)

additions and MQ $\max^*$ operations. After execution of the complexity-reduced version of (3.17), the M best states have to be determined, i.e., the $\tilde{\alpha}(\mathbf{x}_{\kappa,t}^{(j)})$ with the largest value. This can be done using a simple search with $MQ - \sum_{n=1}^{M} n = MQ - \frac{1}{2}(M^2 + M)$ compare operations (states that have already been chosen don't need to be compared anymore). For determining the extrinsic information, only the M best $\tilde{\alpha}(\mathbf{x}_{\kappa,t})$ are utilized, and therefore the complexity of (3.21) reduces to $w^*(M+1)$ additions and $w^* M$ $\max^*$ operations. Note that we only consider the case where only the forward recursion is carried out as inter-frame redundancy is exploited. This is the relevant case for *FlexCode* as the intra-frame correlations are successfully removed by the (decorrelating) transform.

3.3.2 M-SDSD with Conditional Quantization

A further complexity reduction of the source decoder can be achieved if the transmitter can be modified. In this case, the conventional scalar quantizer can be replaced by the *conditional quantizer* (CQ) introduced in Section 3.2.1. The conditional quantizer has first been described in [SVAC08] and corresponds to a quantizer with memory: depending on the previously quantized sample $\bar{u}_{\kappa,t-1}^{(i)}$ a reduced codebook is utilized which considers only the entries which have a transition probability $P(\bar{u}_{\kappa,t}^{(j)}|\bar{u}_{\kappa,t-1}^{(i)}) > \mathcal{T}$, with $\mathcal{T}$ being the probability threshold of the quantizer. Depending on $\mathcal{T}$, the number of transitions in the trellis diagram is considerably reduced and thus also the number of operations. A drawback of conditional quantization is however the reduced reconstruction quality in error-free channel conditions. For details, see Section 3.2.1.
The complexity of the M-SDSD is more difficult to determine as the number of transitions per state varies. Therefore, only a tight upper bound for the complexity can be given which however is needed for a hardware realization guaranteeing a certain throughput. The conditional quantizer uses a reduced codebook

$$\mathbb{U}_{\text{red},i} = \left\{ \bar{u}_{\kappa,t}^{(j)} : P\left(\bar{u}_{\kappa,t}^{(j)}|\bar{u}_{\kappa,t-1}^{(i)}\right) > \mathcal{T}, \forall\, \bar{u}_{\kappa,t}^{(j)} \in \mathbb{U} \right\} . \tag{3.24}$$

depending on the previously quantized sample. The number of entries of $|\mathbb{U}_{\text{red},i}|$ varies for different values of i. Let S_M denote the sum of the number of transitions of those M states (i.e., quantizer reproduction levels) having the largest number of entries $|\mathbb{U}_{\text{red},i}|$. This is a worst case for the number of transitions in the SDSD: the M states have been selected which lead to the highest number of transitions that have to be calculated by the SDSD.
Figure 3.6 shows an example of the reduction of states and state transitions in the SDSD trellis diagram. Figure 3.6-a) shows the fully developed trellis for $Q = 16$ quantization levels. In Fig. 3.6-b) the number of states is reduced by the M-SDSD with $M = 6$. However, the number of state transitions ($Q = 16$) per state is unchanged. This number can be reduced by applying conditional quantization leading to the trellis in Fig. 3.6-c). Note that Fig. 3.6-b) and Fig. 3.6-c) only show snapshots. The M selected states may vary for each trellis transition.
In the worst case, the complexity of (3.17) now reduces to $Q + S_M$ additions and S_M $\max^*$ operations. The complexity for determining the M best states remains the same as in the case of the M-SDSD. Finally, the complexity of evaluating (3.21) by the conditional quantizer is not affected, therefore $w^*(M+1)$ additions and $w^* M$ $\max^*$ operations are required.
The total number of operations for the three different algorithms (standard SDSD, M-SDSD and M-SDSD with conditional quantization (CQ-M-SDSD) are summarized in Table 3.3.

3.3.3 Simulation Example

The capabilities of the complexity-reduced ISCD system are demonstrated by a simulation example. The *parameter signal-to-noise ratio* (SNR) between the originally generated parameters u and the es-

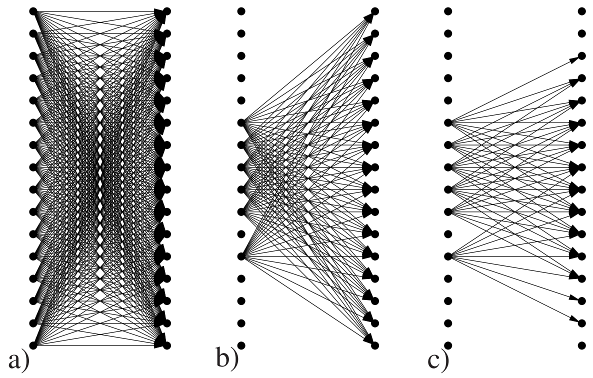


Figure 3.6: Trellis diagrams exploited at the source decoder in (3.17) for $Q = 16$.
a) full trellis diagram
b) exemplarily trellis exploited at one stage of the M-algorithm with $M = 6$
c) if additionally conditional quantization is exploited with $\mathcal{T} = 10^{-2}$

Table 3.3: Operations per parameters needed by the different complexity-reduced SDSD variations (forward-only algorithm)

	ADD	$\max^*$	CMP
Standard	$Q^2 + (2w^* + 1)Q + w^*$	$Q^2 + w^*Q$	
M-SDSD	$(w^* + M + 1)Q + w^*(M + 1)$	$M(Q + w^*)$	$MQ - \frac{1}{2}(M^2 + M)$
CQ-M-SDSD (upper bound)	$(w^* + 1)Q + S_M + w^*(M + 1)$	$S_M + w^*M$	$MQ - \frac{1}{2}(M^2 + M)$

timated parameters $\hat{u}$ is used for quality evaluation. The parameter SNR is plotted for different values of E_s/N_0. The source is realized by K_S independent Gauss-Markov (autoregressive) processes with correlation coefficient ρ fixed to $\rho = 0.9$. This auto-correlation value can be observed in typical speech and audio codecs, e.g., for the scale factors in CELP codecs or MP3. The quantization is performed using a $Q = 16$ level Lloyd-Max codebook $\mathbb{U}$. The utilized block coded index assignment Γ_{R} is a repetition code [CSVA08] ($w^* = 8$). The utilized channel code is a rate $r^{\mathsf{C}} = 1$ recursive non-systematic convolutional code of constraint length $J = 4$ with generator polynomial $G^{\mathsf{C}}(D) = \left(\frac{1}{1+D+D^2+D^3}\right)$. The non-iterative reference scheme as well as the hard-output channel decoding reference uses optimized components for non-iterative systems, i.e., a natural binary index assignment with $w^* = \log_2\lceil Q \rceil = 4$ and a rate $r^{\mathsf{C}} = \frac{1}{2}$ recursive, systematic convolutional code of constraint length $J = 4$ with $G^{\mathsf{C}}(D) = \left(1, \frac{1+D^2+D^3}{1+D+D^3}\right)$.

We assume that the source exhibits inter-frame correlation, i.e., all the single elements u_κ of $\underline{u}_t$ are statistically independent from each other. The different samples u_κ are correlated with their counterpart from previous frames. A frame consists of $K_S = 250$ parameters. In order not to introduce any additional delay, the forward-only SDSD as introduced in Section 3.2.2 is employed.

The simulation results are depicted in Fig. 3.7. At the receiver 15 iterations have been carried out in a first experiment. It can be seen that the utilization of the M-SDSD with $M = 6$ does not cause any noteworthy performance losses for good channel conditions ($E_s/N_0 > -4\,\mathrm{dB}$). The application of conditional quantization further reduces the complexity at the expense of a slightly decreased parameter SNR in good channel conditions [SVAC08]. The fact that the CQ has a better performance in the waterfall region is explained in [SVAC08]. With $\mathcal{T} = 10^{-2}$ and $M = 6$, the factor S_M can be determined to $S_M = 50$, by considering the 6 codebooks $\mathbb{U}_{\mathrm{red},i}$ with the highest number of entries. The complexity per parameter of the three different utilized SDSD algorithms with the configuration of the simulation

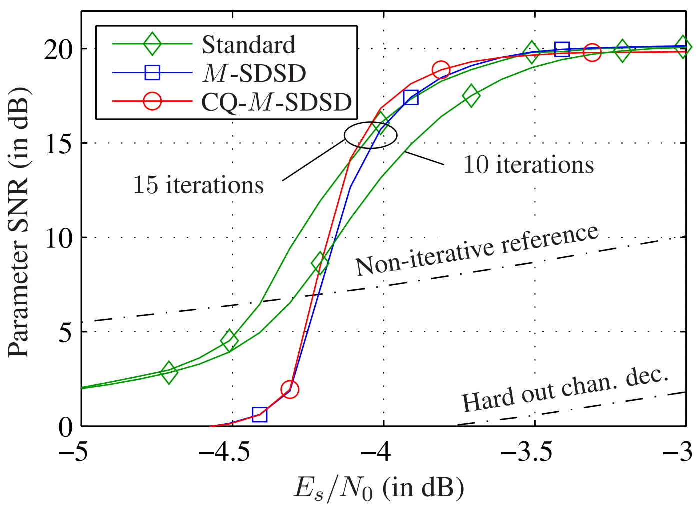


Figure 3.7: Parameter SNR performance for different reduced-search source decoders ($M = 6, \mathcal{T} = 10^{-2}$)

	ADD	$\max^*$	CMP
Standard	536	384	
M-SDSD	296	144	75
CQ-M-SDSD (upper bound)	218	98	75

Table 3.4: Number of operations necessary for the different algorithms for complexity reduction

example are summarized in Table 3.4.
With $w^* = 8$ and $r^{\mathsf{C}} = 1$, the LogMAP decoder [HOP96], [RVH95] requires approximately 656 additions and 256 $\max^*$ operations per parameter. If we assume in a first approximation that all three operations (additions, $\max^*$, and compares) require the same amount of computing power in a hardware realization, the total number of operations amounts to 19545 per parameter if 15 iterations are utilized and the M-SDSD with conditional quantization ($\mathcal{T} = 10^{-2}$) is utilized. If the full SDSD is employed, only 10 iterations can be carried out if the above number of operations (19545) per parameter shall be fixed as an upper bound. The simulation result for 10 iterations is also given in Fig. 3.7. It can be seen that it is advantageous to utilize a complexity-reduced source decoder and a higher number of iterations if the total number of operations to be performed is limited.

Chapter 4

Irregular Index Assignments for Iterative Source-Channel Decoding and Their Applications

In this Chapter, the concept of irregular index assignments [Fle08c], [SVCS08] is recapitulated and some advancements are presented. For instance it is shown how the ISCD system with irregular index assignments can be used to achieve *Unequal Error Protection* (UEP), how near-capacity channel coding of a generic bit stream can be performed using the ISCD system and how error-resilient source compression can be realized.
In Section 4.1 the abstract system model utilized throughout this Chapter is presented, while in Section 4.2 the concept of irregular index assignments is recapitulated. A simulation example and the UEP property are given in Section 4.3. The utilization of the ISCD system for encoding a generic bit stream is presented in Section 4.4 while finally it is shown in Section 4.5 how the ISCD system can be utilized for realizing error-resilient near-lossless source compression by a simple modification of the optimization criterion.

4.1 System Model

In Fig. 4.1 the baseband model of the considered ISCD system is depicted. At time instant t a source encoder generates a frame $\underline{u}_t = (u_{1,t}, \ldots, u_{K_S,t})$ of K_S unquantized source codec parameters u_κ with $\kappa \in \{1, \ldots, K_S\}$ denoting the position in the frame. Each value u_κ is individually mapped to a quantizer reproduction level $\bar{u}_\kappa$, with $\bar{u}_\kappa \in \mathbb{U} = \{\bar{u}^{(1)}, \ldots, \bar{u}^{(Q)}\}$. The set $\mathbb{U}$ denotes the quantizer codebook with a total number of $|\mathbb{U}| = Q$ codebook entries. In this paper, we restrict Q to take only values which are powers of 2, i.e., $Q = 2^w$, with $w \in \mathbb{N} \setminus \{0\}$. A unique bit pattern $\mathbf{x}_\kappa \in \mathbb{X}_\kappa = \{\mathbf{x}_\kappa^{(1)}, \ldots, \mathbf{x}_\kappa^{(Q)}\}$ of w_κ^* bits (i.e., $\mathbb{X}_\kappa \subseteq \mathbb{F}_2^{w_\kappa^*}$, $\mathbb{F} = \{0; 1\}$), with $w_\kappa^* \geq \log_2 Q = w$, is assigned to each quantizer level $\bar{u}_\kappa$ according to the index assignment $\Gamma_\kappa(\bar{u}^{(i)}) = \mathbf{x}^{(i)}$, $i = 1, \ldots, Q$. Note that the index assignment can differ from parameter to parameter. For notational convenience we omit the time index t if the meaning of the equation is non-ambiguous.
The single bits of a bit pattern $\mathbf{x}_\kappa$ are indicated by $x_\kappa(m)$, $m \in \{1, \ldots, w_\kappa^*\}$. If $w_\kappa^* > \log_2 Q = w$, the index assignment Γ_κ is called *redundant index assignment* as it introduces redundancy: More bits than actually necessary are spent to represent a quantizer reproduction level. The index assignment can be considered to be the composite function $\Gamma_\kappa(\bar{u}) = \Gamma_\kappa^{\mathrm{R}}(\Gamma^{\mathrm{NR}}(\bar{u}))$. First, the function Γ^{NR} performs a non-redundant mapping of the Q quantizer reproduction levels to patterns consisting of w bits. Second, the function $\Gamma_\kappa^{\mathrm{R}}$ can be regarded as a (potentially non-linear) block code of rate $r_\kappa^{\mathrm{IA}} = (\log_2 Q)/w_\kappa^* = w/w_\kappa^*$.

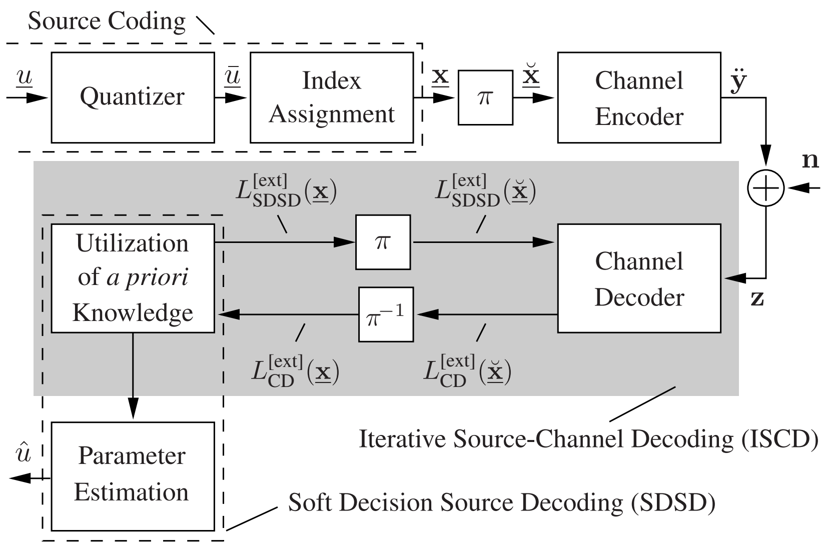

Source Coding
u
Quantizer
$\bar{u}$
Index Assignment
$\underline{\mathbf{x}}$
π
$\underline{\check{\mathbf{x}}}$
Channel Encoder
$\underline{\ddot{\mathbf{y}}}$
$\mathbf{n}$
$L^{[\text{ext}]}_{\text{SDSD}}(\underline{\mathbf{x}})$
$L^{[\text{ext}]}_{\text{SDSD}}(\underline{\check{\mathbf{x}}})$
Utilization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

Figure 4.1: Baseband model of the utilized ISCD system

The concept of non-linear block codes employed as redundant index assignments has been successfully utilized for the robust transmission of source parameters in, e.g., [CVA06]. 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 $N_X \doteq \sum_{\kappa'=1}^{K_S} w^*_{\kappa'} \doteq K_S \cdot \overline{w}^*$ bits. The overall rate of the index assignment is thus

$$\overline{r}^{\text{IA}} \doteq \frac{K_S \cdot w}{\sum\limits_{\kappa=1}^{K_S} w^*_\kappa} = \frac{w}{\overline{w}^*}\,, \tag{4.1}$$

with $\overline{w}^*$ the average number of bits per parameter after index assignment. The frame $\underline{\mathbf{x}}$ of bits is rearranged by a bit interleaver π in a deterministic, pseudo-random like manner. The interleaved frame with $K_S \cdot \overline{w}^*$ bits is denoted as $\underline{\check{\mathbf{x}}}$.
For channel encoding of a frame $\underline{\check{\mathbf{x}}}$, we use a recursive convolutional code of memory J and of rate r^{CC}. In general, any channel code can be used as long as the respective decoder is able to provide the required *extrinsic reliabilities*. For the termination of the convolutional code, J tail bits are appended to $\underline{\check{\mathbf{x}}}$. The encoded frame of length $N_Y \doteq \frac{1}{r^{\text{CC}}}(K_S \cdot \overline{w}^* + J)$ is denoted by $\underline{\mathbf{y}}$. The bits y_k of $\underline{\mathbf{y}}$ are indexed by $k \in \{1, \ldots, N_Y\}$.
Prior to transmission over the channel, the encoded bits y_k are mapped to bipolar values $\ddot{y}_k$ forming a sequence $\underline{\ddot{\mathbf{y}}} \in \{\pm 1\}^{N_Y}$. 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 $\sigma_n^2 = N_0/2$.
The received symbols z_k 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). The channel decoder used in this paper is based on the MAP algorithm [BCJR74]. For the equations of the SDSD, we refer to the literature, e.g., [ACS08] and to Section 3.2.2.2.

4.2 Irregular Index Assignments

In this Section, the concept of irregular index assignments is briefly revised, followed by a design guideline already introduced in [Fle08c]. According to [AKtB04], a necessary condition for a serially concatenated system to be capacity achieving is an inner component with code rate $r^{\text{Inner}} \geq 1$. For the setup

introduced in Section 4.1, this means that the channel code should be of rate $r^{\text{CC}} \geq 1$. For a given channel code, the goal is to find a perfectly matching outer component (source code) to the channel code. This task can be solved for example by the concept of irregular codes [TH02a]. 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. With this concept, capacity achieving codes can be easily found. Furthermore, it becomes easily possible to adapt the code and the rates to changing transmission parameters. This is essentially important in flexible source and channel coders that can adapt on the fly to varying channel and network conditions.
The concept of irregular codes is based on the fact that 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 that searches for optimum weights in order to get an (almost) perfectly matching characteristic can be formulated [TH02a].
Irregular Index Assignments (IIA), introduced in [SVCS08], are an extension of the irregular codes' concept. As stated in Section 4.1, the index assignment for the parameter u_κ comprises a block code Γ_κ^{R} of rate $r_\kappa^{\text{IA}} = (\log_2 Q)/w_\kappa^* = w/w_\kappa^*$. Instead of using the same amount of bit redundancy $w_\kappa^* = \overline{w}^*$ for each parameter in order to achieve an overall rate $w/\overline{w}^*$ outer encoding, we use the concept of irregular codes and vary w_κ^* for each parameter while keeping $\overline{w}^*$ constant. This allows us to optimize the index assignments and to get an SDSD EXIT characteristic which matches the channel decoder's characteristic considerably well.
In [TSV08] the concept of irregular codes has been applied to the inner channel code while utilizing a constant redundant index assignment: This allows us to optimize the inner code to varying channel conditions and transmission scenarios. In this contribution however, we assume that both are constant. In order to account for changing source statistics, we adapt the outer code (i.e., the index assignment) to the (fixed) channel code.
In the following, we will briefly revise the optimization problem originally introduced by Tüchler [TH02a, Tüc04] and then present several modifications. The EXIT characteristic of a specific index assignment Γ_ℓ^{R} shall be denoted by $\mathcal{C}_{\text{SDSD},\ell}$. The vector $\mathbf{c}_\ell \doteq (c_{\ell,1}, \ldots, c_{\ell,P})^T$ contains P sample points of the characteristic $\mathcal{C}_{\text{SDSD},\ell}$. The matrix $\mathbf{C} \doteq (\mathbf{c}_1, \ldots, \mathbf{c}_L)$ contains all the characteristics. Let $\mathbf{d}$ be a vector consisting of P sample points of the inverse channel code EXIT characteristic $\mathcal{C}_{\text{CC}}^{-1}$, measured at the channel quality for which the system is optimized. This channel quality usually is set to be the Shannon limit or slightly above it, such that the optimized system is capacity-achieving. The L weighting factors are grouped into the vector $\mathbf{g}$. The rates of the different index assignments are denoted by r_ℓ^{IA}, $1 \leq \ell \leq L$, which can be grouped to the vector $\mathbf{r} \doteq (r_1^{\text{IA}}, r_2^{\text{IA}}, \ldots, r_L^{\text{IA}})^T$. The overall target rate of the index assignment is denoted by $\overline{r}^{\text{IA}}$.
Three constraints have to be fulfilled:

1. The sum of the L weighting factors g_ℓ has to be equal 1:
$$\sum_{\ell=1}^{L} g_\ell = 1. \tag{4.2}$$
2. The overall rate $\overline{r}^{\text{IA}}$ is the combination of all weighted rates of the subcodes:
$$\sum_{\ell=1}^{L} r_\ell \cdot g_\ell = \overline{r}^{\text{IA}}. \tag{4.3}$$
3. All weighting factors represent a part of the signal, so have to be between 0 and 1:
$$0 \leq g_\ell \leq 1, \qquad \forall \ell \in \{1, \ldots, L\} \tag{4.4}$$

With (4.4) follows (in matrix notation)

$$\mathbf{A}\cdot\mathbf{g} = \begin{pmatrix}\mathbf{I}_L\\ -\mathbf{I}_L\end{pmatrix}\cdot\mathbf{g} = \begin{pmatrix}1 & & \\ & \ddots & \\ & & 1\\ -1 & & \\ & \ddots & \\ & & -1\end{pmatrix}\cdot\begin{pmatrix}g_1\\ \vdots\\ g_L\end{pmatrix} \geq \begin{pmatrix}0\\ \vdots\\ 0\\ -1\\ \vdots\\ -1\end{pmatrix} = \begin{pmatrix}\mathbf{0}_L\\ -\mathbf{1}_L\end{pmatrix} = \mathbf{b} \tag{4.5}$$

with $\mathbf{A} = \begin{pmatrix}\mathbf{I}_L\\ -\mathbf{I}_L\end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix}\mathbf{0}_L\\ -\mathbf{1}_L\end{pmatrix}$, where $\mathbf{1}_L$ is a column vector of the length L filled with ones, $\mathbf{0}_L$ a column vector of length L filled with zeros and $\mathbf{I}_L$ die identity matrix of size $L\times L$. (4.2) and (4.3) can be combined

$$\mathbf{A}_{\text{eq}}\cdot\mathbf{g} = \begin{pmatrix}\mathbf{1}_L^T\\ \mathbf{r}^T\end{pmatrix}\cdot\mathbf{g} = \begin{pmatrix}1 & \cdots & 1\\ r_1 & \cdots & r_L\end{pmatrix}\cdot\begin{pmatrix}g_1\\ \vdots\\ g_L\end{pmatrix} = \begin{pmatrix}1\\ \overline{r}^{\text{IA}}\end{pmatrix} = \mathbf{b}_{\text{eq}} \tag{4.6}$$

with $\mathbf{A}_{\text{eq}} = \begin{pmatrix}\mathbf{1}_L^T\\ \mathbf{r}^T\end{pmatrix}$ and $\mathbf{b}_{\text{eq}} = \begin{pmatrix}1\\ \overline{r}^{\text{IA}}\end{pmatrix}$.

Finally, the optimized curve shall not intersect the inverse characteristic of the channel decoder and an open decoding tunnel has to be present, i.e., the inner and outer decoder characteristics are not allowed to cross because otherwise a error free decoding would not be possible. To achieve this the matrix $\mathbf{C}$ multiplied with the L weighting factors $\mathbf{g} = (g_1,\ldots,g_L)$ has to be bigger than the inverse EXIT-characteristic $\mathcal{C}_{CD}^{-1}$ of the channel decoder $\mathbf{d}$. The matrix $\mathbf{C}$ contains the vectors $\mathbf{c}_\ell = (c_{\ell,1},\ldots,c_{\ell,P})^T$ including the P sample points of the EXIT-characteristics $\mathcal{C}_{\text{SDSD},\ell}$ of the L selectable index assignments. Hence this leads to the equation

$$\mathbf{C}\cdot\mathbf{g} = \begin{pmatrix}\mathbf{c}_1 & \cdots & \mathbf{c}_L\end{pmatrix}\cdot\mathbf{g} = \begin{pmatrix}c_{1,1} & \cdots & c_{L,1}\\ \vdots & \ddots & \vdots\\ c_{1,P} & \cdots & c_{L,P}\end{pmatrix}\cdot\begin{pmatrix}g_1\\ \vdots\\ g_L\end{pmatrix} > \begin{pmatrix}d_1\\ \vdots\\ d_P\end{pmatrix} = \mathbf{d}. \tag{4.7}$$

All the constraints can thus be grouped to

$$\mathbf{C}\cdot\mathbf{g} > \mathbf{d} \tag{4.8}$$
$$\mathbf{A}\cdot\mathbf{g} \geq \mathbf{b} \tag{4.9}$$
$$\mathbf{A}_{\text{eq}}\cdot\mathbf{g} = \mathbf{b}_{\text{eq}} \tag{4.10}$$

with

$$\mathbf{A}_{\text{eq}} = \begin{pmatrix}\mathbf{1}_L^T\\ \mathbf{r}^T\end{pmatrix}, \qquad \mathbf{b}_{\text{eq}} = \begin{pmatrix}1\\ \overline{r}^{\text{IA}}\end{pmatrix} \tag{4.11}$$
$$\mathbf{A} = \begin{pmatrix}\mathbf{I}_L\\ -\mathbf{I}_L\end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix}\mathbf{0}_L\\ -\mathbf{1}_L\end{pmatrix} \tag{4.12}$$

and $\mathbf{1}_L$ denoting the length L all-ones column vector, $\mathbf{0}_L$ the length L all-zeros column vector and $\mathbf{I}_L$ the $L\times L$ identity matrix. Finding the optimal $\mathbf{g}$ is a linear least-squares optimization problem

$$\mathbf{g}_{\text{opt}} = \arg\min_{\mathbf{g}} \|\mathbf{C}\mathbf{g} - \mathbf{d}\|_2 \tag{4.13}$$

subject to (4.8), (4.9) and (4.10). The optimization problem can be solved using numerical algorithms (e.g., [GMW81, TH02a]).

The optimization determines the weights g_ℓ which are the weighting factors of the EXIT characteristics. They also determine the fraction of bits $g_\ell N_X$ to be assigned to each index assignment. From these fractions $g_\ell N_X = g_\ell K_S \overline{w}^*$ the corresponding $K_{S,\ell}$ (number of source parameters assigned to each index assignment) can be determined by

$$K_{S,\ell} = \mathrm{rnd}\left[g_\ell K_S \overline{w}^* \frac{r_\ell^{\mathrm{IA}}}{w}\right] = \mathrm{rnd}\left[g_\ell K_S \frac{r_\ell^{\mathrm{IA}}}{\overline{r}^{\mathrm{IA}}}\right], \tag{4.14}$$

with rnd being an appropriate rounding operation such that $\sum_{\forall \ell} K_{S,\ell} = K_S$.

4.2.1 Design Guideline for Irregular Index Assignments

In the following, we present a simple design guideline in order to generate redundant index assignments with rates $r_\kappa^{\mathrm{IA}} = w/w_\kappa^*$, $w_\kappa^* \in \{w+1, \dots, w_{\max}^*\}$ needed for the optimization of the irregular index assignments. The guideline starts with an (almost) arbitrary generator matrix $\mathbf{G} = (g_{i,j})_{w \times w_{\max}^*}$ of size $\dim \mathbf{G} = w \times w_{\max}^*$ and with elements $g_{i,j} \in \mathbb{F}_2$. A generator matrix $\mathbf{G}_{w^*}$ for a rate $r_\kappa^{\mathrm{IA}} = w/w_\kappa^*$ index assignment is then obtained by

$$\mathbf{G}_{w^*} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{I}_{w^*} \\ \mathbf{0}' \end{pmatrix} \tag{4.15}$$

with $\mathbf{0}'$ denoting here the $(w_{\max}^* - w^*) \times w^*$ all-zero matrix. Equation (4.15) means that $\mathbf{G}_{w^*}$ consists of the first w^* columns of $\mathbf{G}$. 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

$$\mathbf{G} = \begin{pmatrix} \mathbf{I}_w & \mathbf{P} \end{pmatrix}. \tag{4.16}$$

b) The block code generated by $\mathbf{G}_{w+1}$ has a minimum Hamming distance $d_{\min}(\mathbf{G}_{w+1}) \geq 2$.

The second condition is necessary for the EXIT characteristic to reach the $(1,1)$ point [CVA06], [KGM06], [PYH07] and is accomplished if $\mathbf{G}_{w+1}$ realizes a parity check code, i.e., $\mathbf{G}_{w+1} = (\mathbf{I}_w \;\; \mathbf{1}_w)$. The generation of the (irregular) index assignments using a generator matrix as presented in this section enables the receiver to apply a simple yet effective stopping criterion. For details, we refer to [SVCS08].

4.3 Irregular Index Assignment Performance and Unequal Error Protection Properties

In the previous Section, we have given the theoretical background of the concept of irregular index assignments. In this Section, we present a simulation example and show by this example that the concept of irregular index assignments inherently possesses the ability to perform *unequal error protection* (UEP). We furthermore show how to take account of this ability by respecting the source properties and give a modification of the optimization rule.

4.3.1 Simulation Example

We illustrate the generation of irregular index assignments by means of an example, similar to the one given in [SVCS08]. Instead of using any specific speech, audio, or video encoder, we consider blocks consisting of $K_S = 250$ statistically independent source parameters modelled by K_S independent 1st order Gauss-Markov processes with auto-correlation $\rho = 0.9$. The source exhibits inter-frame correlation which is exploited at the receiver using the SDSD described in, e.g., in Section 3.2.2, [Fle08c], or [ACS08], with no backward recursion being carried out. The source parameters are quantized using a $Q = |\mathbb{U}| = 16$ level Lloyd-Max quantizer codebook (see for instance [JN84, VM06]), i.e., $w = 4$. Furthermore, we assume that the overall coding rate of the index assignment shall be of rate $\overline{r}^{\text{IA}} = \frac{1}{2}$, which gives an average number of $\overline{w}^* = 8$ bits per source parameter. The channel code is a memory $J = 3$, rate 1 recursive convolutional code with generator polynomials $G^{\text{CC}}(D) = \left(\frac{1}{1+D+D^2+D^3}\right)$. An exemplary generator matrix $\mathbf{G}$ for $w = 4$ and $w^*_{\text{max}} = 15$, fulfilling the conditions a) and b) introduced in Section 4.2.1 and generating redundant index assignments with rates $4/5, 4/6, \dots, 4/15$ could be

$$\mathbf{G} = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,0\,0\,0 \\ 0\,1\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1\,0\,1 \\ 0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\,1 \end{pmatrix}. \tag{4.17}$$

The generator matrix $\mathbf{G}$ is found by taking all possible permutations of $(1, 0, 0, 0)^T$, $(1, 1, 0, 0)^T$, $(1, 1, 1, 0)^T$ and $(1, 1, 1, 1)^T$ and arranging them as the columns of $\mathbf{G}$ such that the conditions for $\mathbf{G}$ are fulfilled. No special code is searched and optimized as the good properties of the overall system are determined by the irregularity of the index assignment.
The index assignments generated by applying (4.15) to the generator matrix (4.17) for realizing the redundant part of the index assignment Γ^{R}_κ are denoted by $\text{BC}^{Q}_{w^*}$.
Example: The block code index assignment BC^{16}_6 is given by $\text{BC}^{16}_6 = \{\mathbf{x}|\mathbf{x} = \Gamma(\bar{u}), \bar{u} = \bar{u}^{(1)}, \dots, \bar{u}^{(Q)}\} = \{0,\ 6,\ 13,\ 15,\ 23,\ 25,\ 30,\ 36,\ 43,\ 45,\ 50,\ 56,\ 60,\ 66, 73,\ 75\}$ in octal representation with the least significant bit corresponding to $x(w^*)$. A common assumption is that the non-redundant part of the index assignment Γ^{NR} performs a natural binary representation of the codebook indices, i.e., $\Gamma^{\text{NR}}(\bar{u}^{(q)}) = (q-1)_2$, with $(q-1)_2$ denoting the binary representation of $q-1$. For instance, to the quantizer reproduction level $\bar{u}^{(6)}$, the binary representation $\mathbf{x}' = \Gamma^{\text{NR}}(\bar{u}^{(6)}) = (5)_2 = (0101)_2$ is assigned, leading to

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

after the second step Γ^{R}_κ of the redundant index assignment $\Gamma_\kappa(\bar{u}^{(6)}) = \Gamma^{\text{R}}_\kappa(\Gamma^{\text{NR}}(\bar{u}^{(6)}))$.
For an overall rate-$\frac{1}{2}$ transmission with the given parameters, it can be observed that a minimum channel quality of $E_\text{s}/N_0 \approx -4.3\,\text{dB}$ is necessary to reach a reconstruction SNR of the decoded parameters of $\approx 20\,\text{dB}$. This minimum channel quality can be found using the OPTA limit [CSVA06] with an AWGN channel and $\rho = 0.9$. The EXIT characteristics of the channel decoder $\mathcal{C}_{\text{CD}}$ and the characteristics of the different index assignments $\text{BC}^{16}_{w^*}$ are illustrated in Fig. 4.2. It can be seen that the characteristic $\mathcal{C}_{\text{SDSD},4}$ of the regular index assignment BC^{16}_8, meeting the rate requirements $\overline{r}^{\text{IA}} = \frac{1}{2}$, has an intersection with the channel decoder characteristic, resulting in a decoder failure at the given minimum channel quality. The optimization of the irregular index assignment leads to the characteristic $\mathcal{C}^{\text{IIA}}_{\text{SDSD}}$, matching considerably well the channel decoder characteristic with an open decoding tunnel.

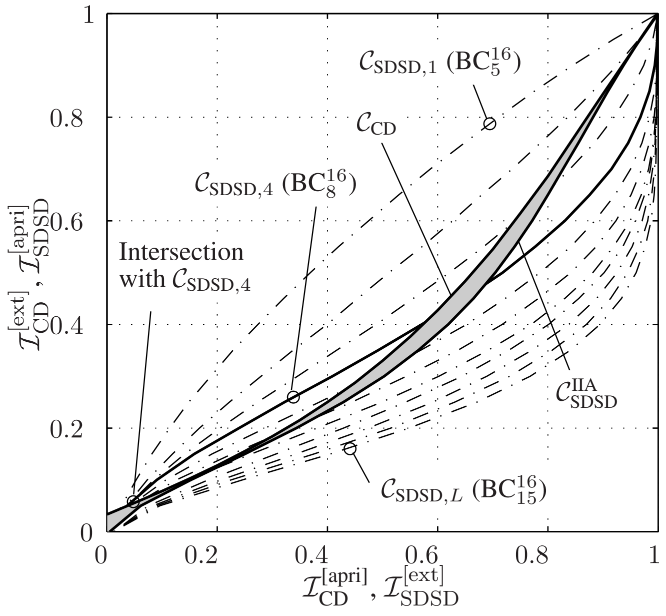


Figure 4.2: EXIT chart analysis of the irregular index assignments at $E_s/N_0 = -4.3\,\text{dB}$

Rate r_ℓ^{IA}	Γ_ℓ	g_ℓ	$g_\ell K_S \overline{w}^*$	$K_{S,\ell}$
$4/5$	BC$_5^{16}$	0.375	750	$K_S^{(4/5)} = 150$
$4/6$	BC$_6^{16}$	0.045	90	$K_S^{(4/6)} = 15$
$4/10$	BC$_{10}^{16}$	0.115	230	$K_S^{(4/10)} = 23$
$4/15$	BC$_{15}^{16}$	0.465	930	$K_S^{(4/15)} = 62$
$\overline{r}^{\text{IA}} = \frac{1}{2}$		$\sum = 1$	$\sum = K_S \overline{w}^* = 2000$	$\sum = K_S = 250$

Table 4.1: Result of the irregular index assignment example

The results of the optimization and the calculated portions $K_{S,\ell}$, which have been determined using (4.14), are summarized in Table 4.1. 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 four of them.
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 $Q = 2^w$ in this contribution).
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 MMSE estimation rule given in (3.20) is utilized. The simulation results are depicted in Fig. 4.3. Two conventional non-iterative transmission systems are also given as a reference. In both reference systems, the channel code is a rate $r^{\text{CC}} = \frac{1}{2}$ recursive systematic convolutional code with generator polynomial $G_{\text{RSC}}^{\text{CC}}(D) = \left(1, \frac{1+D+D^3}{1+D+D^2+D^3}\right)$. The code can be decoded using a (soft-input, hard-output) Viterbi decoder: Then the source symbols $\hat{u}$ are reconstructed using a simple table lookup (Reference A). On the other hand, the code can also be decoded using a LogMAP decoder followed by a soft decision source decoder (Reference B). In both cases, a natural binary index assignment is utilized. No additional

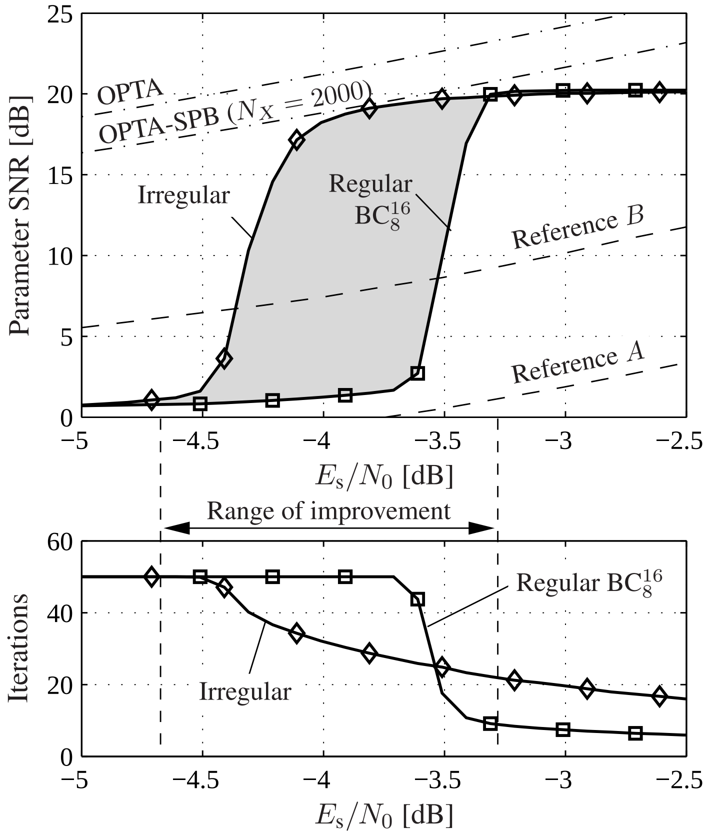


Figure 4.3: Parameter SNR and mean number of iterations for a system with regular and irregular index assignments

redundancy is introduced by the index assignment.

While the ISCD system utilizing the regular index assignment BC_8^{16} achieves considerable gains compared to the references and is already able to closely reach the theoretical OPTA limit, additional gains of $\approx 0.7\,\mathrm{dB}$ can be obtained by using the irregular index assignment. 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 theoretical OPTA-SPB limit [CSVA06] in a wider range of channel conditions.

The maximum number of iterations has been fixed to 50 for both systems. The average number of utilized iterations at the receiver is depicted in the lower part of Fig. 4.3. The number of iterations rapidly decreases in the waterfall region if the regular index assignment is used. The system employing *irregular index assignments* (IIA) needs more iterations due to the narrow decoding tunnel. As can be seen in Fig. 4.3, less than 50 iteration would be enough for the system using the regular index assignment because of the early intersection of the EXIT characteristics and the large open tunnel which can be observed in good channel conditions. However, the utilized stopping criterion limits the number of iterations to the necessary minimum. During bad channel conditions, the proposed stopping criterion is not useful as the parity equations are never fulfilled. In this case, *early-termination* mechanisms, like the ones proposed in [LW07], could be used. If the channel quality is known simultaneously at the receiver and at the transmitter, it can be beneficial to change the index assignment at some point in order to only use the irregular index assignments if a gain in terms of reconstruction quality can be achieved (i.e., range of improvement for $-4.7\,\mathrm{dB} < E_\mathrm{s}/N_0 < -3.3\,\mathrm{dB}$).

4.3.2 Unequal Error Protection

In most modern speech, audio, and video codecs, some parameters are more important than others. Errors in the less important parameters can easier be tolerated because the adverse effects on the quality of the reconstructed audiovisual signal are less annoying. Therefore, a higher symbol error rate, or a lower parameter SNR respectively, can be allowed for those parameters. On the other hand, more important parameters, like, e.g., gain factors need to be better protected. This leads to *Unequal Error Protection* (UEP), where several types of parameters get different levels of error protection.

The proposed ISCD system with irregular index assignments directly incorporates unequal error protection. It can be observed that the parameters which are assigned to the high-rate index assignments have a slightly lower parameter SNR after decoding than those parameters which are assigned a low-rate index assignment. This is visualized in Fig. 4.4 where the parameter SNR after decoding has been measured for each of the 250 parameters in one block in the simulation example described in Section 4.3.1 at channel qualities of $E_\mathrm{s}/N_0 = -4.25\,\mathrm{dB}$ and $E_\mathrm{s}/N_0 = -4\,\mathrm{dB}$ (see Fig. 4.3). The utilized irregular index assignment is summarized in Table 4.1. The allocation of parameters to index assignments has been defined as follows: The first parameters of the block are encoded using the high-rate index assignments (rates $4/5$, $4/6, \ldots$) and the parameters towards the end of the block are encoded using the low-rate index assignments. It can be seen that the first $K_{S,1} + K_{S,2} = 165$ parameters encoded with rate $4/5$ and $4/6$ index assignment show a lower reconstruction parameter SNR than the $K_{S,3} = 23$ parameters encoded using the rate $4/10$ index assignment. The $K_{S,4} = 62$ parameters encoded using the rate $4/15$ index assignment show the best reconstruction quality. Note that this behavior only occurs in the waterfall region. This region is quite narrow in the simulation example in Section 4.3.1 ($-4.5\,\mathrm{dB} \leq E_\mathrm{s}/N_0 \leq 3.7\,\mathrm{dB}$), however, if less iterations are performed at the receiver (for instance because of complexity constraints), the waterfall region can be wider. For good channel qualities ($E_\mathrm{s}/N_0 > 3.7\,\mathrm{dB}$ in the above example), all parameters can be assumed to be perfectly decoded.

During system design, it is therefore advantageous to employ the low-rate index assignments for parameters that are sensitive to transmission errors. On the other hand the high-rate index assignments can be utilized for those parameter where decoding failures result in a low subjective quality degradation in the

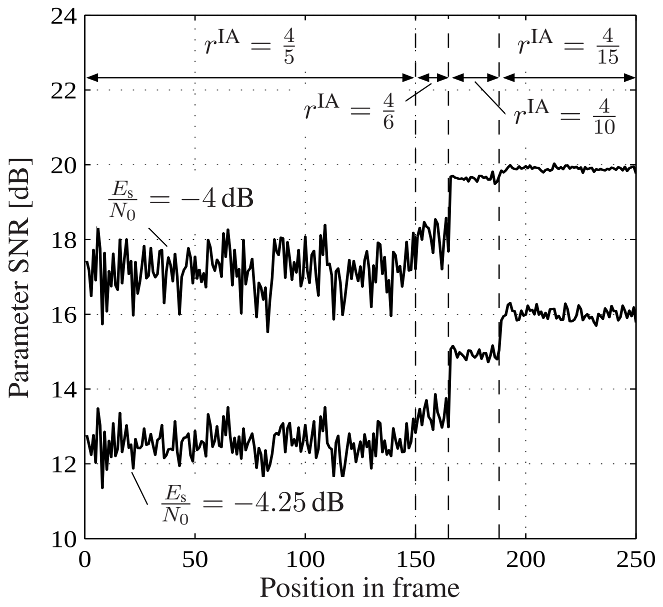


Figure 4.4: Unequal error protection capabilities for $E_\mathrm{s}/N_0 = -4.25\,\mathrm{dB}$ and $E_\mathrm{s}/N_0 = -4\,\mathrm{dB}$

reconstructed audio or video signal.
If the source properties are known, the optimization of the irregular index assignments can also be modified such that the unequal error protection requirements of the source are incorporated into the optimization as additional constraints. We illustrate the procedure by a small example. Suppose that only a fraction f_L of the source parameters are less sensitive to transmission errors. Therefore, at most $f_L K_S$ parameters must be encoded using the high-rate index assignments: For instance, if we fix the condition that less than $f_L K_S$ parameters are encoded with the index assignments of rates $4/5$ and $4/6$ we get

$$K_{S,1} + K_{S,2} \leq f_L K_S \, . \tag{4.18}$$

If the rnd operation in (4.14) is neglected, substituting the different $K_{S,\ell}$ in (4.18) by their expressions given in (4.14), we get

$$g_1 r_1^{\mathrm{IA}} + g_2 r_2^{\mathrm{IA}} \leq f_L \overline{r}^{\mathrm{IA}} \, . \tag{4.19}$$

On the other hand, if the source shows the property that at least $f_H K_S$ parameters need high error protection and shall be encoded using the 3 lowest-rate index assignments, we get

$$K_{S,L-2} + K_{S,L-1} + K_{S,L} \geq f_H K_S \tag{4.20}$$

which transforms to

$$g_{L-2} r_{L-2}^{\mathrm{IA}} + g_{L-1} r_{L-1}^{\mathrm{IA}} + g_L r_L^{\mathrm{IA}} \geq f_H \overline{r}^{\mathrm{IA}} \, . \tag{4.21}$$

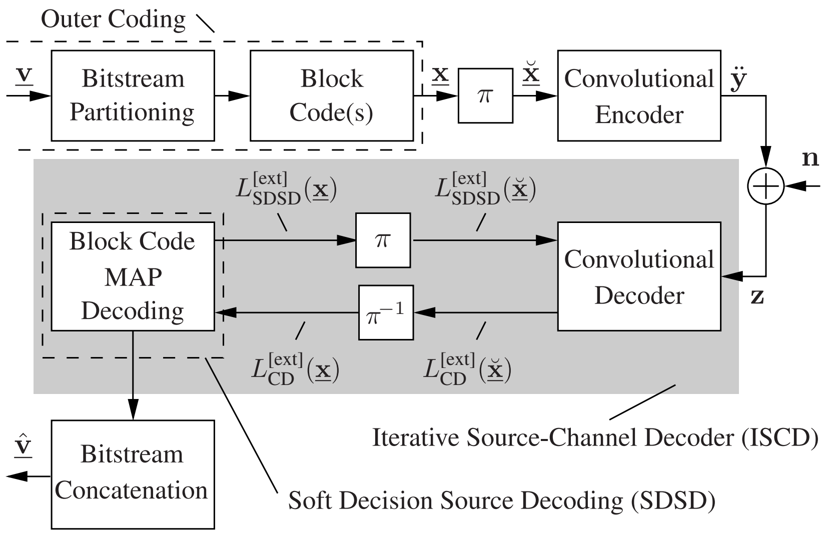
The constraint $\mathbf{A}\mathbf{g} \geq \mathbf{b}$ (see Equation (4.9)) of the least squares optimization problem (4.13) can now be modified in order to incorporate (4.19) and (4.21)

$$\mathbf{A} = \begin{pmatrix} \mathbf{I}_L \\ -\mathbf{I}_L \\ -(1\,1\,0\cdots 0) \odot \mathbf{r}^T \\ (0\cdots 0\,1\,1\,1) \odot \mathbf{r}^T \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \mathbf{0}_L \\ -\mathbf{1}_L \\ -f_L \cdot \overline{r}^{\mathrm{IA}} \\ f_H \cdot \overline{r}^{\mathrm{IA}} \end{pmatrix} \tag{4.22}$$

with $\odot$ denoting the element-wise multiplication of two vectors.
This illustrative example has shown how to incorporate properties of the source into the optimization procedure for irregular index assignments. In this example, unequal error protection has been realized by defining the maximum number of parameters that have weak error protection and a minimum number of parameters having strong error protection. Of course, more different levels of protection can be defined leading to more additional inequality constraints. However, the more constraints are defined, the more difficult it becomes to find a solution to the optimization problem (4.13). Also note that unequal error protection only occurs in the waterfall region ($-4.5\,\mathrm{dB} < E_\mathrm{s}/N_0 < -3.7\,\mathrm{dB}$ in the simulation example in Fig. 4.3).

4.4 Channel Coding of a Generic Bit Stream Using Iterative Source-Channel Decoding with Irregular Index Assignments

During the development of the *FlexCode* channel coder, it has been found that the developed ISCD approach cannot be used if arithmetic coding is used as there is no direct relationship between source codec parameters (i.e., transform coefficients) and bit stream. In order to utilize the ISCD approach also in this case and to prevent the development of a dedicated channel coding concept for the arithmetically coded bit stream, it has been decided to modify the ISCD approach for channel coding generic bit streams.
The modified block diagram is given in Fig. 4.5. Instead of source codec parameters $\underline{u}$, now a bit stream $\underline{\mathbf{v}}$ is to be encoded and transmitted. Therefore, the quantizer in Fig. 4.1 of the ISCD system



Outer Coding
$\underline{\mathbf{v}}$
Bitstream Partitioning
Block Code(s)
$\underline{\mathbf{x}}$
π
$\breve{\underline{\mathbf{x}}}$
Convolutional Encoder
$\ddot{\mathbf{y}}$
$\mathbf{n}$
$L^{[\text{ext}]}_{\text{SDSD}}(\underline{\mathbf{x}})$
$L^{[\text{ext}]}_{\text{SDSD}}(\breve{\underline{\mathbf{x}}})$
Block Code MAP Decoding
π
Convolutional Decoder
π^{-1}
$\mathbf{z}$
$L^{[\text{ext}]}_{\text{CD}}(\underline{\mathbf{x}})$
$L^{[\text{ext}]}_{\text{CD}}(\breve{\underline{\mathbf{x}}})$
$\hat{\underline{\mathbf{v}}}$
Bitstream Concatenation
Iterative Source-Channel Decoder (ISCD)
Soft Decision Source Decoding (SDSD)

Figure 4.5: Modified baseband model of the ISCD system for generic channel coding

model is replaced by a bit stream partitioner which demultiplexes the bit stream (which is the output of the arithmetic encoder). Then each individual group of bits is encoded by a small (n, k) block code (generator matrix with a small number of rows, i.e. $k < 10$). After demultiplexing, the collected block coded groups are interleaved and encoded by a (rate-1) convolutional code prior to transmission over the channel.

The receiver remains unchanged and consists of the serial concatenation of a convolutional decoder (which is a MAP decoder [BCJR74], [RVH95]) and a *Soft Decision Source Decoder* (SDSD). As no *a priori* knowledge on the different groups of bits is generally available, the SDSD does not exploit any redundancies and therefore reduces to a MAP block code decoder. After performing a fixed (or variable) number of iterations, the MAP input bit group is determined and by concatenation an estimated bit stream is generated again which can then be arithmetically decoded.

To reach the Shannon limit for a given channel and a serial Turbo concatenation it is necessary to fit the EXIT-characteristic of the inner channel decoder to an outer component. This can e.g. be achieved with irregular codes using different codes with possibly different rates in one frame. The principle of irregular block codes is based on the feature that a EXIT-characteristics can be determined by the weighted sum of characteristics of all part codes within one frame. Using this criterion it is possible to perfectly fit an outer component to an inner one [Tüc04], [TH02a] and therefore adapt the code to the channel properties. This way offers a great flexibility. This is basically the same concept as the *irregular index assignments* introduced earlier in this chapter and the same optimization procedures (given in Section 4.2) can be applied.

This scheme corresponds to a serial Turbo scheme consisting of the concatenation of an outer block code and an inner convolutional code. The block code has the generator matrix

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_1 & & & & & \\ & \mathbf{G}_2 & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \mathbf{G}_{\Upsilon-1} & \\ & & & & & \mathbf{G}_{\Upsilon} \end{pmatrix}$$

if Υ different small blocks are utilized. The overall EXIT characteristic of the outer component consisting of this block code is given by the weighted superposition of the EXIT characteristic of the Υ different subcodes. This can be interpreted again as a problem of irregular codes and the previously introduced optimization procedure can be applied.

Example Assume following generator matrix:

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & \mathbf{G}_1 & & & & & & \\ & & & \mathbf{G}_2 & & & & & \\ & & & & \ddots & & & & \\ & & & & & \mathbf{G}_2 & & & \\ & & & & & & \mathbf{G}_3 & & \\ & & & & & & & \ddots & \\ & & & & & & & & \mathbf{G}_3 \end{pmatrix}$$

with

$$\mathbf{G}_1 = \left(\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \end{array} \right),$$

$$\mathbf{G}_2 = \left(\begin{array}{cc|ccc} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{array} \right),$$

$$\mathbf{G}_3 = \left(\begin{array}{ccc|cc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{array} \right).$$

To achieve a encoding rate $\overline{r} = 1/2$, one possibility would be to weight $g_1 = 1/4$ of the overall coded bits with G_1, $g_2 = 7/12$ with G_2 and $g_3 = 1/6$ with G_3, because then the constraint (4.2) for irregular index assignments (and thus irregular block codes) is fulfilled with

$$\frac{1}{4} + \frac{7}{12} + \frac{1}{6} = 1.$$

Similarly the constraint (4.3) with

$$\frac{1}{4} \cdot \frac{2}{3} + \frac{7}{12} \cdot \frac{2}{5} + \frac{1}{6} \cdot \frac{3}{5} = \frac{1}{2}$$

and the constraint (4.4) with

$$0 \leq \frac{1}{6} \leq \frac{1}{4} \leq \frac{7}{12} \leq 1$$

are also fulfilled.
Like the optimization in Section 4.2, the different EXIT characteristics of the block codes can be grouped in a matrix $\mathbf{C}$ and then the optimal weighting factors $\mathbf{g}$ can be found by the least-squares optimization problem

$$\mathbf{g}_{\text{opt}} = \arg\min_{\mathbf{g}} ||\mathbf{C} \cdot \mathbf{g} - \mathbf{d}||_2 \qquad (4.23)$$

subject to (4.5) and (4.6). Numerical methods, as for instance given in [Tüc04], [TH02b], [GMW91], [GMW81] can be used to solve this problem.

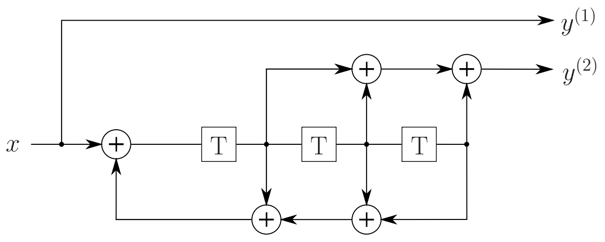


Figure 4.6: (17,7) – convolutional code with doping rate 100 (1 output bit of y_1 to 100 output bits of y_2)

The optimization process calculates die weightings g_ℓ, which match the weighting factors for the EXIT-characteristics. The number of bits $N(g_\ell)$, which are encoded with the ℓ'th block code, are calculated by

$$N(g_\ell) = \mathsf{rnd}\left[g_\ell\,\mathcal{V}\,\overline{w}^*\frac{r_\ell}{w}\right] = \mathsf{rnd}\left[g_\ell\,\mathcal{V}\,\frac{r_\ell}{\overline{r}}\right]$$

with

$$\sum_{\forall\ell} N(g_\ell) = \mathcal{V},$$

where the rounding operation rnd has to be chosen in such a way that $\mathcal{V}$ is equal to the number of input bits.
The chosen implementation rounds every $N(g_\ell)$ down to the last complete block and encodes the left input bits with the last generator matrix G_L. The receiver, knowing the overall rate $\overline{r}$ and the number of received bits, calculates the approximately number of input bits. With the implementation used at the encoder the decoder now verifies if the calculated number of input bits results in the number of received bits. If this is not the case the number of input bits is decreased by one and the operation is repeated till an unique assignment is found. A disadvantage of this method is that the number of input bits can not be calculated exactly but only the assignment of the block codes at the receiver. This is not a big issue if encoder and decoder agree to have e.g. a '1' as the last bit and cut off the last '0's. The source decoder have to add a certain amount of '0's in that case.

4.4.1 Simulation Example

The approach will now be clarified by a complete example. As an inner component we select a doped recursive non-systematic $(17,7)$ convolutional code according to [tB00] shown by Figure 4.6. Doping in this case means that the output is almost exclusively determined by y_2. Only every i. bit for a doping rate of 1:i is substituted by a bit from output y_1. In the example we use a doping rate of 1:100 so every 100'th output bit is substituted.
For block codes of different length with different number of parity bits we have pre-calculated SDSD-EXIT-characteristics to choose from. In this example block codes with $1, 2$ or 3 input bits each with 1 to 9 parity bits were available. The target rate should be $\overline{r} = 1/2$. The characteristic for the inner component is recorded for a $E_s/N_0 = -2.82\,\text{dB}$, close to the Shannon limit for a rate $r = 1/2$.
Table 4.2 shows the result of the optimization problem (4.23) with the following generator matrices:

$$\mathbf{G}_1 = \left(\begin{array}{c|ccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right),\quad \mathbf{G}_2 = \left(\begin{array}{cc|cccc} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right),$$

$$\mathbf{G}_3 = \left(\begin{array}{cc|ccc} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{array}\right),\quad \mathbf{G}_4 = \left(\begin{array}{cc|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right),\quad \mathbf{G}_5 = \left(\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right)$$

generator matrix	rate r_ℓ	g_ℓ
$\mathbf{G}_1$	$1/10$	0.026
$\mathbf{G}_2$	$2/6$	0.336
$\mathbf{G}_3$	$2/5$	0.105
$\mathbf{G}_4$	$2/4$	0.160
$\mathbf{G}_5$	$2/3$	0.372
	$\overline{r} = 1/2$	$\sum = 1$

Table 4.2: Result of the optimization problem (4.23)

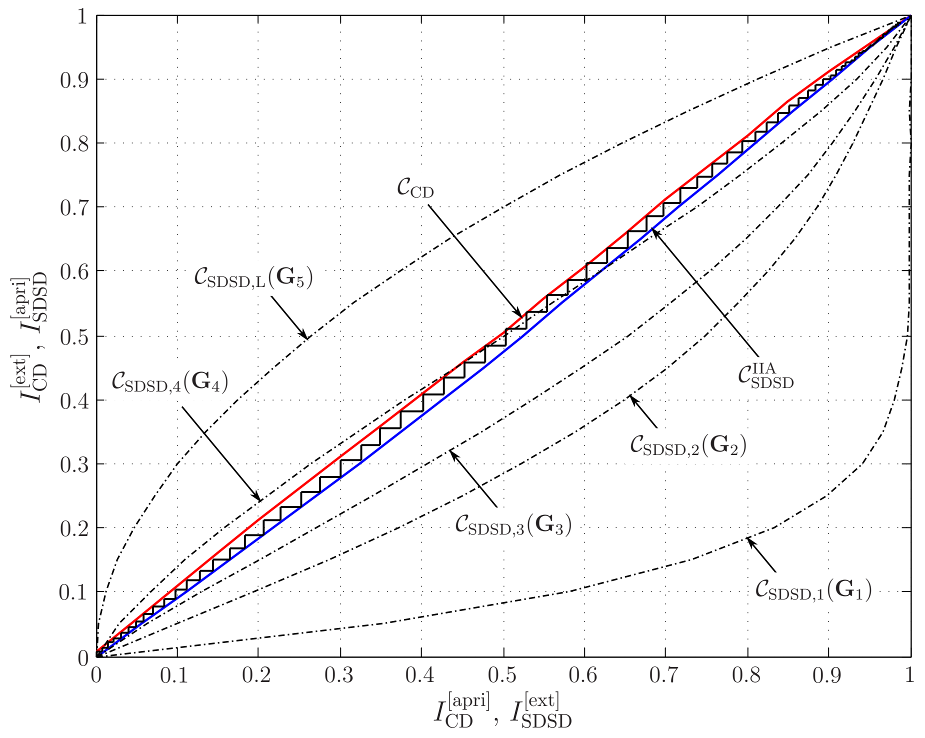


Figure 4.7: EXIT chart with characteristics for channel decoder, SDSD of different block codes, as well as for the irregular SDSD

Figure 4.7 shows the different SDSD-EXIT characteristics of the selected block codes as dashed lines as well as their weighted combination as blue line. The upper right corner can be reached through an open decoding tunnel to achieve error free decoding.

The curves in Figure 4.8 represent the bit error rate dependent on E_s/N_0 for different numbers of iterations. It can be seen that for high number of iterations very good results can be achieved. At a error rate of 10^{-4} we are just $0.2\,\mathrm{dB}$ away from the Shannon limit. However, this bound is only valid for blocks with unlimited length. So we also have a look at the Sphere-Packing Bound (SPB) by [Sha59]. [DDP98] gives a good overview which we want to summarize as follows.

The code word error probability P_w for a code block of length $\mathcal{X}$ is described for the SPB by:

$$P_w \geq f(\mathcal{X}, \theta_s, A),$$

where

$$A = \sqrt{2\,E_s/N_0} = \sqrt{2\,\overline{r}\,E_B/N_0}$$

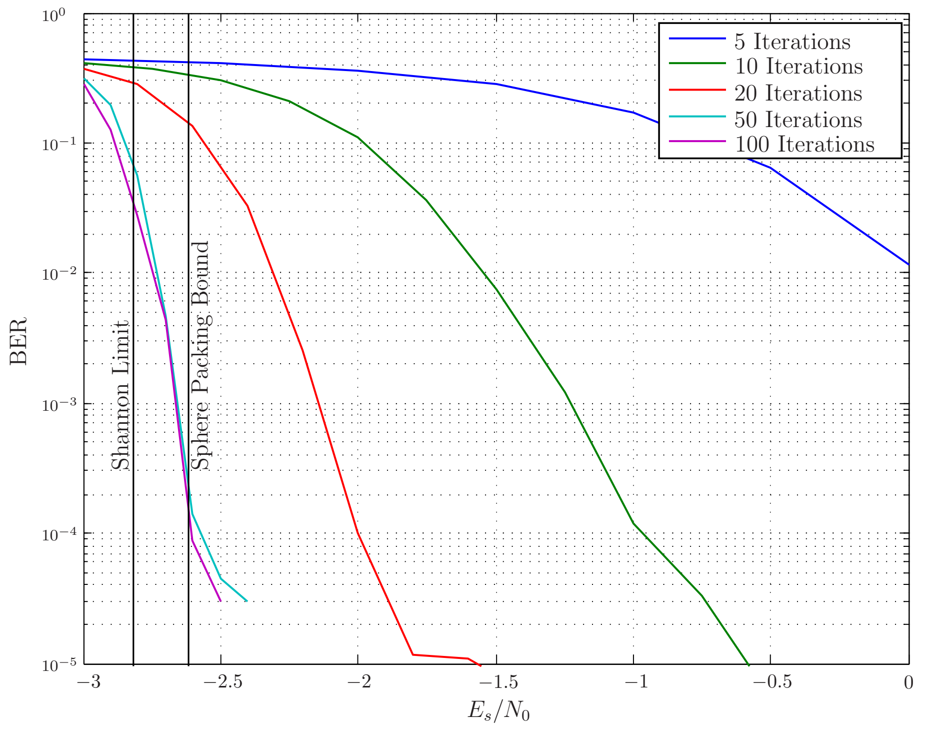


Figure 4.8: Bit error rate of the proposed system with 5, 10, 20, 50 and 100 iterations

is the amplitude depending on the channel quality. Then, $f(\mathcal{X}, \theta, A)$ gives the probability that a $\mathcal{X}$-dimensional Gaussian random vector with mean $(A, 0, \dots, 0)$ and with the identity matrix as covariance falls outside a $\mathcal{X}$-dimensional cone of half-angle θ around its mean. If this cone encompasses a fraction of $1/2^{\mathcal{V}}$ of the total solid angle of the X-dimensional Euclidean space, its half-angle is denoted as θ_s. The information block size $\mathcal{V}$ is given by

$$\mathcal{V} = \overline{r}\mathcal{X}.$$

If $\mathcal{V}$ is big enough ($\mathcal{V} \gtrsim 100$), we can use asymptotic approximations of [DDP98]. In this case the angle θ_s is

$$\theta_s \approx \arcsin(2^{-\overline{r}}).$$

Hence it follows

$$f(\mathcal{X}, \theta, A) \approx \frac{[g(\theta, A)\sin(\theta)\exp(-(A^2 - A\,g(\theta, A)\cos(\theta))/2)]^{\mathcal{X}}}{\sqrt{\mathcal{X}\pi}\sqrt{1 + g^2(\theta, A)}\sin(\theta)(A\,g(\theta, A)\sin^2(\theta) - \cos(\theta))},$$

with

$$g(\theta, A) = \frac{1}{2}\left(A\cos(\theta) + \sqrt{A^2\cos^2(\theta) + 4}\right).$$

If we consider a code block length $\mathcal{V} = 20000$ and a error probability of $P_w = 10^{-4}$ like used in the simulation shown in Figure 4.8, the Shannon limit is moved by $+0.203\,\mathrm{dB}$. This bound is also shown in Figure 4.8 and crosses the curves for 50 and 100 Iterations approximately at the given symbol error probability of 10^{-4}.

4.4.2 Irregular Block Codes for Short Code Block Length

To proof the eligibility of the proposed *FlexCode* coding concept, the simulation has also be done for smaller code block lengths. The doped convolutional code from the previous section leads, especially

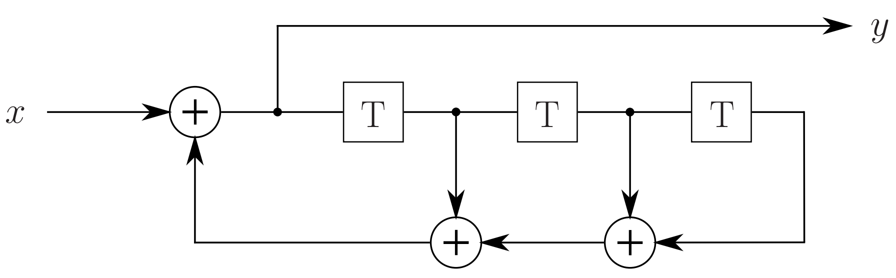


Figure 4.9: Recursive, non-systematic (17,10) convolutional code

for small code block lengths, to very bad results, so we used the rate $r = 1$, $(17, 10)$ convolutional code shown in Figure 4.9 and a irregular SDSD optimized for this scenario with the weighting factors of table 4.3 and the generator matrices

$$\mathbf{G}_1 = \left(\begin{array}{cc|cccc} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right), \quad \mathbf{G}_2 = \left(\begin{array}{cc|ccc} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{array}\right),$$

$$\mathbf{G}_3 = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right).$$

Figure 4.10 show the results of this system with 50 iterations. You can see that especially for the code block length between 500 and 1000 the error floor starts at a high bit error rate.
For better comparison with the *FlexCode* reference system described in Deliverable D-2.2 [Fle08c], we simulated the proposed and the reference system with a fixed code block length of 700 for 5, 10, 20 and 50 iterations. The results are shown in Figure 4.11. The parameter S of the interleaver was set to $S = 14$ for both systems. The following observations can be made:

- The system with irregular block codes considerably benefits from the increasing of the iterations, while the Turbo Code system just profits minimal. This can be explained with a wider open decoding tunnel in the EXIT-Chart for higher E_S/N_0 in the reference system.
- The error floor of the Turbo Code system begins later and lies deeper by factor 10 compared to the system using irregular block codes.

Applied to the concept of irregular block codes shown in Figure 4.5, where the bit stream $\underline{\mathbf{v}}$ is the output of the arithmetic coder of the *FlexCode* source coder, this leads to results shown in Figure 4.12. Again the irregular block code system needs a very high number of iterations to be as good as the Turbo system, but with a higher error floor. It also introduces a higher computational complexity.

generator matrix	rate r_ℓ	g_ℓ
$\mathbf{G}_1$	$2/6$	0.150
$\mathbf{G}_2$	$2/5$	0.580
$\mathbf{G}_3$	$3/4$	0.270
	$\overline{r} = 1/2$	$\sum = 1$

Table 4.3: Result of the optimization problem (4.23) for the non-doped $(17, 10)$ convolutional code

In addition it has to be noted that for the simulation shown in Figure 4.12 the S parameter of the interleaver was set to $S = 20$ because the inner component is a rate $r = 1$ convolutional code and therefore the code block length felt not below 1000 bit. Otherwise the results would have been even worse in comparison to the Turbo system, which has a minimal code block length of 500.

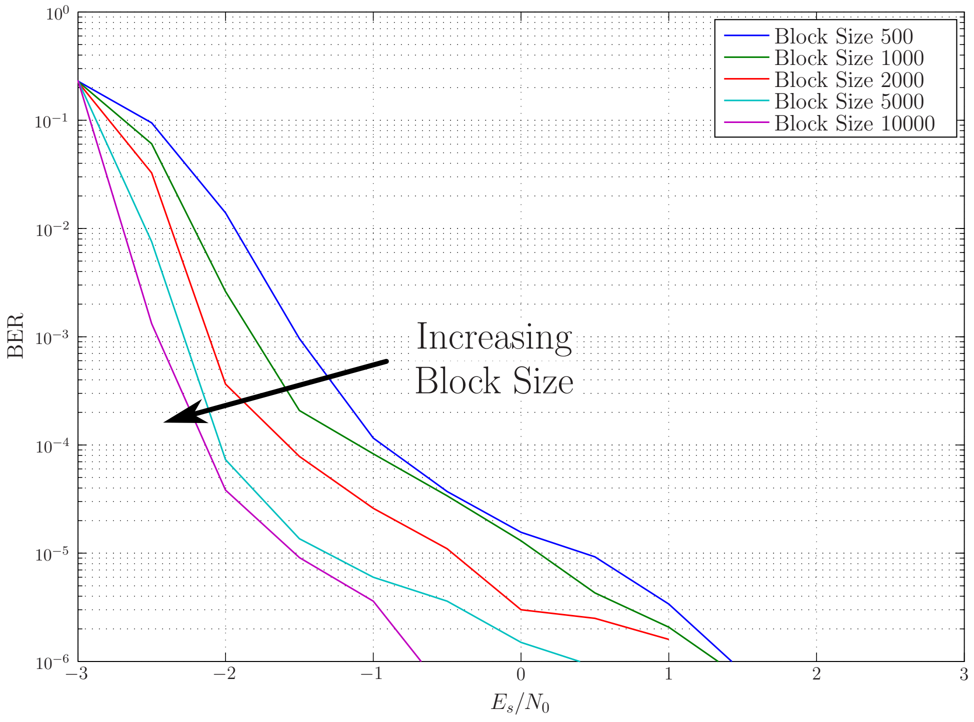


Figure 4.10: Bit error rate for irregular block code in combination with non-doped convolutional code with code block length of 500, 1000, 2000, 5000 and 10000 bits and 50 iterations

4.5 Error-Resilient Source Compression Using Iterative Source-Channel Decoding

In this Section, we show how the proposed system for iterative source-channel decoding (ISCD) (see Deliverable D-2.2 [Fle08c] and, e.g., [ACS08] for details) can be efficiently utilized to realize a near-entropy entropy coding scheme resilient to transmission errors. The conventional entropy coding schemes like, e.g., arithmetic coding [BCK07], are very sensitive to bit errors. A single bit error can cause the complete frame to be wrongly decoded. Therefore a strong channel coding algorithm is necessary which however results in additional bit rate. The approach we introduce here adds only as much redundancy as necessary to achieve the reconstruction, given a channel with known quality. However, the approach does not guarantee perfect lossless reconstruction but only near-lossless reconstruction. However, in real-world codecs a perfect lossless reconstruction is not absolutely necessary as the source decoder artifacts caused by an incorrectly decoded parameter can be negligible.

In [MB02] and [GZ92], it has been shown that Turbo codes can also be used as source encoders. Conventional entropy source encoders such as Huffman codes or arithmetic codes are very sensitive to transmission errors while the Turbo source coding approach automatically incorporates error protection and can adapt on the fly to changing channel conditions by increasing or decreasing the amount of artificial redundancy introduced by the channel code. Iterative source-channel decoding has also been applied to

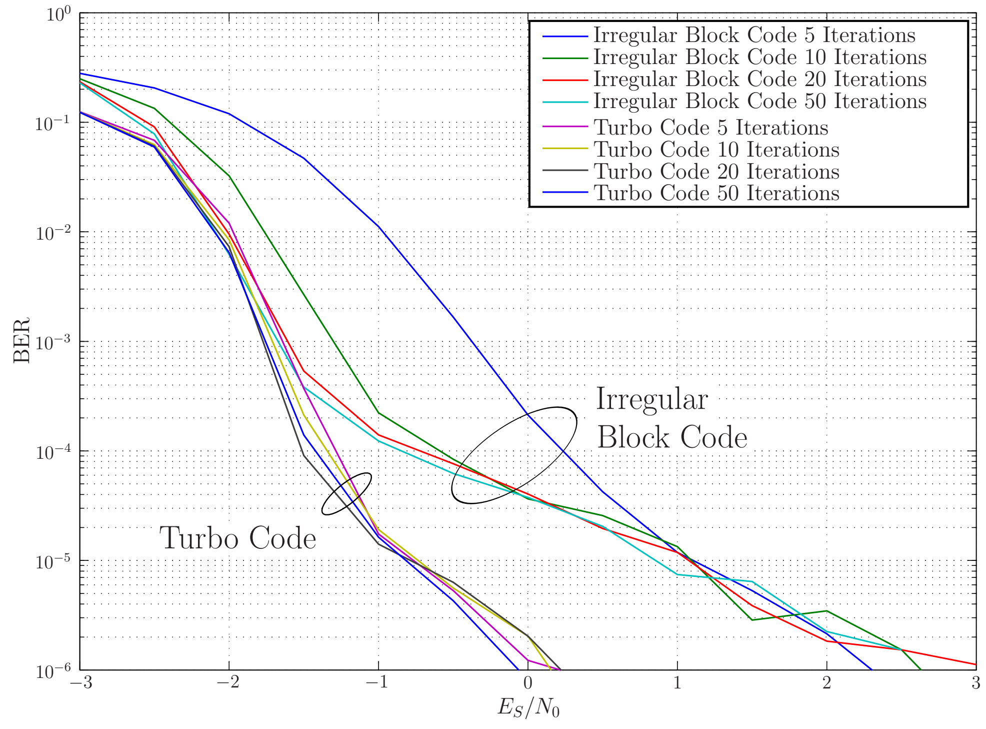


Figure 4.11: Bit error rate for a code block length of 700 for irregular block code in combination with non-doped convolutional code in comparison to parallel Turbo Code reference system with 5, 10, 20 and 50 iterations

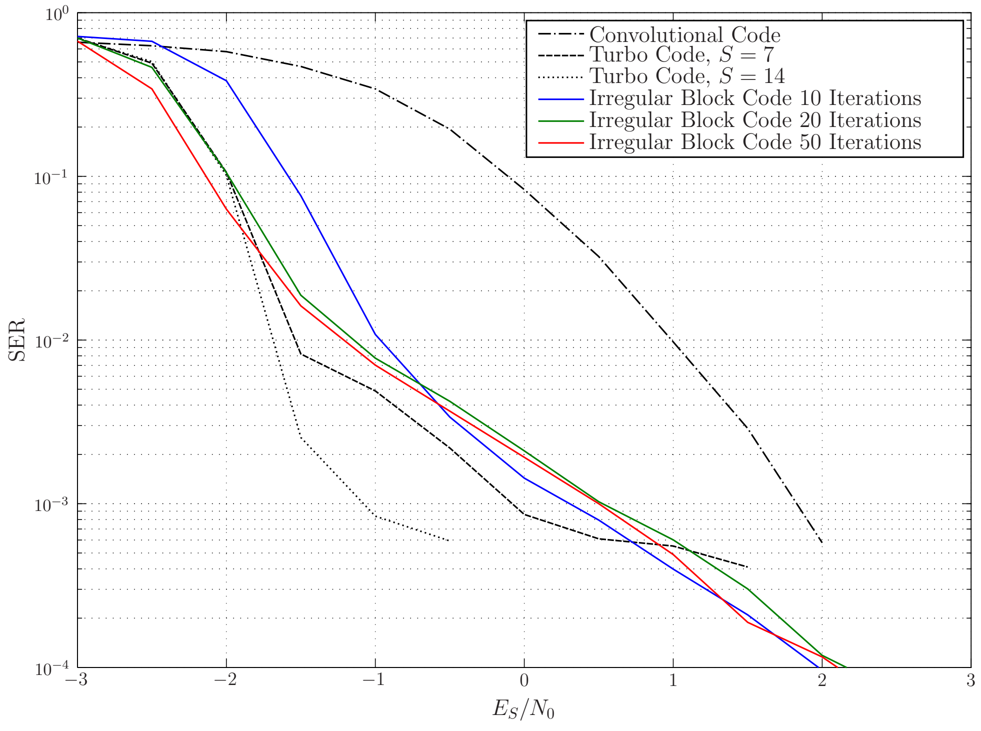


Figure 4.12: Symbol error rate of reference coder described in Deliverable D-2.2 [Fle08c] for usage of irregular block code concept in comparison to Turbo Code

variable-length codes (VLCs) [GFGR01], however, it has already been shown in [Tho07], [TSV08] that the use of a system with *fixed-length codes* (FLCs) and redundant index assignments leads to a lower-complexity receiver while keeping the reconstruction quality as well as the amount of transmitted data (almost) constant.

In this Section we introduce a novel concept for near-lossless compression of scalar-quantized source codec parameters. This concept uses a joint source-channel coding approach with ISCD at the receiver, similar to the Turbo source coding principle. The inner (channel) code of the transmitter is of rate $r \geq 1$, such that the system becomes capacity-achieving [AKtB04], [Tho07]. If this inner (channel) code is fixed, the outer code, i.e., the (redundant) index assignment of the different parameters can be matched quite well to the inner code using the principles of irregular codes [TH02a], allowing a simple optimization using EXIT charts [ten01]. The concept of irregularity has been successfully applied to the ISCD system [SVCS08] by modifying the (redundant) index assignment, i.e., the assignment of bit patterns to codebook indices, of a (scalar) quantizer to get so-called *irregular index assignments* (IIA). These irregular index assignments extend the concept of redundant index assignments [CVA06]. In this contribution, we show that the optimization of the irregular index assignments can be modified such that the compression ratio is maximized, leading to an efficient, flexible compression system which can easily adapt to varying channel conditions.

In this Section we utilize the abstract system model already introduced in Section 4.1 for the demonstration of the ISCD concept. This system model is similar to the one utilized in Section 3.1. It will be shown in Section 4.5.1 how the concept of irregular index assignments can be used to realize an error-resilient near-lossless entropy coder. This is achieved by modifying the optimization criterion of the index assignment. The capabilities of the proposed system are shown by a simulation example in Section 4.5.2.

4.5.1 Near-Lossless Source Coding Using Irregular Index Assignments

It has been shown in, e.g., [MB02] and [GZ92], that Turbo codes can be efficiently used as source codes, realizing a near-lossless compression scheme. Near-lossless means that the perfect reconstruction is not guaranteed. Classical entropy coding compression schemes like Huffman or arithmetic codes are able to achieve high compression ratios with moderate computational complexity, however, in the presence of channel noise, severe error propagation and synchronization losses are possible. Soft decision source decoding and iterative source-channel decoding can also be applied to entropy codes [GFGR01], with increased computational complexity however. It has been shown in [Tho07] that with lower computational complexity at the receiver, a system utilizing fixed-length codes can achieve comparable results (in terms of reconstruction quality and symbol error rate) to a system with variable-length codes, in the case that channel noise is present.

The ISCD system using redundant index assignments introduced in Section 4.1 is used in this paper for realizing a near-lossless source coding system. In this case, the convolutional code is a rate $r^{\mathrm{CC}} > 1$ code obtained by puncturing a rate < 1 code and the index assignment has to be optimized such that a minimum number of transmitted bits $N_Y = (N_X + J) \cdot \frac{1}{r^{\mathrm{CC}}}$ results.

Thus, the task of the source coder is to find an index assignment which minimizes the number of transmitted bits and allows near-lossless decoding of the parameters at the receiver/decoder. The approach presented here is based on the concept of *Irregular index assignments* (IIA), introduced in [SVCS08] and is an extension of the irregular codes' concept. As stated in Section 4.1, the index assignment for the parameter u_κ comprises a block code $\Gamma_\kappa^{\mathrm{R}}$ of rate $r_\kappa^{\mathrm{IA}} = (\log_2 Q)/w_\kappa^* = w/w_\kappa^*$. Instead of using the same amount of bit redundancy $w_\kappa^* = \overline{w}^*$ for each parameter in order to achieve an overall rate $w/\overline{w}^*$ outer encoding, we use the concept of irregular codes and vary w_κ^* for each parameter while keeping $\overline{w}^*$ constant. This allows us to optimize the index assignments and to get an SDSD EXIT characteristic

which matches the channel decoder characteristic considerably well.
In order to perform source coding, the optimization goal is a different one as in [TH02a] and [SVCS08]. The index assignments which have a high rate (and thus a small number of output bits) shall be preferred. Therefore, the goal of the optimization is to find an EXIT characteristic which results in the smallest number of transmitted bits with the constraint that an open decoding tunnel exists. Therefore, the weighting factors $\mathbf{g} = (g_1, \ldots, g_L)^T$ of the L different utilized index assignments need to be chosen such that the weights g_ℓ corresponding to high-rate index assignments are preferred. The optimization goal is in fact to minimize the number of bits $N_X = K_S\overline{w}^*$ after index assignment. The optimized weights $\mathbf{g}$ indicate the proportions of bits *after* index assignment and from these, the different $K_{S,\ell}$, i.e., the number of parameters encoded with the ℓ'th index assignment, can be determined. The number of resulting output bits after encoding a portion of $K_{S,\ell}$ parameters with an index assignment of rate r_ℓ^{IA} amounts to

$$K_{S,\ell}\frac{w}{r_\ell^{\text{IA}}} = g_\ell N_X \tag{4.24}$$

and it holds $\sum_\ell K_{S,\ell} \overset{!}{=} K_S$. Rewriting (4.24) to

$$N_X g_\ell r_\ell^{\text{IA}} = wK_{S,\ell} \tag{4.25}$$

and summing up over all L different index assignments leads to

$$N_X \sum_{\ell=1}^{L} r_\ell^{\text{IA}} g_\ell = w\sum_{\ell=1}^{L} K_{S,\ell} \tag{4.26}$$

which can be rewritten as

$$N_X\mathbf{r}^T\mathbf{g} = wK_S \Rightarrow N_X = \frac{wK_S}{\mathbf{r}^T\mathbf{g}}\,, \tag{4.27}$$

with $\mathbf{r} = (r_1^{\text{IA}}, \ldots, r_L^{\text{IA}})^T$. As K_S and w are constant, minimizing the number of total bits N_X corresponds to maximizing $\mathbf{r}^T\mathbf{g}$. This means that the task of optimizing the irregular index assignment such that decoding is still possible and resulting with a minimum number of output bits N_X can be formulated as a linear programming optimization problem with

$$\mathbf{g}_{\text{opt}} = \arg\max_{\mathbf{g}} \mathbf{r}^T\mathbf{g} = \arg\min_{\mathbf{g}}(-\mathbf{r}^T\mathbf{g}) \tag{4.28}$$

subject to

$$\mathbf{C}\cdot\mathbf{g} > \mathbf{d} + \mathbf{t}' \tag{4.29}$$

$$0 \leq g_\ell \leq 1 \quad \forall \ell \in \{1, \ldots, L\} \tag{4.30}$$

$$\sum_{\ell=1}^{L} g_\ell = 1\,, \tag{4.31}$$

with $\mathbf{C} \doteq (\mathbf{c}_1, \ldots, \mathbf{c}_L)$ composed of $\mathbf{c}_\ell \doteq (c_{\ell,1}, \ldots, c_{\ell,P})^T$, which consists of P sample points of the characteristic $\mathcal{C}_{\text{SDSD},\ell}$ of a specific index assignment Γ_ℓ^{R}. The vector $\mathbf{d}$ consists of P sample points of the inverse channel code EXIT characteristic $\mathcal{C}_{\text{CD}}^{-1}$, measured at the channel quality for which the system is optimized. The constraint (4.29) ensures that an open decoding tunnel is present. In (4.29), the vector $\mathbf{t}'$ denotes an offset vector which is chosen such that a larger open decoding tunnel is present, leading to better convergence properties at the receiver. In fact, the constraint $\mathbf{C}\cdot\mathbf{g} > \mathbf{d}$ would only guarantee an infinitely small decoding tunnel which could only be exploited with an infinite block size ($K_S \to \infty$). The constraints (4.30) and (4.31) ensure that all the weighting factors g_ℓ are in the range $0 \leq g_\ell \leq 1$ and

that $\sum_\ell g_\ell = 1$. The solution to this linear programming optimization problem can be easily found using numerical methods (see, e.g., [GMW91]).
Using the factors $\mathbf{g}_{\text{opt}}$ resulting from the optimization (4.28), the number of parameters $K_{S,\ell}$ which are encoded with the index assignment of rate r_ℓ^{IA} can be determined. An equation system with $L+1$ unknowns, consisting of the L different equations (4.24) ($\ell = 1\ldots, L$) and $\sum_\ell K_{S,\ell} \overset{!}{=} K_S$ can be formulated

$$\underbrace{\begin{pmatrix} g_1 & -\frac{w}{r_1^{\text{IA}}} & 0 & \cdots & 0 \\ g_2 & 0 & -\frac{w}{r_2^{\text{IA}}} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ g_L & 0 & \cdots & 0 & -\frac{w}{r_L^{\text{IA}}} \\ 0 & 1 & \cdots & \cdots & 1 \end{pmatrix}}_{\doteq \mathbf{F}} \cdot \begin{pmatrix} N_X \\ K_{S,1} \\ \vdots \\ K_{S,L} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_S \end{pmatrix} . \tag{4.32}$$

The matrix $\mathbf{F}$ can also be written in short notation with

$$\mathbf{F} = \begin{pmatrix} \mathbf{g} & -w \cdot \text{diag}(\mathbf{r})^{-1} \\ 0 & \mathbf{1}_L^T \end{pmatrix} . \tag{4.33}$$

The solution to (4.32) is given by

$$\begin{pmatrix} N_X \\ \mathbf{K} \end{pmatrix} = \mathbf{F}^{-1} \cdot \begin{pmatrix} \mathbf{0}_L \\ K_S \end{pmatrix} \tag{4.34}$$

with $\mathbf{K} = (K_{S,1}, \ldots, K_{S,L})^T$ and $\mathbf{0}_L$ denoting the length L all-zeros column vector. The fractions $K_{S,\ell}$ as well N_X can be determined by (numerically) solving (4.34). Note that due to its special structure, $\mathbf{F}$ has full rank and can be inverted. The value $N_X = K_S \overline{w}^*$ can also be determined using (4.27).

4.5.2 Simulation Example

We show the concept of near-lossless source coding using irregular index assignments by a simulation example. A block consisting of $K_S = 10000$ parameters resulting from a Gaussian distributed autoregressive source of first order (Gauss-Markov source) shall be compressed. The parameters show intra-frame correlation (i.e., the parameters in one frame are correlated) with correlation coefficient $\rho = 0.9$. The parameters are quantized using a $Q = 16$ level Lloyd-Max codebook $\mathbb{U}$. The index assignments BC_5^{16}, BC_6^{16}, ..., BC_{15}^{16} (with $\text{BC}_{w^*}^{Q}$) are generated using the design guidelines and the generator matrix given in [SVCS08].
For demonstrating the concept, we utilize a rate $r_{\text{orig}}^{\text{CC}} = \frac{1}{2}$ recursive systematic convolutional code of memory $J = 3$ with generator polynomials $G_{\text{orig}}^{\text{CC}}(D) = \left(1, \frac{1}{1+D+D^2+D^3}\right)$ which is punctured to an overall rate $r^{\text{CC}} = 2$ code by using the puncturing matrix

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} . \tag{4.35}$$

The EXIT characteristic $\mathcal{C}_{\text{CD}}$ of the rate $r^{\text{CC}} = 2$ convolutional code for perfect channel conditions (i.e., $E_s/N_0 \to \infty$) as well as the SDSD EXIT charts of the different index assignments of rates $4/15 \to 4/5$ are depicted in Fig. 4.13. The irregular characteristic $\mathcal{C}_{\text{SDSD}}^{\text{IIA},A}$ resulting from the optimization (4.28) is also depicted in Fig. 4.13. A quite narrow open decoding tunnel is present which means that quite a large number of iterations have to be performed at the receiver. The numerical result of the optimization is

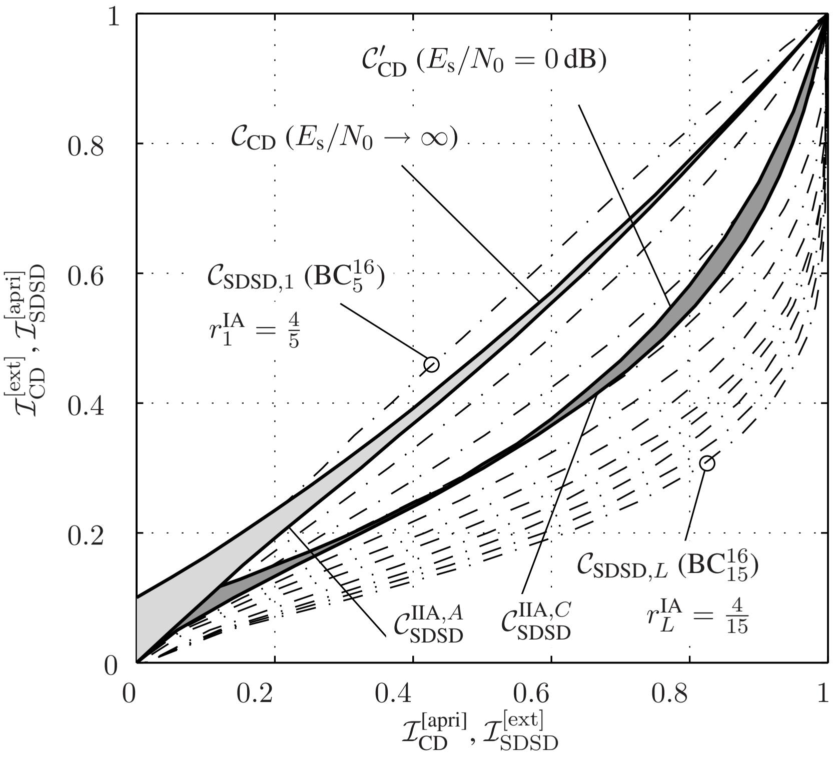


Figure 4.13: EXIT chart analysis of the irregular index assignments for source coding

given in Table 4.4 (Setup A): only the rate $4/5$ and $4/6$ index assignments are utilized. For $K_S = 10000$, this leads to a total number of $N_X = 55285$ bits after index assignment, i.e., a total number of 27644 bits after convolutional coding (≈ 2.76 bit per parameter). The upper limit of the reachable compression is given by the conditional entropy $H(\bar{U}_\kappa|\bar{U}_{\kappa-1})$ which in this setup amounts to $H(\bar{U}_\kappa|\bar{U}_{\kappa-1}) = 2.62$ bit per parameter leading to a minimal number of 26202 bits per block. The entropy can only be achieved if the decoding tunnel becomes infinitely narrow, however, this is not possible due to the channel decoder characteristic ($\mathcal{I}_{\text{CD}}^{[\text{ext}]}(\mathcal{I}_{\text{CD}}^{[\text{apri}]} = 0) \approx 0.1$). The utilization of a different channel code with a smaller value of $\mathcal{I}_{\text{CD}}^{[\text{ext}]}(\mathcal{I}_{\text{CD}}^{[\text{apri}]} = 0)$ might lead to better compression properties.

However, the decoding tunnel might be too narrow for the decoding because of the finite block size. By selecting an offset vector $\mathbf{t}' > \mathbf{0}_L$ a broader decoding tunnel can be found. The results of the optimization for this case are also given in Table 4.4 (Setup B).

The main advantage of the proposed system is the high flexibility. If the system shall be designed such that channel induced errors may occur, the parameter settings for the irregular index assignment can be easily adapted. As an example, we assume that the channel quality can drop down to $E_s/N_0 = 0\,\text{dB}$. In this case, the EXIT characteristic of the channel decoder changes and an intersection occurs. The EXIT characteristic of the rate $r^{\text{CC}} = 2$ convolutional code for $E_\text{s}/N_0 = 0\,\text{dB}$ is also depicted in Fig. 4.13 and denoted by $\mathcal{C}'_{\text{CD}}$. Using this characteristic, the linear programming optimization (4.28) can be carried out and a new optimized EXIT curve $\mathcal{C}_{\text{SDSD}}^{\text{IIA},C}$ results. Of course, as the channel quality becomes worse, additional artificial redundancy has to be introduced resulting in a larger block size N_X after index assignment. The result of the optimization is also given in Table 4.4 (Setup C). This irregular index assignment results in a total number of $N_X = 77670$ bits, i.e., 38837 bits after channel coding.

• Setup A ($E_\mathrm{s}/N_0 \to \infty$, $\mathbf{t}' = \mathbf{0}_L$)

Rate r_ℓ^{IA}	Γ_ℓ	g_ℓ	$K_{S,\ell}$	$K_{S,\ell} \cdot \frac{w}{r_\ell^{\mathrm{IA}}}$
4/5	BC_5^{16}	0.4265	$K_S^{(4/5)} = 4715$	23575
4/6	BC_6^{16}	0.5735	$K_S^{(4/6)} = 5285$	31710
		$\sum = 1$		$\sum = N_X = 55285$

• Setup B ($E_\mathrm{s}/N_0 \to \infty$, $\mathbf{t}' > \mathbf{0}_L$)

Rate r_ℓ^{IA}	Γ_ℓ	g_ℓ	$K_{S,\ell}$	$K_{S,\ell} \cdot \frac{w}{r_\ell^{\mathrm{IA}}}$
4/5	BC_5^{16}	0.2688	$K_S^{(4/5)} = 3061$	15305
4/6	BC_6^{16}	0.7312	$K_S^{(4/6)} = 6939$	41634
		$\sum = 1$		$\sum = N_X = 56939$

• Setup C ($E_\mathrm{s}/N_0 = 0\,\mathrm{dB}$, $\mathbf{t}' = \mathbf{0}_L$)

Rate r_ℓ^{IA}	Γ_ℓ	g_ℓ	$K_{S,\ell}$	$K_{S,\ell} \cdot \frac{w}{r_\ell^{\mathrm{IA}}}$
4/5	BC_5^{16}	0.1578	$K_S^{(4/5)} = 2451$	12255
4/6	BC_6^{16}	0.0650	$K_S^{(4/6)} = 842$	5052
4/9	BC_9^{16}	0.7772	$K_S^{(4/9)} = 6707$	60363
		$\sum = 1$		$\sum = N_X = 77670$

Table 4.4: Results of the irregular index assignment for near-lossless source coding

Figure 4.14 depicts the parameter SNR after reconstruction as a function of the performed iterations. For all three setups, a perfect channel quality ($E_\mathrm{s}/N_0 \to \infty$) has been studied in the example. Additionally, for Setup C, a channel quality of $E_\mathrm{s}/N_0 = 0\,\mathrm{dB}$ has been utilized. The more bits are utilized (and thus the larger the decoding tunnel is) the faster the convergence to the maximum parameter SNR of $\approx 20\,\mathrm{dB}$ (due to the quantization with codebook $\mathbb{U}$). For Setup A, it can be observed that even with 90 iterations we can only approach the maximum attainable parameter SNR. This is due to the very narrow decoding tunnel caused by $\mathbf{t}' = \mathbf{0}_L$ and the finite block size of $K_S = 10000$ parameters. In Setup B, with a slightly wider decoding tunnel ($\mathbf{t}' > \mathbf{0}_L$ with $\max_\ell t'_\ell = 0.01$) a faster convergence can be observed and the maximum parameter SNR is reached. Setup C with $E_\mathrm{s}/N_0 = 0\,\mathrm{dB}$ shows a slower convergence than Setup A because of the very narrow decoding tunnel visible in Fig. 4.13 (especially during the first iterations). For Setup C with $E_\mathrm{s}/N_0 \to \infty$ a large decoding tunnel is open. This leads to a very fast convergence (7 iterations) towards the maximum parameter SNR.

Notice that Setup A for $E_\mathrm{s}/N_0 \to \infty$ is unable to reach the maximum available parameter SNR even for a high number of iterations. This is due to the error floor which occurs due to the relatively small block size of 10000 parameters and the narrow decoding tunnel: The decoder is unable to reach the $(1, 1)$ point in the EXIT chart. This error floor is the reason that the system is only able to perform near-lossless source coding. This phenomenon has also been observed in [GZ92, Tho07]. For narrow decoding tunnels, the error floor is higher as for the setups with a large decoding tunnel. The reason for this is that the decoding gets stuck during the iterations if only a narrow tunnel is available. In the case

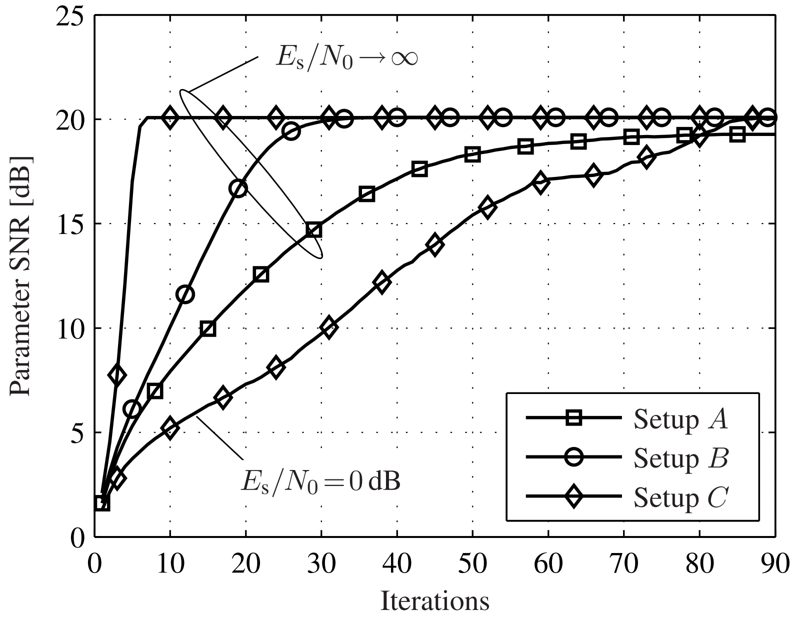


Figure 4.14: Reconstruction parameter SNR for the ISCD system employing irregular index assignments

of Setup C, it takes more than 90 iterations to reach the error floor at a channel quality of $E_s/N_0 = 0\,\text{dB}$ (see Fig. 4.13). However, if the channel quality gets better, a larger decoding tunnel can be observed and the error floor is reached after less than 10 iterations.

Chapter 5

Conclusion

In this document we have presented the final *FlexCode* channel coder. In Chapter 2 the channel coder implementation is detailed and both operating modes of the channel coder are presented. It is shown how the presented channel coding concept is used for iterative source-channel decoding and how multiple description is integrated within the *FlexCode* channel coder if channels where both bit errors and packet losses may occur, are employed. Several implementational details are given, such as the rate adaptation in both operating modes, the selection of good-working parameter individual block codes, and how the utilization of *a priori* knowledge may help to improve the subjective quality of the *FlexCode* codec in adverse channel conditions.
In Chapters 3 and 4 several findings and research advancements that have been made during the *FlexCode* project are presented. In Chapter 3 complexity reduction principles for iterative source-channel decoding (ISCD) are given. ISCD is the underlying concept of the *FlexCode* channel coder. Complexity reduction measures by either modifying the quantizer or by modifying the search procedure in the soft decision source decoder are given. It is also shown how both optimizations can be combined leading to a low-complexity source decoder by keeping the performance loss in terms of reconstruction SNR small. In Chapter 4 the concept of irregular index assignments and several applications of this powerful technique are given. These applications range from near Shannon-limit error control coding of arbitrary bit streams to error-resilient source compression and unequal error protection. All these applications are realized using the same underlying concept thus achieving and maintaining a high amount of flexibility, leading to the *FlexCode* "one system fits all" philosophy.

Bibliography

[ACS08] M. Adrat, T. Clevorn, and L. Schmalen. "Iterative Source-Channel Decoding & Turbo DeCodulation". R. Martin, U. Heute, and C. Antweiler, editors, *Advances in Digital Speech Transmission*, chapter 13, pages 365–398. John Wiley & Sons, Ltd., January 2008.

[Adr03] M. Adrat. *Iterative Source-Channel Decoding for Digital Mobile Communications*. PhD thesis, Aachener Beiträge zu Digitalen Nachrichtensystemen (ed. P. Vary), RWTH Aachen, 2003.

[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.

[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.

[BGK$^+$08] S. Bruhn, V. Grancharov, W. B. Kleijn, J. Klejsa, M. Li, J. Plasberg, H. Pobloth, S. Ragot, and A. Vasilache. "The FlexCode Speech and Audio Coding Approach". *ITG Conference Speech Communications*, Aachen, Germany, October 2008.

[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.

[Bre09] T. Breddermann. “Optimization of Structured Parameter Individual Block Codes for Iterative Source-Channel Decoding”. Personal communication, March 2009.

[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.

[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.

[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.

[CSVA08] T. Clevorn, L. Schmalen, P. Vary, and M. Adrat. “On Redundant Index Assignments for Iterative Source-Channel Decoding”. *IEEE Comm. Lett.*, 12(7):514–516, July 2008.

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

[DDP98] S. Dolinar, D. Divsalar, and F. Pollara. “Code Performance as a Function of Block Size”. *TMO Progress Report*, 42:133, 1998.

[FA98] V. Franz and J. B. Anderson. “Concatenated Decoding with a Reduced-Search BCJR Algorithm”. *IEEE J. Select. Areas Commun.*, 16(2):186–195, February 1998.

[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. available online at `http://www.flexcode.eu`.

[Fle07b] FlexCode. “Deliverable D-3.1: Ordered List of Real World Service Scenarios”. Technical report, European Union, March 2007. available online at `http://www.flexcode.eu`.

[Fle08a] FlexCode. “Deliverable D-1.1: Baseline Source Coder”. Technical report, European Union, February 2008. available online at `http://www.flexcode.eu`.

[Fle08b] FlexCode. “Deliverable D-1.2: Final Source Coder”. Technical report, European Union, October 2008. available online at `http://www.flexcode.eu`.

[Fle08c] FlexCode. “Deliverable D-2.2: Baseline Channel Coder”. Technical report, European Union, February 2008. available online at `http://www.flexcode.eu`.

[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.

[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.

[GMW81] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[GMW91] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization*, volume 1. Addison Wesley, Redwood City, 1991.

[Goe01] N. Goertz. “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.

[GZ92] J. Garcia-Frias and Y. Zhao. “Compression of Binary Memoryless Sources Using Punctured Turbo Codes”. *IEEE Communications Letters*, 6(9):394–396, September 2992.

[Hag]

[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.

[IT07] ITU-T. “G.729.1: G.729 Based Embedded Variable Bit-Rate Coder: An 8-32 kbit/s Scalable Wideband Coder Bitstream Interoperable with G.729”. Technical report, ITU-T, 2007.

[JN84] N. S. Jayant and P. Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Audio*. Prentice-Hall, 1984.

[KCR98] I. Kozintsev, J. Chou, and K. Ramchandran. “The Role of Arithmetic Coding based Continuous Error Detection in Digital Transmission Systems”. *IEEE International Symposium on Information Theory (ISIT)*, Cambridge, MA, USA, August 1998.

[KGM06] J. Kliewer, N. Goertz, and A. Mertins. “Iterative Source-Channel Decoding with Markov Random Field Source Models”. *IEEE Trans. Signal Processing*, 54:3688–3701, October 2006.

[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.

[KK09] J. Klejsa and W. B. Kleijn. “Rate Distribution Between Model and Signal for Multiple Descriptions”. *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, April 2009.

[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.

[LC03] S. Lin and D. J. Costello. *Error Control Coding*. Pearson Higher Education, 2nd edition, 2003.

[LHHH05] I. Land, S. Huettinger, P. A. Hoeher, and J. Huber. “Bounds on Information Combining”. 51(2):612–619, February 2005.

[LK03] X. Liu and S. N. Koh. “Simplification of Soft-Bit Speech Decoding and Application to MELP Encoded Speech”. *Electronics Letters*, 39(3):332–333, February 2003.

[LW07] F.-M. Li and A.-Y. Wu. “On the New Stopping Criteria of Iterative Turbo Decoding by Using Decoding Threshold”. *IEEE Trans. Signal Processing*, 55(11):5506–5516, November 2007.

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

[MB02] P. Mitran and J. Bajcsy. “Turbo Source Coding: A Noise-Robust Approach to Data Compression”. *Proc. of Data Compression Conference (DCC)*, page 465, Snowbird, UT, USA, April 2002.

[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.

[PYH07] A. Q. Pham, L. L. Yang, and L. Hanzo. “Joint Optimization of Iterative Source and Channel Decoding Using Over-Complete Source-Mapping”. *Proceedings of IEEE Vehicular Technology Conference (VTC)*, Baltimore, MD, USA, September 2007.

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

[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, Finland, June 2007.

[Sha59] C. E. Shannon. “Probability of Error for Optimal Codes in a Gaussian Channel”. *The Bell Systems Technical Journal*, 38(3):611–656, 1959.

[SV09] L. Schmalen and P. Vary. “Near-Lossless Compression and Protection by Turbo Source-Channel (De-)Coding Using Irregular Index Assignments”. *Proc. of IEEE ICASSP*, Taipei, Taiwan, April 2009.

[SVA08] L. Schmalen, P. Vary, and M. Adrat. “Reduced-Search Source Decoders for Iterative Source-Channel Decoding”. *ITG Conference Speech Communications*, Aachen, Germany, October 2008.

[SVAC08] L. Schmalen, P. Vary, M. Adrat, and T. Clevorn. “Complexity-Reduced Iterative Source-Channel Decoding by Conditional Quantization”. *Proc. of 5th International Symposium on Turbo Codes & Related Topics*, Lausanne, CH, September 2008.

[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.

[tB00] S. ten Brink. “A Rate One-Half Code for Approaching the Shannon Limit by 0.1dB”. *IEEE Electronics Letters*, 36(15):1293–1294, July 2000.

[ten01] S. ten Brink. “Convergence Behaviour of Iteratively Decoded Parallel Concatenated Codes”. *IEEE Trans. Commun.*, 49(10):1727–1737, October 2001.

[TH02a] M. Tüchler and J. Hagenauer. “EXIT Charts of Irregular Codes”. *Proc. of Conf. on Information Sciences and Systems (CISS)*, Princeton, NJ, USA, March 2002.

[TH02b] M. Tüchler and J. Hagenauer. “EXIT Charts of Irregular Codes”. *Proceedings of Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, USA, March 2002.

[Tho07] 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.

[TSV08] R. Thobaben, L. Schmalen, and P. Vary. “Joint Source-Channel Coding with Inner Irregular Codes”. *Proc. of IEEE International Symposium on Information Theory (ISIT)*, Toronto, Canada, July 2008.

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

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

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

[WM04] K. K. Y. Wong and P. J. McLane. “Bi-Directional Soft-Output M-Algorithm for Iterative Decoding”. *Proc. of IEEE ICC*, June 2004.

[ZKK09] G. Zhang, J. Klejsa, and W. B. Kleijn. “Analysis of K-Channel Multiple Description Quantization”. *Proc. of Data Compression Conference (DCC)*, Snowbird, UT, USA, March 2009.