# Extensions and Applications of High Rate Staircase Codes

Pratana Kukieattikool
Matrikelnummer : 1228356

Institute of Telecommunications
Technische Universität Wien

Supervisor : Univ. Prof. Dipl.-Ing. Dr.-Ing. Norbert Görtz

Submitted in partial fulfilment of the requirements for the degree of

*Doktor der Technischen Wissenschaften*

April 2016

This thesis is dedicated to my father and mother.

# Acknowledgements

I would like to express my sincerest thanks to Professor Norbert Görtz for giving me a chance to conduct a research project under his supervision. Without his support, guidance and encouragement, this work could not have been accomplished. The knowledge and advice I learned from him will be helpful for my career as a researcher throughout the entirety of my life.

I would like to thank all people from Multimedia System group for their friendships and support through these last years of my doctoral study.

I am grateful for the support by "Technologiestipendien Suedostasien" in the frame of ASEA-Uninet, granted by the Austrian Agency for the International Cooperation in Education & Research (OeAD), and the National Electronics and Computer Technology Center Thailand (NECTEC) for granting me the scholarship to do this study.

Finally, I would like to give my appreciation to my family who supported and encouraged me throughout the years of doing this doctoral degree. My father who inspired me to study as much as I am able to. My uncle Supath Kookiattikoon, who has helped correct my English in many parts of my work. Dr.Wasu Chaopanon, who read my work and gave me suggestion for improving the thesis. Timothy Tercero, who proof read my final draft. My husband and son, who have always believed in me and have been patient in waiting for me to be together the day I finish my study. I thank all of you for your support.

# Abstract

Staircase codes are a class of high performance forward-error-correction codes for high-rate transmission and hard decision decoding. They are initially designed for high-rate fiber optic transmission, which intends to correct errors in binary symmetry channels. However, to apply these high performance high-rate codes on wireless channels, some other aspects require close attention, which are addressed in this work.

Staircase codes for wireless transmissions on burst-error channels, which can be modelled using Gilbert and Elliott's model, are investigated. The Staircase codes with Reed-Solomon codes, as component codes, are tested in random-error as well as burst-error channels with different burst lengths and bit error probabilities, and are compared with the baseline Staircase codes with binary Bose-Chaudhuri-Hochquenghem (BCH) component codes. Furthermore Staircase codes with interleaving are implemented and tested in random-error, as well as burst-error channels with different burst lengths and bit error probabilities. For both types of component codes, the software complexities and decoding latencies are compared to see which codes have major impact on the decoding time of the Staircase codes.

For time-variant wireless channels (both optical and RF), Staircase codes with adaptive rates are proposed and used in type-II hybrid ARQ frameworks, so that throughput is maximized by avoiding retransmissions of the whole Staircase blocks that initially – at high code rate – might not have been decoded successfully. These rate-adaptive Staircase codes employ at their core the standard BCH component codes, but they are concatenated with Reed-Solomon codes as extra

components to implement burst-error correction and rate-adaptivity. Bit error performance and throughput of the rate-adaptive Staircase codes are investigated by analysis and confirmed with simulations.

The possibility of using staircase codes in the framework of distributed source coding (DSC) is also investigated as a further potential application. The bit error Slepian-Wolf coding, which is the bit error from lossy source coding, and the rate curve of Slepian-Wolf coding using Staircase codes with BCH component codes to compress the data are obtained.

# Contents

# List of Figures

# Chapter 1

# Introduction and Overview of the Thesis

Wireless communication becomes more and more essential to our everyday life. At the same time, communication devices such as smart phones, tablets, or laptops are maturely developed and are easily accessible for the general public. The reachability and data accessibility anywhere and anytime is thus an important concern. Meanwhile the transferred data volume is increasing rapidly due to the demand of high resolution of the video image files for entertainment or communication. These aspects cause the massive growth of wireless rate requirements above 10 Gbit/s; therefore, high code rate and lower error floor are a compulsory criterion. At the moment wireless communication is usually based on Wi-Fi or 3G/4G telephone networks, which employ radio frequency (RF) waves to connect end users. Due to the increase of transferred data rate, optical wireless is a possible alternative, which will allow for much higher bit rate.

To protect the transmitted data against channel disturbances, an error correction code is inserted into the data in a process called channel encoding. The end user or device must attempt to reconstruct the data from the received and corrupted data with the decoder. The decoding process that uses side-information from the channel to indicate the reliability of the data bits is called soft-decision decoding, while the process without using the side-information from the channel to indicate the reliability is called hard-decision decoding. Even though soft-

decision decoding allows for $\approx 2$ dB coding gain over hard-decision decoding, the latter is preferable for high-rate links because of much lower complexity resulting in lower decoding latency [13].

Among the best off-the-shelf codes for high-rate and hard-decision decoding are the so-called "Staircase codes" [6]. Those codes belong to a class of product codes but they are un-terminated. They can be constructed from algebraic component codes such as Bose-Chaudhuri-Hochquenghem (BCH), Reed-Solomon (RS), or Hamming codes (e.g., [46]). Staircase codes are iteratively decoded in a "shifted-window" fashion: this iterative decoding scheme is more efficient than message passing decoding for low-density parity-check (LDPC) codes (e.g., [46]) in the sense that it requires much lower data flow in the decoding hardware [6]; moreover it has smaller complexity, because it processes hard decisions, in contrast to the high performance message passing decoding of LDPC codes, which processes with soft information. Staircase codes achieve very high performance due to iterative decoding and a very low error floor due to the concatenation of their component codes. The baseline Staircase codes [6] have an error floor at $4.0 \times 10^{-21}$ and 9.41 dB net coding gain at an output bit error probability of $10^{-15}$. Therefore the concept of Staircase codes is of interest for utilisation in the area of high-rate wireless communication and in distributed source coding for sources with high correlation, which are the main topics of this thesis.

The aim of this thesis is to study the application of Staircase codes for high-rate and hard-decision decoding in wireless transmissions where different aspects are regarded other than the original purpose of Staircase codes, which had fixed high rate and were designed for random-error channels. These new aspects are burst-error channels and time varying channels. In addition, the application of the Staircase codes in a distributed source coding scheme, is examined to observe how high-rate channel codes can be used for distributed source coding of highly correlated (discrete) sources.

We investigate Staircase codes for high-rate wireless links that are prone to bundles (bursts) of channel errors caused by interference or link blockages. For indoor optical wireless channels, [12] shows that channels with line of sight component including all reflections, or diffuse channels follow a modified Rayleigh

distribution. The generative model[1] for burst errors can be derived from the Rayleigh distribution as stated in [79]. Nevertheless, we simulate burst errors with a Gilbert-Elliot model [29], [24] due to its simplicity. As stated in [46], [62], [52], RS codes are encoded symbol-wise and decoded such that the codes are capable of correcting burst errors. To bring about this result, we deploy RS codes as component codes of the Staircase scheme. The performances of the new RS Staircase codes on channels with different input error probabilities and burst lengths are investigated. It is known that the block interleaving is one method to help the codewords to combat burst errors in the channel, thus the deployment of interleaving in Staircase codes scheme is examined. Then the software complexities in decoding of the component codes are evaluated and decoding latency of the component codes are measured, so that a suitable (well-performing) RS component code can be selected.

The high-performance, high-rate Staircase codes [6] were originally designed with fixed rates for fiber optic communication. However, for the transmission over wireless channel, where the channel varies over time, an adaptation of the code rate is essential. With an extension by additional RS codes with the application in a type-II hybrid ARQ scheme proposed in this work, adaptive code rates are enabled so that the Staircase codes can be successfully decoded in a wider range of input bit error probabilities, thereby increasing throughput performance. The bit error performance and throughput performance of the rate-adaptive Staircase codes are analysed and confirmed with simulations.

We also employ Staircase codes in distributed source coding to investigate the performance of those codes in a different application. This is motivated by the high performance of hard-decision decoding of the Staircase codes for high-rate transmission. This high-rate transmission corresponds to high correlation of the sources in distributed source coding scheme. In addition it is simple to get the distributed source coding set up from the Staircase channel codes.

After this introduction, Chapter 2 will describe the concepts of channel coding, which are needed to understand the coding concepts in this work.

Then Chapter 3 describes the encoding and decoding methods for Staircase

[1] "The generative models are parameterized mathematical models capable of generating a statistically similar error sequence as produced by the real channel." [79]

codes and their component codes, which are Bose-Chaudhuri-Hochquenghem (BCH), Reed-Solomon (RS), and low-density parity-check (LDPC) codes.

Chapter 4 begins with G.709[1] compatible Staircase codes, which will be our baseline codes throughout this thesis; then the causes of high error floors of some Staircase codes are discussed and simulated. The LDPC codes with bit flipping decoding are implemented into Staircase codes as component codes and the performance is evaluated.

In Chapter 5 Staircase codes for high-rate wireless transmissions on burst-error channels are investigated and the complexities and decoding latencies of the component codes are compared.

In Chapter 6 rate compatible Staircase codes for high-rate wireless transmission are proposed, the analysis and simulation of the performance and throughput of the proposed rate-adaptive Staircase codes are given, and a performance comparison between the proposed scheme and a conventional retransmission scheme is presented.

Chapter 7 deals with the Staircase codes in distributed source coding (DSC). The performance analysis of Staircase codes in the DSC is given. The simulations of Staircase codes in the DSC scheme are performed so as to obtain the bit error Slepian-Wolf coding and the Slepian-Wolf rate curves. The bit error Slepian-Wolf coding is the bit error from lossy source coding, while the Slepian-Wolf rate curve is the minimum rate achievable from source coding at each value of $H(X|Y)$.

Chapter 8 summarises the main results and make suggestions for future work.

[1] recommendation for Interfaces for optical transport networks by the International Telecommunication Union

# Chapter 2

# Basic Concepts of Channel Coding

This chapter is based on [46], [11], [9], [52] [20] and [34].

The main theoretical background of channel coding is *Shannon information theory* [67], which states that information can be transmitted and received without error if the *rate* $R$ [in bits per channel use] of transmission is smaller than the *channel capacity* $C$, where very long blocks of transmitted data are assumed. In channel coding the *information sequence* $u$ is transformed by the encoder and the *encoded sequence* $v$ (called as a *codeword*) is obtained. Then the codewords are transformed by the modulator, sent through the channel, and then the demodulator produces the *received sequence* $r$. At last the decoder transforms the received sequence $r$ into the estimated information sequence $\hat{u}$. In Figure 2.1 the modulator and the demodulator are included in the channel block so that we consider the input and the output at the encoder and the decoder only.


Figure 2.1: Block transmission system.

There are two general types of codes, which are *Block codes* and *Convolutional codes*. In block coding the encoder transforms a message tuple $u = (u_0, u_1 ... u_{k-1})$

of length $k$ into the codeword $v = (v_0, v_1, ...v_{n-1})$ of length $n$ which adds redundancy with $n > k$. The message symbols and codeword symbols are from the same symbol alphabet $\mathcal{D} = \{d^{(1)}, d^{(2)}, ..., d^{(q)}\}$, which has $q$ elements. Therefore there are $q^k$ different possible codewords that exist in the vector space, which contains $q^n$ different length-n words. The set is called $(n, k)$ block code and has a rate $R = k/n$. By a convolutional code, the $k$-bit blocks of the information sequence $u$ are encoded into $n$-symbol blocks, of which the rate is $R = k/n$. The encoder has memory of order $m$ which means each codeword depends on the past $m$ values for the feedforward encoder.

## 2.1 Channel Model

The modulator transforms the encoded sequence into a suitable signal to transmit through the channel. There is a simple form of noise disturbance in the channel which is called *additive white Gaussian noise* (AWGN). It is a linear addition of wideband noise with constant spectral density and Gaussian amplitude distribution. Fading, multipath propagation, frequency selectivity, interference, nonlinearity and dispersion are not included in this model. The AWGN is often used due to its simplicity, and it allows to compare different coding schemes in a fair way.

### 2.1.1 Discrete Memoryless Channel (DMC)

When the detector outputs each channel symbol independently of the previous channel symbols, the channel is memoryless and the model is called *discrete memoryless channel* (DMC). This model is characterized by the *transition probabilities* $P(j|i)$, where $0 \leq i \leq M - 1$ represents a M-ary modulator input symbol and $0 \leq j \leq Q - 1$ represents a Q-ary demodulator output symbol, and $P(j|i)$ is the probability of receiving $j$ given that $i$ was transmitted.

### 2.1.2 Binary Symmetric Channel (BSC)

In case of binary modulation, symmetric noise, and 2 level demodulator outputs, the channel is called *binary symmetric channel* (BSC). The transition probability

$p_c$, which defines the probability of error transmission, can be calculated from the modulation signal set, the probability distribution of the noise, and the output quantization threshold of the demodulator. The transition probability of BPSK modulation on an AWGN channel with coherent detection and binary output quantization is [52]

$$p_c = Q(\sqrt{\frac{2E_s}{N_0}}) \tag{2.1}$$

with $E_s = nE_b$, $E_s$ is the energy-per-symbol, $E_b$ is the energy-per-bit, $n$ is number of bit per symbol, $N_0/2$ is the noise power spectral density, $Q(\cdot)$ is the complementary Gaussian error function given as

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt = \frac{1}{2}\text{erfc}(\frac{x}{\sqrt{2}}), \quad x > 0. \tag{2.2}$$

The binary output of the demodulator is then passed to the decoder. As decoding is carried out with binary decision, this type is called *hard-decision decoding*. The channel capacity $C$ of the BSC is given by [20]

$$C = 1 - H_2(p_c) \tag{2.3}$$

with the entropy

$$H_2(p_c) = -p_c \log(p_c) - (1 - p_c) \log(1 - p_c). \tag{2.4}$$

The model is illustrated in Figure 2.2.

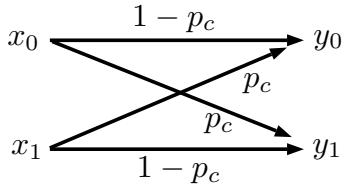



Figure 2.2: Binary symmetric channel (BSC).

### 2.1.3 Binary Erasure Channel (BEC)

For this channel model and in case of binary modulation, the demodulator output 3 levels, one of which is an erasure, the bit error probability $p_c = 0$ and the erasure probability is $p_e$: the model is called *binary erasure channel* (BEC). The channel capacity of the BEC is given by [20]

$$C = 1 - p_e. \tag{2.5}$$

The model is illustrated in Figure 2.3 .

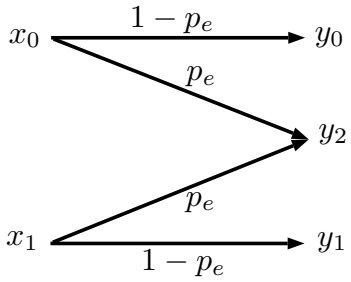



Figure 2.3: Binary erasure channel (BEC).

### 2.1.4 Additive White Gaussian Noise Channel (AWGNC)

When the input to the channel $X$ is a continuous random variable with zero mean and variance $\sigma_x^2$, and the channel has only AWGN, zero mean, and variance $\sigma_n^2$, the channel output $Y = X + N$ is also a Gaussian random variable with zero mean and variance $\sigma_x^2 + \sigma_n^2$. This channel is called the *additive white Gaussian noise channel (AWGNC)*. The channel capacity is given by [52]

$$C = \frac{1}{2}\log_2(1 + \frac{\sigma_x^2}{\sigma_n^2}). \tag{2.6}$$

The quantity $\sigma_x^2$ represents the average power in the transmitted signal $X$ and $\sigma_n^2$ represents the average power in the noise signal $N$ (both with zero mean).

In case there is only a 2 level input such as by BPSK modulation with amplitude $a$, the channel is called *binary additive white Gaussian channel (BAWGNC)*.

The channel capacity is given by [52]

$$C = -\int_{-\infty}^{\infty} \phi(y, E_b, \sigma^2) \log_2 \phi(y, E_b, \sigma^2) dy - \frac{1}{2} \log_2 2\pi e\sigma^2, \tag{2.7}$$

where

$$\phi(y, a, \sigma^2) = \frac{1}{\sqrt{8\pi\sigma^2}} \left[ e^{-(y-a)^2/2\sigma^2} + e^{-(y+a)^2/2\sigma^2} \right], \tag{2.8}$$

and $\sigma^2$ is the noise variance. Figure 2.4 illustrates this type of channel.

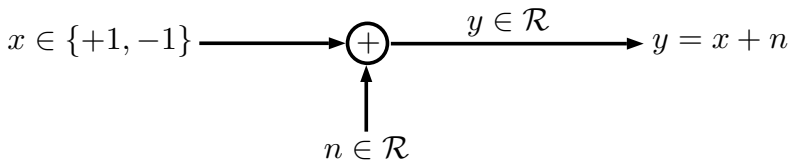



Figure 2.4: Binary additive white Gaussian channel (BAWGNC).

For transmission of a signal through a continuous-time channel with bandwidth $W$, transmitter power $P$ watts, and white Gaussian noise with two-sided power spectral density $N_0/2$. The channel capacity is given by [52]

$$C = W \log_2(1 + \frac{P}{N_0 W}) \quad \text{bits/second.} \tag{2.9}$$

Decoding for this type of channel is often carried out directly with the continuous channel outputs; such schemes referred to by *soft-decision decoding* are not used in this thesis due to much higher complexity than schemes based on hard-decisions.

## 2.2 Maximum Likelihood Decoding

For a coded system on an AWGN channel with quantised output $\mathbf{r}$, the decoder produces an estimated output $\hat{\mathbf{u}}$ based on $\mathbf{r}$ as well as the corresponding codeword

$\hat{\mathbf{v}}$. The error probability of the decoder is given by

$$P(E) = \sum_r P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r})P(\mathbf{r}), \tag{2.10}$$

where $P(\mathbf{r})$ is the probability of the received sequence $\mathbf{r}$ that is independent of decoding rules used. Therefore to minimize the error probability is equivalent to maximizing $P(\hat{\mathbf{v}} = \mathbf{v}|\mathbf{r})$

$$P(\mathbf{v}|\mathbf{r}) = \frac{P(\mathbf{r}|\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})}. \tag{2.11}$$

If all codewords are equally likely then to maximize (2.11) is equivalent to maximizing $P(\mathbf{r}|\mathbf{v})$. For a discrete memoryless channel (DMC)

$$P(\mathbf{r}|\mathbf{v}) = \prod_i P(r_i|v_i). \tag{2.12}$$

The decoder that uses (2.12) is called *maximum likelihood decoder* (MLD). Because $\log x$ is a monotonically increasing function of x, maximizing (2.12) is equivalent to maximize the log-likelihood function

$$\log P(\mathbf{r}|\mathbf{v}) = \sum_i \log P(r_i|v_i). \tag{2.13}$$

For a BSC with transition probability $p_c$, let $d(\mathbf{r}, \mathbf{v})$ be the *Hamming distance* between $\mathbf{r}$ and $\mathbf{v}$, which is the number of bit positions that are different. For a block code of length $n$ we get [34]

$$\begin{aligned} \log P(\mathbf{r}|\mathbf{v}) &= d(\mathbf{r}, \mathbf{v}) \log p_c + [n - d(\mathbf{r}, \mathbf{v})] \log(1 - p_c) \\ &= d(\mathbf{r}, \mathbf{v}) \log \frac{p_c}{1 - p_c} + n \log(1 - p_c). \end{aligned} \tag{2.14}$$

Since $\log \frac{p_c}{1-p_c} < 0$ for $p_c < \frac{1}{2}$ and $n \log(1 - p_c)$ is a constant for all $v$, the MLD rule for the BSC chooses $\hat{\mathbf{v}}$ as the codeword $\mathbf{v}$ that minimizes the distance $d(\mathbf{r}, \mathbf{v})$ between $\mathbf{r}$ and $\mathbf{v}$, hence this rule is called *minimum distance decoder*.

## 2.3 Performance Measurement

The performance of a coded communication system is measured by the number of decoding errors it produces. The probability of word (or block) error is called *word-error rate* ($WER$) or *block-error probability* ($BLER$), on the other hand the probability of bit error is called *bit-error probability* ($BER$).

Another figure of merit is called *coding gain* ($CG$), which is the reduction of the $E_b/N_0$ [1] required to achieve a specific error probability in the data for a coded system compared to an uncoded system. The $CG$ can be obtained by measuring the horizontal distance between $BER_{coded}$ and $BER_{uncoded}$ curves in the output $BER$ over received power graphs at a certain output $BER$ value. A formula for the $CG$ can be given as [76]

$$CG = 20 \log_{10} \left[ \frac{\text{erfc}^{-1}(2BER_{coded})}{\text{erfc}^{-1}(2BER_{uncoded})} \right] \quad \text{(dB)} \tag{2.15}$$

where $\text{erfc}^{-1}$ is the inverse function of the error function and AWGN channel is assumed, if not otherwise stated. When the code rate $R$ is also determined in a $CG$ in binary symmetric channel, we have the *Net Coding Gain* ($NCG$) given by [76]

$$NCG = CG + 10 \log_{10} R \quad \text{(dB)}. \tag{2.16}$$

It is desirable to minimize the $E_b/N_0$ required to achieve a specific error rate, which is equivalent to maximizing the coding gain in the system using the same modulation signal set. A theoretical bound on the minimum $E_b/N_0$ required for a coded system with code rate $R$ to achieve error-free or an arbitrary small error probability is the Shannon limit. It guarantees the existence of a coded system achieving error free communication but without limits on complexity or delay of the encoding and decoding algorithm.

[1] Energy per bit to noise power spectral density ratio, also known as signal-to-noise ratio (SNR) per bit

## 2.4 Minimum Distance and Minimum Weight

The *Hamming weight* of a code vector $\mathbf{c}$ of length $n$ is the number of elements in $\mathbf{c}$ that are different from 0

$$w_H(\mathbf{c}) = \sum_{j=0}^{n-1} wt(c_j) \quad \text{with} \quad wt(c_j) = \begin{cases} 0, & c_j = 0 \\ 1, & c_j \neq 0. \end{cases} \tag{2.17}$$

The *minimum Hamming weight* of a block code $\mathcal{C}$ is the minimum of Hamming weights of all nonzero codeword $\mathbf{c} \in \mathcal{C}$

$$w_{H,min}(\mathcal{C}) = \min_{\mathbf{c}\in\mathcal{C},\mathbf{c}\neq 0} w_H(\mathbf{c}). \tag{2.18}$$

The *Hamming distance* of 2 code vectors $\mathbf{a}$, $\mathbf{c}$ is the number of different elements in $\mathbf{a}$ and $\mathbf{c}$

$$d_H(\mathbf{a},\mathbf{c}) = \sum_{j=0}^{n-1} wt(a_j + c_j) \quad \text{with} \quad wt(a_j + c_j) = \begin{cases} 0, & c_j = a_j \\ 1, & c_j \neq a_j. \end{cases} \tag{2.19}$$

The *minimum Hamming distance* of a block code $\mathcal{C}$ is the minimum of the Hamming distance of any two different codewords $\mathbf{a}, \mathbf{c} \in \mathcal{C}$

$$d_{H,min}(\mathcal{C}) = \min_{\mathbf{a},\mathbf{c}\in\mathcal{C},\mathbf{a}\neq\mathbf{c}} d_H(\mathbf{a},\mathbf{c}). \tag{2.20}$$

## 2.5 Linear Block Codes

Here we define a linear block code $\mathcal{C}(n,k,d)$ as a code with block length $n$, with dimension $k$ and minimum distance $d$. The code is linear, when every linear combination of two codewords $\mathbf{c}, \mathbf{c}' \in \mathcal{C}$ is again in $\mathcal{C}$. Such a code can correct up to $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors and detect up to $d-1$ errors.

The codeword $\mathbf{c} = (c_0, c_1, ...c_{n-1})$ can be generated from the information vector $\mathbf{i} = (i_0, i_{1,}, ....i_{k-1})$ via

$$\mathbf{c} = \mathbf{i} \cdot \mathbf{G}, \tag{2.21}$$

where $\mathbf{G}$ is the *generator matrix* with dimension $(k \times n)$, of which each row is a base vector of linear vector space. All of the codewords are constructed by linear combination of the rows of the generator matrix. Besides, the linear block code $\mathbf{c}$ can be checked by the *parity check matrix* $\mathbf{H}$: for each codeword $\mathbf{c}$

$$\mathbf{H} \cdot \mathbf{c}^T = \mathbf{0} \tag{2.22}$$

has to apply. The dimension of $\mathbf{H}$ is $((n-k) \times n)$[1].

For linear codes, the minimum distance is equal to minimum weight. That is

$$d_{H,min}(\mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C}, \mathbf{c} \neq 0} w_H(\mathbf{c}). \tag{2.23}$$

This property is important because it is simpler to consider the minimum weight of a code rather than its minimum distance. To correct the received codeword $\mathbf{r} = \mathbf{c} + \mathbf{f}$, where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{f}$ is the error, the decoder will choose the codeword with the smallest distance to the received codeword. Therefore, to preserve the best output bit error probability after decoding, the following equation should be used:

$$w_H(\mathbf{f}) \leq \left\lfloor \frac{d_{H,min} - 1}{2} \right\rfloor. \tag{2.24}$$

*Systematic encoders* construct the codeword with the $k$ information symbols unchanged as components, the redundancy are the $(n-k)$ parity symbols appended to the data bits . Then the parity check matrix has the following structure

$$\mathbf{H} = (\ \mathbf{A} \mid \mathbf{I}\ ); \mathbf{I} \text{ is the Identity matrix of dimension } (n-k) \times (n-k). \tag{2.25}$$

The generator matrix has the structure (for the binary case)

$$\mathbf{G} = (\ \mathbf{I}' \mid \mathbf{A}^{\mathbf{T}}\ ); \mathbf{I}' \text{ is Identity matrix dimension } (k \times k). \tag{2.26}$$

The *syndrome* $\mathbf{s}$ is obtained from the check equation. For the received word

[1]The minimum dimension $\mathbf{H}$ could contain redundancy rows (see LDPC codes)

$\mathbf{r} = \mathbf{c} + \mathbf{f}$, $\mathbf{c} \in \mathcal{C}$, $\mathbf{f}$ is the error, the check equation

$$\mathbf{s}^T = \mathbf{H} \cdot \mathbf{r}^T = \mathbf{H} \cdot (\mathbf{c}^T + \mathbf{f}^T) = \mathbf{H} \cdot \mathbf{f}^T \tag{2.27}$$

depends only on the error vector, because $\mathbf{H} \cdot \mathbf{c}^T = \mathbf{0}$ for $\mathbf{c} \in \mathcal{C}$. The function of the decoder is to search for the most likely error vector $\mathbf{f}$ that causes the observed syndrome $\mathbf{s}$.

## 2.6 Decoding Principle

The result of decoding can be *correct decoding*, *incorrect decoding* or *decoding failure*. Correct decoding is when the decoded word $\hat{\mathbf{c}}$ is the same as the codeword $\mathbf{c}$ sent. Incorrect decoding/decoding error occurs when the decoded word $\hat{\mathbf{c}}$ is not the same as the sent codeword $\mathbf{c}$. This means the decoder has tried to correct the received word but it chose the wrong error vector $\mathbf{f}$ which did not occur in the channel. Decoding failure occurs when the decoder has no solution whether it was correct or incorrect decoding. There are different kinds of decoding principles as described below.

### 2.6.1 Error Detection

The decoder checks if the received word $\mathbf{r}$ is a codeword or not. That means that it only checks whether $\mathbf{r} \in \mathcal{C}$. For $\mathbf{r} \notin \mathcal{C}$ an error is detected.

### 2.6.2 Maximum Likelihood Decoding

The received word $\mathbf{r}$ is decoded as a codeword $\hat{\mathbf{v}}$, which was sent with maximum probability, so the decoding rule is

$$\hat{\mathbf{v}}(\mathbf{r}) = \arg\max_{\mathbf{v} \in \mathcal{C}} P(\mathbf{r}|\mathbf{v}). \tag{2.28}$$

In case there is more than one codeword with the same maximum probability, then the decision to favour one of them will be taken randomly. In case of the BSC the codeword $\hat{\mathbf{v}}$ is the one with the minimum Hamming distance to $\mathbf{r}$. The

result is either correct or incorrect but the decoding process always comes up with the result.

### 2.6.3 Symbol Maximum a Posteriori Decoding (MAP)

For decoding, each data symbol $u_i$ is considered separately and the decision of each symbol is based on the entire received word $\mathbf{r}$

$$\hat{u_i}(\mathbf{r}) = \arg\max_{u \in \mathcal{D}} \; P_{U_i|\mathbf{R}}(u|\mathbf{r}), \tag{2.29}$$

where $\mathcal{D}$ is symbol alphabet e.g. binary bits in binary coding.

### 2.6.4 Bounded Minimum Distance Decoding (BMD)

The received word $\mathbf{r}$ is decoded only when it is situated within the correction sphere of radius equal to or smaller than $\left\lfloor \frac{d-1}{2} \right\rfloor$ with $d$ the minimum distance of the code. The result of decoding can be all 3 cases: correct decoding, incorrect decoding and decoding failure.

## 2.7 Asymptotic Bounds

The following bounds [9] specify the possible rate $R$ of a codeword $\mathbf{c}$ with minimum distance $d$ that does not go against the channel coding theory for $n \to \infty$. Here we consider $(n, k, d)$ block code with message length $k$, codeword length $n$ and minimum distance $d$ for only BSC with error probability $p$.

### 2.7.1 Singleton Bound (Upper bound)

For any two codewords that are different in at least $d$ positions, if we remove the first $d-1$ symbols of each codeword, we get the shortened codewords of length $n-(d-1)$ which are different in at least one position [34]. Thus, the $2^k$ shortened codewords are all different in the set of $2^{n-d+1}$ codewords, which is possible only

if $2^k \leq 2^{n-d+1}$. Taking the ld(.) on both sides, we get

$$n - k \geq d - 1, \tag{2.30}$$

which can be interpreted as lower bound on the number of parity symbols $n - k$.

When we divide both sides of (2.30) by $n$, we get an upper bound on code rate $R$ given as

$$R \leq 1 - \frac{d}{n} + \frac{1}{n}. \tag{2.31}$$

For asymptotic regime where $n \to \infty$, thus $\frac{d}{n} \to 2p$ [34], the Singleton bound can finally be given as

$$R \leq 1 - 2p. \tag{2.32}$$

### 2.7.2 Hamming Bound (Upper bound)

In each decoding sphere of the BMD decoder, which corresponds to Hamming sphere $\mathcal{S}_t(\mathbf{c})$ of radius $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ around each codeword $\mathbf{c}$, the number of words can be given as (for BSC) [34]

$$|\mathcal{S}_t(\mathbf{c})| = \sum_{i=0}^{t} \binom{n}{i}. \tag{2.33}$$

The total number of words contained in any decoding sphere is equal to $q^k|\mathcal{S}_t(\mathbf{c})|$, which must not be greater than the total number of words in the vector space, which is equal to $2^n$. Therefore we get

$$\sum_{i=0}^{t} \binom{n}{i} \leq 2^{n-k} \tag{2.34}$$

Taking the ld(.) on both sides, and then divide both side by $n$, the upper bound on code rate $R$ is given as

$$R \leq 1 - \frac{1}{n}\text{ld}\left(\sum_{i=0}^{t}\left(\begin{array}{c} n \\ i \end{array}\right)\right). \tag{2.35}$$

For asymptotic regime where $n \to \infty$ and $t/n \leq 1/2$, (2.35) can be written using asymptotic expression [34] as

$$R \leq 1 - H(\frac{t}{n}), \tag{2.36}$$

where

$$H(p) = -p \cdot \text{ld}\ p - (1-p) \cdot \text{ld}\ (1-p). \tag{2.37}$$

Then for asymptotic regime $t/n \equiv \delta/2$ and $\delta \to 2p$ [34], finally we get

$$R \leq 1 - H(p). \tag{2.38}$$

This bound corresponds to the channel capacity $C$ and is always tighter than the Singleton bound.

### 2.7.3 McEliece Rodemich Rumsey Welch Bound (Upper bound)

This bound is the most famous upper bound and can be given for $n \to \infty$ as

$$R \leq H\left(\frac{1}{2} - \sqrt{2p \cdot (1-2p)}\right). \tag{2.39}$$

### 2.7.4 Varshamov Bound (Lower bound)

This bound is a lower bound on code rate $R$ and can be given for $n \to \infty$ as

$$R \geq 1 - 2p\ \text{ld}\frac{1}{2p} - (1-2p)\ \text{ld}\frac{1}{(1-2p)} = 1 - H(2p). \tag{2.40}$$

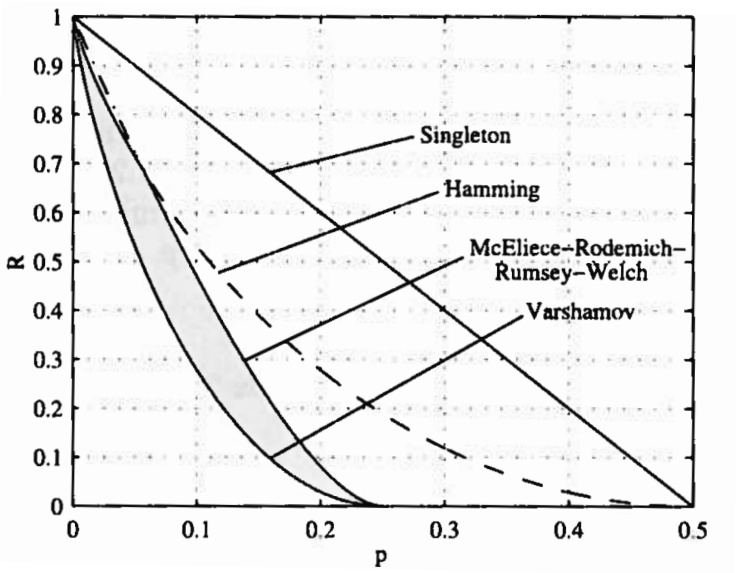



Figure 2.5: Asymtotic bounds. [9]

These asymptotic bounds are illustrated in Figure 2.5. Good codes exist above the Varshamove bound and below all upper bounds (in grey area).

## 2.8 Product Codes

Product codes construct long efficient codes from short component codes. These component codes can be $\mathcal{C}_1(n_1, k_1, d_1)$ and $\mathcal{C}_2(n_2, k_2, d_2)$. The product code $(n_1 n_2, k_1 k_2)$ is a code array of $n_1$ columns and $n_2$ rows, in which every row is a codeword in $\mathcal{C}_1$ whereas every column is a codeword in $\mathcal{C}_2$. First, the information of $k_1 k_2$ symbols are arranged in the upper left corner as in Figure 2.6, then each row is encoded into a codeword in $\mathcal{C}$ which results in an array of $k_2$ rows and $n_1$ columns. Secondly each column is encoded into a codeword in $\mathcal{C}$ which results in a code array of $n_2$ rows and $n_1$ columns. At last the codeword is transmitted row-by-row or column-by-column. The minimum weight of this code is $d_1 d_2$.

The error pattern that can be corrected depends on the algorithm of decoding. One method is a two-step decoding, firstly columns then rows. The improved performance can be obtained by the iterative decoding, which decodes columns,

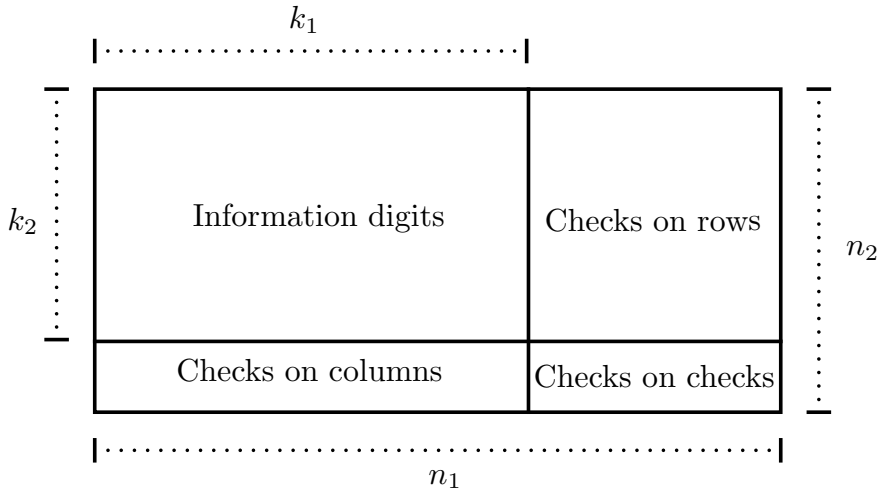



Figure 2.6: Code array for the product code.

rows, columns, rows and so on. "The product code has error correction capability equal to $\lfloor \frac{d_1 d_2 - 1}{2} \rfloor$ errors, but two step decoding is not enough to achieve this error correction capability" [46], thus it is better to decode iteratively to achieve the high error correction capability of the product codes.

The *incomplete product code* is the product code without checks on checks. It is a $(k_1 n_2 + k_2 n_1 - k_1 k_2, k_1 k_2)$ linear block code with minimum distance $d_1 + d_2 - 1$. It has higher rate but smaller minimum distance than the *complete product code*.

### 2.8.1 Decoding Thresholds and Error Floor of Product Codes with Iterated Decoding

The decoding threshold of product codes can be stated as follows [41]: an error pattern affecting a product code can be interpreted as a bipartite "error graph", where each node represents a component code decoder, and each edge represents an erroneous symbol. Due to the structure of a product code, each erroneous symbol is contained in two sub-graphs that belong to a "vertical" and a "horizontal" component code. The error graph is a random graph, as the errors are assumed to occur randomly and independently of each other. The process of decoding corresponds to iteratively removing of those nodes and their edges, for

which the number of connected edges to a node is not bigger than the number $t$ of symbols that can be corrected by the component code. It is assumed that the sequence of decoding the component codes is random; decoding of the product code finally fails when an error graph remains that (then) has a least $t+1$ edges. Such a sub-graph is called $(t+1)$-core in Graph theory [57].

As a result, [57] indicates that a random graph $G(n, m)$ with $n$ vertices and $m$ edges has high probability of a $k$-core to appear, when $m$ reaches $c_k\, n/2$, where $c_k = \min_{\lambda>0} \lambda/\pi_k(\lambda)$ and $\pi_k(\lambda) = P\{\text{Poisson}(\lambda) \geq k-1\}$ is the probability that random Poisson-distributed variable (with parameter $\lambda$) has a value of at least $k-1$. The probability mass function of a Poisson($\lambda$) random variable $X$ is given as [85]

$$P_X(x) = \begin{cases} \lambda^x e^{-\lambda}/x! & \text{for } x = 0, 1, 2, ..., \\ 0 & \text{otherwise.} \end{cases} \tag{2.41}$$

The solution for $c_k$ can be approximated by $c_k \approx k + \sqrt{k \log k}$ [41]. Thus, it was concluded in [41] that as $n$ increases while $t > 1$ is fixed, the probability that a random graph contains a $(t+1)$-core vanishes if and only if the total number of errors is smaller than

$$W = n\frac{c_{t+1}}{2}, \tag{2.42}$$

where $t$ is error correction capability of the component codes. This is the decoding threshold of iterated decoding of product codes: if more errors than this threshold $W$ occur in an error pattern of a product code, iterative decoding is very likely to fail. When the iterative decoding succeeds, the tangent of the output bit-error probability curve is steep: this is the waterfall region of the performance curve.

An example of error graph of a product code is given in Figure 2.7. There are 10 rows of component codes, 10 columns of component codes and 22 error positions in the product code array, which corresponds to $n = 20$ vertices, and $m = 22$ edges in the error graph. After column component codes decoded by removing edges connecting to not more than the error correction capability $t = 3$ in the column vertices subgraph, the error graph results in Figure 2.7(c). Then

after row component codes decoded by removing edges connecting to not bigger than the error correction capability $t = 3$ in the row vertices subgraph, the error graph results in Figure 2.7(d), which shows that all errors are able to be corrected as there are no more edges in the error graph.

After decreasing the input bit error probability $p_E$ until the points that the output bit error probabilities $p_O$ fall down slowly, this region in the curve is recognized as *error floor*. First consider the probability $p_{t1}$ that there are $t_1 + 1$ errors in $n$ rows:

$$p_{t1} = \begin{pmatrix} n \\ t_1 + 1 \end{pmatrix} p_E^{(t_1+1)}(1 - p_E)^{(n-t_1-1)}, \tag{2.43}$$

and the probability $p_{t2}$ that there are $t_2 + 1$ errors in $n$ columns given as

$$p_{t2} = \begin{pmatrix} n \\ t_2 + 1 \end{pmatrix} p_E^{(t_2+1)}(1 - p_E)^{(n-t_2-1)}. \tag{2.44}$$

For $n \gg t_1, t_2$ we can approximate $p_{t1}$ and $p_{t2}$ by

$$p_{t1} \approx \begin{pmatrix} n \\ t_1 + 1 \end{pmatrix} p_E^{(t_1+1)}, \tag{2.45}$$

$$p_{t2} \approx \begin{pmatrix} n \\ t_2 + 1 \end{pmatrix} p_E^{(t_2+1)}. \tag{2.46}$$

The product $p_{t1} * p_{t2}$ is the probability that there are $t_1+1$ rows and $t_2+1$ columns in error, each of which are $(t_1 + 1)(t_2 + 1)$ errors in $n^2$ symbols of the array. Such a pattern is the $(t+1)$-core, which cannot be decoded with iterative decoding and causes error floor. As a consequence, the output bit error probability $p_O$ where the error floor occurs can be approximated as [41]

$$p_O \approx \begin{pmatrix} n \\ t_1 + 1 \end{pmatrix} \begin{pmatrix} n \\ t_2 + 1 \end{pmatrix} p_E^{(t_1+1)(t_2+1)}(t_1 + 1)(t_2 + 1)/n^2, \tag{2.47}$$

where $n$ is the number of rows or columns of the product code, $t_1$ is the error

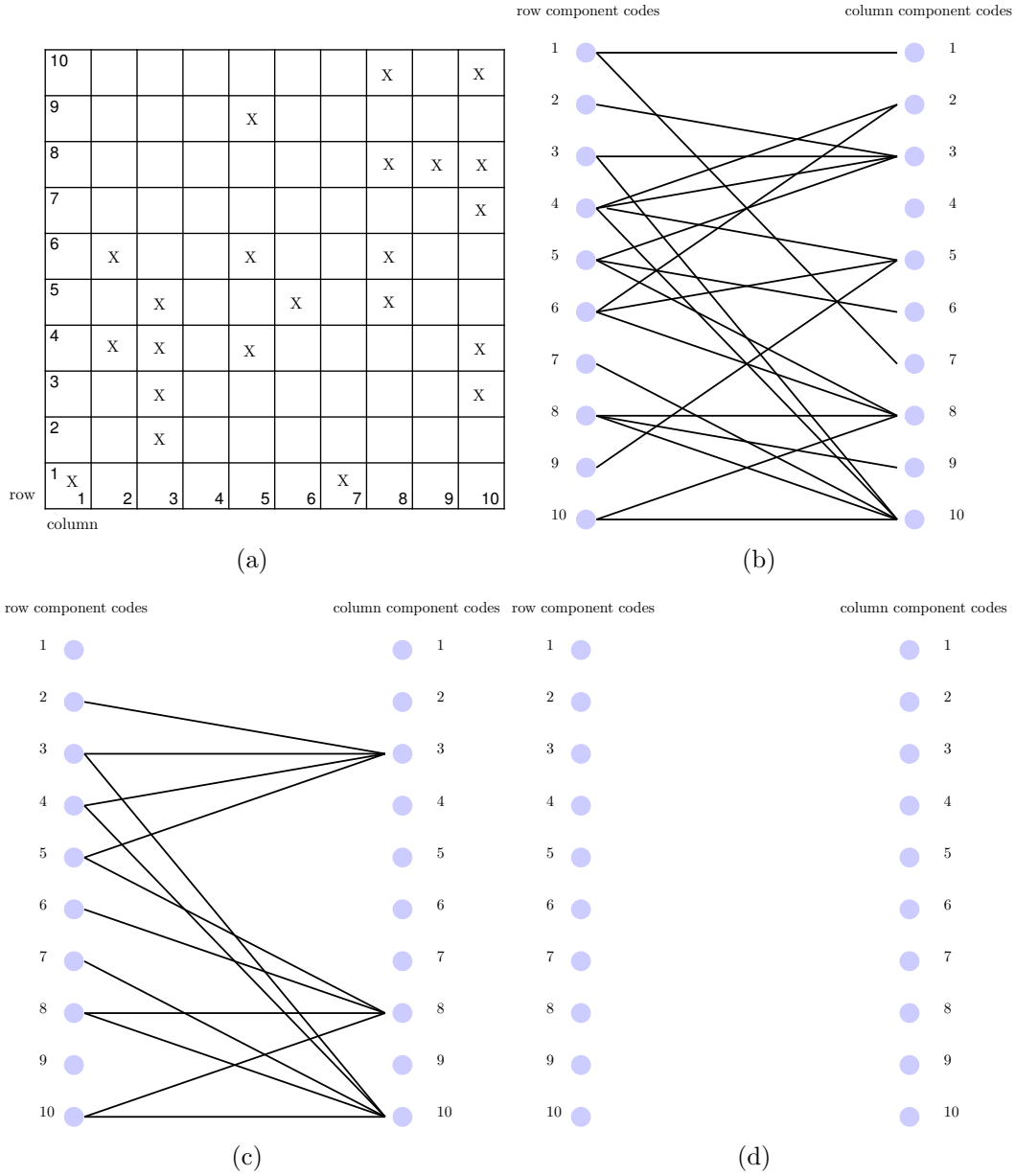



Figure 2.7: Example of an error graph of a product code with $G(n = 20, m = 22)$. All component codes have error correction capability $t = 3$ (a) a product code array with 10 rows and 10 columns, $X$s are positions in error (b) an error graph before decoding (c) an error graph after the decoding of column component codes (d) an error graph after the decoding of row component codes.

correction capability of the row component codes and $t_2$ is the error correction capability of the column component codes.

## 2.9 Finite Fields and Extension Fields

A *finite field* $(\mathrm{GF}(q), +, \cdot)$ [9] is a set of elements from symbol alphabet $\mathcal{D} = \{d^{(1)}, d^{(2)}, ..., d^{(q)}\}$, which has $q$ elements or order of $q$. We can perform addition "+" and multiplication "$\cdot$" of two elements, $a, b \in \mathrm{GF}(q)$, without leaving the set: $a + b \in \mathrm{GF}(q)$ and $a \cdot b \in \mathrm{GF}(q)$. The addition operation is commutative thus $a+b = b+a$ with the additive identity 0. The multiplication is also commutative $a \cdot b = b \cdot a$ with the multiplicative identity 1. The distributive law over addition for three elements, $a, b, c \in \mathrm{GF}(q)$, must hold: $a \cdot (b + c) = a \cdot b + a \cdot c$. The inverse element $-a \in \mathrm{GF}(q)$ of $a \in \mathrm{GF}(q)$ for addition operation exists for which $a + (-a) = 0$. The multiplicative inverse $a^{-1} \in \mathrm{GF}(q)$ of $a \in \mathrm{GF}(q)\backslash\{0\}$ exists such that $a \cdot a^{-1} = 1$.

If a field is constructed from a prime $p$, it is called a *prime field* and can be written as $\mathrm{GF}(p)$, for $p = 2$ we obtain binary field $\mathrm{GF}(2)$. For any prime $p$ a finite field of $p$ elements exists. For any positive integer $m$, it is possible to extend the prime field $\mathrm{GF}(p)$ to $p^m$ elements, which is called an *extension field* of $\mathrm{GF}(p)$ denoted by $\mathrm{GF}(p^m)$. Furthermore, it has been proven that the order of any finite field is a power of a prime. The $\mathrm{GF}(q^m)$ can be represented as $m$- tuple [34]

$$\mathrm{GF}(q^m) = \{(a_0, a_1, ..., a_{m-1}) \mid a_i \in \mathrm{GF}(q)\}, \tag{2.48}$$

or as a polynomial

$$\mathrm{GF}(q^m) = \left\{ \sum_{i=0}^{m-1} a_i X^i \mid a_i \in \mathrm{GF}(q) \right\}, \tag{2.49}$$

where $X$ is a dummy parameter for the polynomial. Through the polynomial representation, the addition of two elements of $\mathrm{GF}(q^m)$ can be done by adding the coefficients, while the multiplication operation can be done by multiplication of polynomials modulo $f(X)$, where $f(X)$ denotes any fixed prime polynomial

of degree $m$ over GF($q$). The *prime polynomial* is an irreducible polynomial[1] in GF($q$) that has the highest degree coefficient equal to one. Linear codes $\mathcal{C}(\mathrm{GF}(q); n, k, d_{min})$ can be interpreted as subspace of GF($q^n$) with dimension $k$, as a consequence they can be written in the polynomial notation.

A *primitive element* $\alpha$ is an element in GF($q$), of which each power $j = 0, .., q-2$ of this element ($\alpha^j$) represents all non zero elements in this GF($q$).

A *primitive polynomial* is a prime polynomial over GF($q$) of degree $m$, which has zeros $z_l \in \mathrm{GF}(q^m)$ as primitive elements of GF($q^m$). These zeros are all distinct, then $z_l = z^{(q^l)}$ for $l = 0, 1, ..., m-1$. Thus in GF($q^m$) we can factor $f(X)$ as

$$f(X) = \prod_{i=0}^{m-1} (X - z^{(q^l)}). \tag{2.50}$$

A *minimal polynomial* $\Phi(X)$ of any element $\beta$ in GF($q^m$) is

$$\Phi(X) = \prod_{i=0}^{e-1} (X + \beta^{2^i}), \tag{2.51}$$

where $e$ is the smallest integer that $\beta^{2^e} = \beta$. The minimal polynomial is irreducible in GF($q$) and will be used in the BCH code construction.

## 2.10 Cyclic Codes

A cyclic code is a linear block code $\mathcal{C}$ over GF($q$) of block length $n$ if, whenever $\mathbf{v} = (v_0, v_1, ...v_{n-1})$ is in $\mathcal{C}$, then $\mathbf{v}' = (v_{n-1}, v_0, ...v_{n-2})$ is also in $\mathcal{C}$. The codeword $\mathbf{v}'$ is obtained by cyclically shifting the components of the codeword $\mathbf{v}$ one place to the right. The code can also be represented in a polynomial form of degree of at most $n-1$ according to [34]

$$v(X) = \left\{ \sum_{i=0}^{n-1} v_i X^i \mid \ v_i \in \mathrm{GF}(q) \right\}. \tag{2.52}$$

[1] a polynomial that cannot be divided by any polynomials of degree $\geq 1$

The polynomial coefficients $v_i$, $i = 0, ..., n-1$ are the components of the codeword $\mathbf{v}$. The cyclic property can be written in polynomial form as

$$v(X) \in \mathcal{C} \quad \Rightarrow \quad v^{(l)}(X) = R_{X^n-1}[X^l v(X)] \in \mathcal{C} \ \text{ for } l = 0, 1, .., n-1, \quad (2.53)$$

where $R_{X^n-1}[\cdot]$ is the ring of polynomials modulo $X^n - 1$. Cyclic codes have a structure that is strongly related to finite fields; hence they can be encoded and decoded algorithmically and computationally more efficiently than the tabular decoding techniques used for general linear codes.

A cyclic code $v(X)$ consists of all multiples of the *generator polynomial* $g(X)$ of degree $n - k$ by data polynomial $u(X)$ of degree at most $k - 1$

$$v(X) = u(X)g(X). \quad (2.54)$$

The generator polynomial $g(X)$ must divide $X^n - 1$ as follows

$$X^n - 1 = g(X)h(X); \quad (2.55)$$

the polynomial $h(X)$ of degree $k$ is called *check polynomial*.

To obtain *systematic encoding*, the data is inserted into the high-order coefficients of the codeword, and then the check symbols are computed. The codeword is

$$v(X) = X^{n-k}u(X) + p(X), \quad (2.56)$$

and the *parity polynomial* $p(X)$ is a polynomial of degree at most $n-k-1$ given by

$$p(X) = -R_{g(X)}[X^{n-k}u(X)]. \quad (2.57)$$

The systematic encoder is constructed with the shift register [46] in Figure 2.8, of which the generator polynomial is $g(X) = 1 + g_1X + g_2X^2 + ... + g_{n-k-1}X^{n-k-1} + X^{n-k}$. Firstly, the "gate" is turned on and the $k$ information digits $u_0, u_1, ...u_{k-1}$ are shifted into the circuit, and at the same time to the channel, as the switch to "codeword" is connecting to "message" (as shown in Figure 2.8). As soon

as the complete message has entered the circuit, the $n-k$ digits in the register correspond to the remainder, which are the parity check digits. Then the "gate" is turned off to break the feedback connection, and at the same time the parity check digits are shifted into the channel by connecting the switch between "parity-check digits" and "codeword".

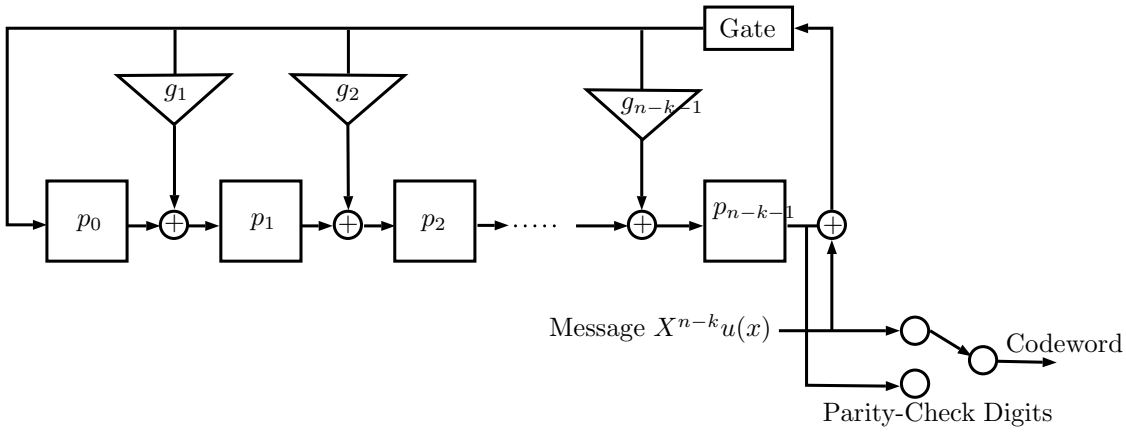



Figure 2.8: Systematic encoding for $(n, k)$ cyclic code. [46]

Let $v(X)$ denote the transmitted codeword, the *received polynomial* $r(X)$ is equal to $v(X) + e(X)$, where $e(X)$ is *error polynomial*. The coefficients of the error polynomial are non-zero when channel errors occurred. The *syndrome polynomial* $s(X)$ used for decoding are the remainder under division by $g(X)$ given as

$$s(X) = R_{g(X)}[r(X)] = R_{g(X)}[v(X) + e(X)] = R_{g(X)}[e(X)]. \tag{2.58}$$

The decoding task is finding the unique $e(X)$ with the least number of non zero coefficients satisfying (2.58). The syndrome can also be generated by the shift register as shown in Figure 2.9. The received polynomial $r(X)$ is shifted in to the register from the left end with all stages set initially to zero. When the whole $r(X)$ has been shifted into the register the syndrome $s(X)$ is fetched from the contents in the register. This circuit can be used as a component of a decoder for $(n, k)$ cyclic code known as *Meggitt decoder*, which contains a syndrome register,

an error-pattern detector, and a buffer register. Furthermore it can be used as a component of error-trapping decoder, for which the "gate" in Figure 2.9 controls the shifting of the syndrome register until contents of the register are the same as the error digits (details see [46]).

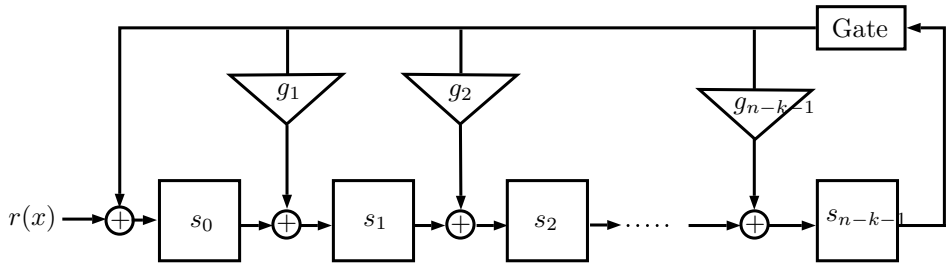



Figure 2.9: An $(n-k)$-stages syndrome circuit with input from the left end. [46]

## 2.11 Rate Adaptation Methods for Block Codes

Assume that a $(N, K)_q$ block code in GF($2^q$) with generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}$ is given as a "mother code" with rate $R = K/N$. Furthermore it is assumed that the code is systematically encoded, i.e., the $K$ data symbols are mapped to $N$ code symbols by appending $N - K$ parity symbols.

### 2.11.1 Extending and Puncturing

In extending and puncturing, the number of data symbol $K$ is kept constant as in the mother code. However by extending, the number of parity bits $N - K$ is increased letting the rate decrease. By puncturing, the number of parity bits $N - K$ is decreased letting the rate increase [34].

In practice the mother code is from a lower-rate and the rate adaptive encoder sends the punctured codewords through the channel corresponding to the rate requirement. The puncturing pattern is known to the decoder and hence the received punctured word is treated as a codeword of the mother code with erased symbols.

### 2.11.2 Lengthening and Shortening

For lengthening and shortening, the number of $N - K$ parity symbols remains constant as for the mother code. However by lengthening, the block length $N$ is increased by increasing the number of data symbols $K$. By shortening, the block length $N$ is decreased by decreasing number of data symbols $K$. The code rate of lengthening is increased whereas the rate of shortening is decreased [34].

In practice a shortened codeword has a number of data symbols equal to $K - D$, where $D$ is the number of shortening symbols. The vector of length $K$, with $D$ zeros filled at the pre-determined positions, is encoded resulting in a codeword of length $N$, but only $N - D$ symbols are sent through the channel. The locations of the $D$ filled zeros are known to the decoder, and therefore the decoder fills the received word with $D$ zeros at those positions, which results in the received word of length $N$. The received word is decoded with the mother code decoder, resulting in the codeword of length $K$. At last the $D$ zero positions are dropped and the decoded symbol of length $K - D$ is the result.

### 2.11.3 Augmenting and Expurgating

For augmenting and expurgating the block length $N$ is kept constant. The number of data symbols $K$ is increased and the number of parity symbols $N - K$ is decreased by the same value through the use of augmenting, thus resulting in an increasing rate. The number of data symbols $K$ is decreased and the number of parity symbols $N - K$ is increased by the same value through the use of expurgating, thus resulting in a decreasing rate [34]. In practice augmenting and expurgating are more complex than shortening or puncturing, since both the number of data symbols and the number of parity symbols change at the same time.

# Chapter 3

# Staircase Codes and their Component Codes

## 3.1 Staircase Codes Principle

Staircase codes are a class of forward-error-correction codes (FEC) first proposed in [6]. The code is constructed to be a continuous product code, which can be efficiently hard-decoded, and is therefore suitable for high-speed optical communications. It is claimed in [6] that "this code, using syndrome based decoding, is significantly more efficient than message-passing-decoding of LDPC code, in terms of data flow, and has better performance than any other code recommended in ITU-T recommendations G.975 and G.975.1 [77]. The error floor analysis of the code results in a very low error floor at $4.0 \times 10^{-21}$, contrary to the LDPC codes that always have a higher error floor."[6] In this section we will give insight into the encoding and decoding of Staircase codes.

### 3.1.1 Encoding

Staircase codes are characterized by the relationship of the successive matrices of symbols denoted $B_0, B_1, B_2...$ of $m$ by $m$ matrices $B_i,\ i \in \mathcal{Z}^+$. The elements in $B_i$ are restricted here to GF(2), but the concept can also be applied to non-binary cases. Conventional FEC in systematic form is used as the *component code* referred to as $\mathcal{C}$ with a block length of $2m$ symbols, $r$ of which are parity

symbols. The component codes can be Hamming, BCH, RS codes etc.

The encoding proceeds recursively on the $B_i$. The block $B_0$ is initialized as an all zeros matrix $m \times m$ and also known to the decoder. For each $i$ we have $m \times (m-r)$ information bits arranged in the left-most column of $B_i$ which results in $m \times (m-r)$ arrays denoted by $B_{i,L}$. Thereafter the array $A = [B_{i-1}^T B_{i,L}]$ is formed, resulting in an array of size $m \times 2m - r$, where $B_{i-1}^T$ is the transpose of $B_{i-1}$. Then the entries of $B_{i,R}$ are computed such that each row of the array $[B_{i-1}^T B_{i,L} B_{i,R}]$ is a valid codeword of $\mathcal{C}$ as can be seen in Figure 3.1.

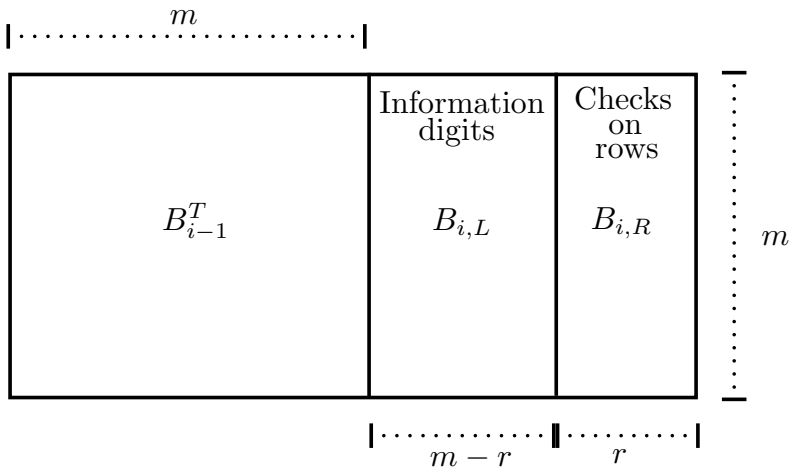



Figure 3.1: Staircase code array for encoding.

Staircase codes have a structure related to product codes, but they are unterminated, and are therefore decoded with varying latencies. The visualization in Figure 3.2 shows how the name staircase is originated.

The code has the *overall rate*

$$R = \frac{m \times (m-r)}{m \times m} = 1 - \frac{r}{m}. \tag{3.1}$$

According to the product codes properties, "If the component codes $\mathcal{C}$ have a minimum distance $d_{min}$, then the Staircase code has a minimum distance of at least $d_{min}^2$" [6].

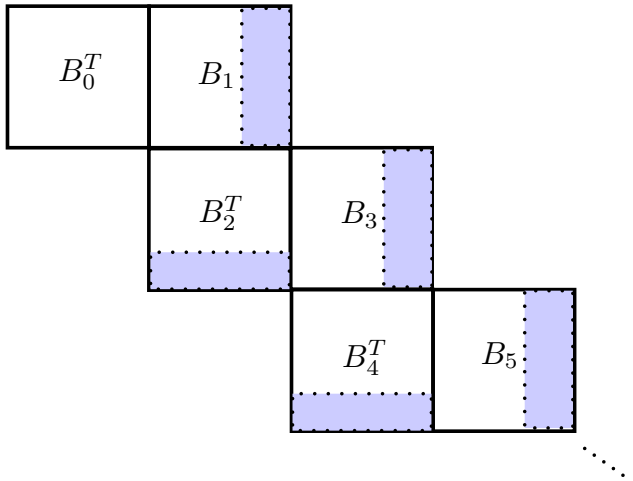



Figure 3.2: "Staircase" visualization of Staircase codes: the parity-check symbols are located in the shaded boxes.

### 3.1.2 Decoding

Staircase codes are unterminated, and thus can be decoded using a sliding window with arbitrary length $L$. The consecutively received blocks $B_i, B_{i+1}, B_{i+2}, ...B_{i+L-1}$ are decoded as follows:

Firstly the codewords that terminate in block $B_{i+L-1}$ are decoded. Since one codeword involves two blocks, both blocks of $B_{i+L-1}$ and $B_{i+L-2}$ are decoded and updated. Then the codewords that terminate in block $B_{i+L-2}$ are decoded and updated. This process continues until the codewords that terminate in block $B_i$ are each decoded and updated. Because decoding the block that is terminated at $B_j$ also impacts the block $B_{j+1}$, it is efficient to process back to $B_{i+L-1}$ and repeat until the sufficient number of iteration is reached. Lastly the decoder outputs the information of the block $B_i$. After finishing one round, the window slides to the right, i.e., the "decided" block $B_i$ is removed and the new block $B_{i+L}$ is accepted, and the whole process repeats.

## 3.2 Component Codes

### 3.2.1 BCH Codes

The Bose, Chaudhuri, and Hocquenghem(BCH) codes form a class of cyclic codes for random-error correction. Binary BCH codes were discovered by Hochquenghem [35] and independently by Bose and Chaudhuri [10]. The most efficient decoding algorithm for these codes is Berlekamp's iterative algorithm [7],[8] and Chien's search algorithm [18]. What follows below is based on [46].

#### 3.2.1.1 Encoding

Here we will consider only binary primitive BCH codes. For any positive integer $m$ ($m > 3$) and $t$ ($t < 2^{m-1}$), there exists a binary BCH code with the following parameters: primitive block length $n = 2^m - 1$, number of parity check digits $n - k < mt$ and minimum distance $d_{min} \geq 2t + 1$. This code can correct any combination of $t$ or fewer errors in a block of $n = 2^m - 1$ digits. Such a code is called a *t-error-correcting BCH code*. Let $\alpha$ be a primitive element[1] in GF($2^m$). The generator polynomial $g(x)$ of a $t$-error correcting BCH code of length $2^m - 1$ is the *lowest-degree polynomial* over GF(2) that has $\alpha, \alpha^2, \alpha^3, ..., \alpha^{2t}$ as its roots

$$g(\alpha^i) = 0 \text{ for } \ 1 \leq i \leq 2t, \tag{3.2}$$

then $g(x)$ is the *least common multiple (LCM)* of $\Phi_1(X)$,$\Phi_2(X)$,....,$\Phi_{2t}(X)$ given as

$$g(X) = LCM\{\Phi_1(X), \Phi_2(X), ...., \Phi_{2t}(X)\}, \tag{3.3}$$

where $\Phi_i(X)$ is the minimal polynomial[2] of $\alpha^i$. Because every even power of $\alpha$ in the sequence has the same minimal polynomial as some preceding odd power of

[1] $\alpha \in$ GF($q$) is primitive if every nonzero element $x \in$ GF($q$) $\setminus \{0\}$ can be represented as a power of $\alpha$; $x = \alpha^j$ for some $j \in \{0, 1, ..., q-2\}$

[2] A *minimal polynomial* $\Phi(X)$ of any element $\beta$ in GF($q^m$) is $\Phi(X) = \prod_{i=0}^{e-1}(X + \beta^{2^i})$, where $e$ is the smallest integer that $\beta^{2^e} = \beta$.

$\alpha$ in the sequence $g(x)$ can be reduced to

$$g(X) = LCM\{\Phi_1(X), \Phi_3(X), ...., \Phi_{2t-1}(X)\}. \tag{3.4}$$

This binary BCH codes with length $n = 2^m - 1$ is called *primitive* BCH codes.

The general definition of binary BCH codes with a designed distance $d_0$ is generated by the binary polynomial of degree that has consecutive powers of $\beta$ as roots, which are $\beta^{l_0}, \beta^{l_0+1}, ..., \beta^{l_0+d_0-2}$ , where $\beta$ is an element of GF($2^m$) and $l_0$ is a nonnegative integer. The generator polynomial is

$$g(X) = LCM\{\Psi_1(X), \Psi_1(X), ...., \Psi_{d_0-2}(X)\}, \tag{3.5}$$

and the length of the code is

$$n = LCM\{n_0, n_1, ..., n_{d_0-2}\}, \tag{3.6}$$

where $\Psi_i(X)$ is the minimal polynomial, and $n_i$ is the order of $\beta^{l_0+i}$ for $0 \leq i < d_0 - 1$. This general BCH code has minimum distance of at least $d_0$ and has a number of parity check digits not more than $m(d_0 - 1)$; hence it can correct not more than $\lfloor (d_0 - 1)/2 \rfloor$ errors.

#### 3.2.1.2 Decoding

The general approach of BCH error-correcting procedure has four steps as follows:

1. Compute the syndrome $\mathbf{S} = (S_1, S_2, ..., S_{2t})$ from the received polynomial $r(x)$. For a t-error correcting BCH code, the syndrome is a $2t$ tuple

$$\mathbf{S} = (S_1, S_2, ..., S_{2t}) = \mathbf{r} \cdot \mathbf{H^T}, \tag{3.7}$$

where $\mathbf{r}$ is the length $n$ receive vector $\mathbf{r} = r_0 + r_1 X + r_2 X^2 + ...r_{n-1} X^{n-1}$

and the check matrix is

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & ... & \alpha^{n-1} \\ 1 & (\alpha^2) & (\alpha^2)^2 & (\alpha^2)^3 & ... & (\alpha^2)^{n-1} \\ 1 & (\alpha^3) & (\alpha^3)^2 & (\alpha^3)^3 & ... & (\alpha^3)^{n-1} \\ \vdots & & & & & \vdots \\ 1 & (\alpha^{2t}) & (\alpha^{2t})^2 & (\alpha^{2t})^3 & ... & (\alpha^{2t})^{n-1} \end{bmatrix}. \tag{3.8}$$

Then $i$ th component of the syndrome is

$$S_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2\alpha^{2i} + ... + r_{n-1}\alpha^{(n-1)i} \quad \text{for} \quad 1 \leq i \leq 2t. \tag{3.9}$$

The syndrome depends only on the error pattern $e(X)$, thus $S_i = e(\alpha^i)$. The *error pattern* has $\nu$ errors at locations $X^{j_1}, X^{j_2}, ..., X^{j_\nu}$ that is

$$e(X) = X^{j_1} + X^{j_2} + ... + X^{j_\nu}, \tag{3.10}$$

where $0 \leq j_1 < j_2 < ... < j_\nu < n$. Set each $\alpha^i$ ; $1 \leq i \leq 2t$ in (3.10) we obtain $2t$ equations

$$\begin{aligned} S_1 &= \alpha^{j_1} + \alpha^{j_2} + ... + \alpha^{j_\nu} \\ S_2 &= (\alpha^{j_1})^2 + (\alpha^{j_2})^2 + ... + (\alpha^{j_\nu})^2 \\ S_3 &= (\alpha^{j_1})^3 + (\alpha^{j_2})^3 + ... + (\alpha^{j_\nu})^3 \\ &\vdots \\ S_{2t} &= (\alpha^{j_1})^{2t} + (\alpha^{j_2})^{2t} + ... + (\alpha^{j_\nu})^{2t}, \end{aligned} \tag{3.11}$$

where $\alpha^{j_1}, \alpha^{j_2}, ..., \alpha^{j_\nu}$ are unknown. If we have found $\alpha^{j_1}, \alpha^{j_2}, ..., \alpha^{j_\nu}$, the powers $j_1, j_2, ..., j_\nu$ indicate the error locations in $e(X)$. If the number of errors in $e(X)$ is $t$ or fewer, the solution is the error pattern with smallest number of errors and it is the desired solution. Let

$$\beta_l = \alpha^{j_l} \tag{3.12}$$

be the *error-location number*. Then we define the *error-location polynomial*

$$\sigma(X) = (1+\beta_1 X)(1+\beta_2 X)...(1+\beta_v X), \tag{3.13}$$

of which the coefficients must be searched. Note that the roots of $\sigma(X)$ are $\beta_1^{-1}, \beta_2^{-1}, ..., \beta_v^{-1}$ which are the inverses of the error-location numbers. Then by expansion of the error-location polynomial, we get

$$\sigma(X) = \sigma_0 + \sigma_1 X + \sigma_2 X^2 + ... + \sigma_v X^v, \tag{3.14}$$

of which the coefficients are

$$\begin{aligned} \sigma_0 &= 1 \\ \sigma_1 &= \beta_1 + \beta_2 + ... + \beta_\nu \\ \sigma_2 &= \beta_1\beta_2 + \beta_2\beta_3 + ... + \beta_{\nu-1}\beta_\nu \\ &\vdots \\ \sigma_\nu &= \beta_1\beta_2...\beta_\nu. \end{aligned} \tag{3.15}$$

These $\sigma_i$'s are called the *elementary symmetric functions* of error-location numbers and are non-linear functions to the syndromes.

The *linear* relationship between the syndromes and the coefficients of the error-location polynomial can be described by Newton identities, given as [52]

$$\begin{aligned} S_k + \sigma_1 S_{k-1} + ... + \sigma_{k-1} S_1 + k\sigma_k &= 0 \quad ; 1 \le k \le \nu \\ S_k + \sigma_1 S_{k-1} + ... + \sigma_{\nu-1} S_{k-\nu+1} + \sigma_\nu S_{k-\nu} &= 0 \quad ; k > \nu. \end{aligned} \tag{3.16}$$

For $k > \nu$ the notation can be written as

$$S_j = -\sum_{i=1}^{\nu} \sigma_i S_{j-i}, \tag{3.17}$$

which can be generated from the linear feedback shift register such that the relation between the syndromes and the coefficients of the error-location

polynomial can be found.

2. Determine the error-location polynomial $\sigma(x)$ from $\mathbf{S}$ using the Berlekamp-Massey Algorithm or Key equation solver or Peterson's algorithm for BCH codes.

   The *Peterson-Gorenstein-Zierler algorithm* is given here, as from this algorithm the formulas for the coefficient of $\sigma(X)$ are explicitly given for a small number of errors, and thus [6] suggested using this method to find the error-location polynomial for BCH component codes of high-rate Staircase codes.

   Firstly (3.17) can be written in matrix form as

$$\begin{bmatrix} S_1 & S_2 & \dots & S_\nu \\ S_2 & S_3 & \dots & S_{\nu+1} \\ S_3 & S_4 & \dots & S_{\nu+2} \\ \vdots & & & \\ S_\nu & S_{\nu+1} & \dots & S_{2\nu-1} \end{bmatrix} \begin{bmatrix} \sigma_\nu \\ \sigma_{\nu-1} \\ \sigma_{\nu-2} \\ \vdots \\ \sigma_1 \end{bmatrix} = - \begin{bmatrix} S_{\nu+1} \\ S_{\nu+2} \\ \vdots \\ S_{2\nu} \end{bmatrix}. \tag{3.18}$$

   The $\nu \times \nu$ matrix is called $M_\nu$, where $\nu$ has to be determined using the Peterson-Gorenstein-Zierler decoder as follows [52]:

   (a) Set $\nu = t$.

   (b) Form $M_\nu$ and calculate the determinant $\det(M_\nu)$ to check whether it is invertible. If it is not invertible, set $\nu \leftarrow \nu - 1$ and rerun this step.

   (c) If $M_\nu$ is invertible, solve for the coefficient $\sigma_1, \sigma_2, ..., \sigma_\nu$.

   The explicit formulars are as follows [52]:
   **1-error correction** $\sigma_1 = S_1$.
   **2-error correction** $\sigma_1 = S_1$, $\sigma_2 = (S_3 + S_1^3)/(S_1)$.
   **3-error correction** $\sigma_1 = S_1$, $\sigma_2 = \frac{(S_1^2 S_3 + S_5)}{(S_1^3 + S_3)}$, $\sigma_3 = (S_1^3 + S_3) + S_1\sigma_2$.
   **4-error correction** $\sigma_1 = S_1$
   $\sigma_2 = \frac{S_1(S_7+S_1^7)+S_3(S_1^5+S_5)}{S_3(S_1^3+S_3)+S_1(S_1^5+S_5)}$, $\sigma_3 = S_1^3 + S_3 + S_1\sigma_2$, $\sigma_4 = \frac{(S_5+S_1^2S_3)+(S_1^3+S_3)\sigma_2}{S_1}$.
   **5-error correction** $\sigma_1 = S_1$,
   $\sigma_2 = \frac{(S_1^3+S_3)[(S_1^9+S_9)+S_1^4(S_5+S_1^2S_3)+S_3^2(S_1^3+S_3)]+(S_1^5+S_5)(S_7+S_1^7)+S_1(S_3^2+S_1S_5)}{(S_1^3+S_3)[(S_7+S_1^7)+S_1S_3(S_1^3+S_3)]+(S_5+S_1^2S_3)(S_1^5+S_5)}$,

$$\sigma_3 = (S_1^3 + S_3) + S_1\sigma_2,$$
$$\sigma_4 = \frac{(S_1^9+S_9)+S_3^2(S_1^3+S_3)+S_1^4(S_5+S_1^2S_3)+\sigma_2[(S_7+S_1^7)+S_1S_3(S_1^3+S_3)]}{S_1^5+S_5},$$
$$\sigma_5 = S_5 + S_1^2S_3 + S_1S_4 + \sigma_2(S_1^3 + S_3).$$

It can be seen that with an increasing number of error correction capability $t$, the formulae are more complex and thus using Peterson's algorithm is not efficient anymore. Thus, the less complex method known as *Berlekamp-Massey algorithm* is employed. The main idea of this algorithm is to try to find the shortest linear feedback shift register (LFSR) that can generate $\{S_1, S_2, ..., S_{2t}\}$ where the coefficient of this LFSR is the error-location polynomial $\sigma(X)$ of smallest degree. For details of the algorithm see [7], [8], [46], [52].

3. The error-location numbers $\beta_1, \beta_2, ..., \beta_v$ are determined by finding the roots of $\sigma(X)$ using Chien's procedure [18]. This search is exhaustive over all the elements in the field GF($q^m$) with elements $\{1, \alpha, \alpha^2, ..., \alpha^{q^m-2}\}$ by evaluating these values by the error-location polynomial. The efficient implementation in hardware is shown in Figure 3.3 where firstly the register is loaded with the coefficients of the error-location polynomial, and the initial output is

$$A = \sum_{j=1}^{\nu} \sigma_j = \sigma(x) - 1|_{x=1}. \tag{3.19}$$

If $A = 1$ the error location has been found because $\sigma(x) = 0$. Thereafter each register is multiplied by $\alpha^j$ where $j = 1, 2, ..., \nu$ results

$$A = \sum_{j=1}^{\nu} \sigma_j\alpha^j = \sigma(x) - 1|_{x=\alpha}. \tag{3.20}$$

The next step is done until all the non zero elements in the field are evaluated. If all of the roots that are found are different and stay in that field, they can be used to determine the error-location numbers.

4. Flip the bits at locations according to the error-location number in $r(x)$ such that the word is corrected.

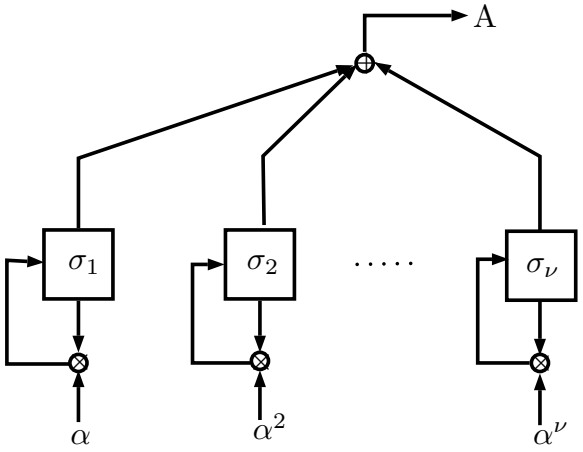



Figure 3.3: Chien search algorithm.

From the fact that polynomial $f^2(X) = f(X^2)$ over GF(2), the received vector $\mathbf{r}^2(X) = \mathbf{r}(X^2)$. When we substitute $\alpha^i$ for $X$ in the received polynomial, we have for the syndrome $S_{2i} = S_i^2$. Therefore the complexity of computing all the syndromes reduce to half, as well as the complexity of finding the error-location polynomial using the Berlekamp-Massey Algorithm.

### 3.2.2 RS Codes

Reed-Solomon (RS) codes are a nonbinary subclass of BCH codes, which were invented by Reed and Solomon [62] independently of the invention of BCH codes in the same year. The efficient decoding algorithm for these codes is also Berlekamp's iterative algorithm [7],[8] and Chien's search algorithm [18] similar to BCH codes. The Euclidean algorithm, which was invented by Sugiyama, Kasahara, Hirasawa, and Namekawa [71], is a simple concept used for decoding of both RS and BCH codes. The BCH and RS codes can additionally be encoded and decoded in frequency domain [9].

#### 3.2.2.1 Encoding

If we first consider nonbinary BCH codes with block length $n = q^m - 1$, and we set $m = 1$, it results in RS codes with block length $n = q - 1$: this is how RS

codes are defined as a subclass of q-ary BCH codes. The primitive element $\alpha$ is a symbol in the Galois field GF($q$) and is not over the extension field as in BCH codes. The parameters of a t-error-correcting RS code: number of parity-check symbols: $n-k=2t$ and minimum distance $d_{min}=2t+1$ cause the RS codes to be one of the *maximum distance separable (MDS)*[1] codes.

The generator polynomial $g(x)$ of a $t$-error correcting RS code of length $q-1$ is the *lowest-degree polynomial* over GF($q$) that has $\alpha, \alpha^2, \alpha^3, ..., \alpha^{2t}$ as its roots

$$g(\alpha^i) = 0 \text{ for } \ 1 \leq i \leq 2t, \tag{3.21}$$

and, because $\alpha^i$ is an element of GF($q$), the minimal polynomials are $X-\alpha^i$, thus, reads

$$\begin{aligned} g(X) &= (X-\alpha)(X-\alpha^2)...(X-\alpha^{2t}) \\ &= g_0 + g_1 X + g_2 X^2 + ... + g_{2t-1} X^{2t-1} + X^{2t}. \end{aligned} \tag{3.22}$$

Because RS codes belong to the class of cyclic codes, the data polynomial $u(X)$ of degree at most $k-1$ is encoded using the shift register as shown in Figure 2.8.

#### 3.2.2.2 Decoding

The steps of RS decoding are similar as those for decoding of BCH codes but with the additional step of determining the error values, because the codewords are not from GF(2), which need only bit flipping at the error locations. The steps are as follows [46]:

1. Compute the syndrome $\mathbf{S} = (S_1, S_2, ..., S_{2t})$ from the received polynomial $r(X)$.

   For a t-error correcting RS code, the syndrome is a $2t$ tuple over GF($q$)

   $$\mathbf{S} = (S_1, S_2, ..., S_{2t}), \tag{3.23}$$

[1] Block codes that achieve equality in the Singleton bound

where $r$ is the received polynomial $r(X) = r_0 + r_1X + r_2X^2 + ...r_{n-1}X^{n-1}$. Then the $i$ th component of the syndrome is given by

$$S_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2\alpha^{2i} + ... + r_{n-1}\alpha^{(n-1)i} \quad \text{for} \quad 1 \leq i \leq 2t. \tag{3.24}$$

The syndrome depends only on the error pattern $e(X)$ of which the coefficients are from GF($q$), so $S_i = e(\alpha^i)$. The *error pattern* has $\nu$ errors at locations $X^{j_1}, X^{j_2}, ..., X^{j_\nu}$ that is

$$e(X) = e_{j_1}X^{j_1} + e_{j_2}X^{j_2} + ... + e_{j_\nu}X^{j_\nu}, \tag{3.25}$$

where $0 \leq j_1 < j_2 < ... < j_\nu < n$. Set each $\alpha^i; 1 \leq i \leq 2t$ in (3.25) and we obtain $2t$ equations

$$\begin{aligned}
S_1 &= e_{j_1}\alpha^{j_1} + e_{j_2}\alpha^{j_2} + ... + e_{j_\nu}\alpha^{j_\nu} \\
S_2 &= e_{j_1}(\alpha^{j_1})^2 + e_{j_2}(\alpha^{j_2})^2 + ... + e_{j_\nu}(\alpha^{j_\nu})^2 \\
S_3 &= e_{j_1}(\alpha^{j_1})^3 + e_{j_2}(\alpha^{j_2})^3 + ... + e_{j_\nu}(\alpha^{j_\nu})^3 \\
&\vdots \\
S_{2t} &= e_{j_1}(\alpha^{j_1})^{2t} + e_{j_2}(\alpha^{j_2})^{2t} + ... + e_{j_\nu}(\alpha^{j_\nu})^{2t}.
\end{aligned} \tag{3.26}$$

For $1 \leq i \leq \nu$, $\beta_i = \alpha^{j_i}$ are then assigned as the *error-location numbers* and $\delta_i = e_{j_i}$ are assigned as the *error values*. If we have found $\alpha^{j_1}, \alpha^{j_2}, ..., \alpha^{j_\nu}$, the powers $j_1, j_2, ..., j_\nu$ indicate the error locations in $e(X)$. If the number of errors in $e(X)$ is $t$ or fewer, the solution is the error pattern with the smallest number of errors and it is the desired solution. Thereafter, the *error-location polynomial* is defined as

$$\begin{aligned}
\sigma(X) &= (1 - \beta_1X)(1 - \beta_2X)...(1 - \beta_vX) \\
&= \sigma_0 + \sigma_1X + \sigma_2X^2 + ... + \sigma_vX^v,
\end{aligned} \tag{3.27}$$

of which the coefficients must be determined.

2. Determine the error-location polynomial $\sigma(x)$ from $\mathbf{S}$ using the Berlekamp-Massey Algorithm or Key equation solver or Peterson-Gorenstein-Zierler

algorithm for RS codes. For details see [7], [8], [46], [52].

3. Evaluate error-location numbers $\beta_1, \beta_2, ..., \beta_v$ by finding the roots of $\sigma(X)$ using Chien's procedure [18] .

4. Determine the error-value-evaluator as

$$\begin{aligned} \mathbf{Z}_0(X) &= \sum_{l=1}^{\nu} \delta_l \beta_l \prod_{i=1, i\neq l}^{\nu} (1 - \beta_i X) && (3.28) \\ &= S_1 + (S_2 + \sigma_1 S_1)X + (S_3 + \sigma_1 S_2 + \sigma_2 S_1)X^2 && (3.29) \\ &\quad + ... + (S_\nu + \sigma_1 S_{\nu-1} + ... + \sigma_{\nu-1} S_1)X^{\nu-1}. \end{aligned}$$

   Find the error values at location $\beta_k$ in Forney's formula

$$\delta_k = \frac{-\mathbf{Z}_0(\beta_k^{-1})}{\sigma'(\beta_k^{-1})}, \tag{3.30}$$

   where

$$\sigma'(\beta_k^{-1}) = -\beta_k \prod_{i=1, i\neq k}^{\nu} (1 - \beta_i \beta_k^{-1}). \tag{3.31}$$

5. Correct the symbols in $r(X)$ at the location corresponding to error-location number by computing $r(X) - e(X)$.

#### 3.2.2.3 Erasure Decoding

Now there are $f$ erasures in addition to the $\nu$ errors, thus the erasures at positions $X^{j_1}, X^{j_2}, ..., X^{j_f}$ have erasure location numbers $\alpha^{j_1}, \alpha^{j_2}, ..., \alpha^{j_f}$. The decoder has to find locations and values of the errors and also the values at each erasure position. Because the locations of the erasures are known, the *erasure-location polynomial* can be already computed as [46]

$$\beta(X) = \sum_{l=1}^{f} (1 - \alpha^{j_l} X). \tag{3.32}$$

The syndromes $\mathbf{S} = (S_1, S_2, ..., S_{2t})$ can be computed from the received poly-

nomial $r(X)$ with the erasure locations equal to zeros resulting the *modified received polynomial* $r^*(X)$ which contains up to $f$ additional errors. The syndrome polynomial can be given as

$$S(X) = S_1 + S_2X + ... + S_{2t}X^{2t-1}. \tag{3.33}$$

We know from 3.2.2.2 that the *error-location polynomial* is

$$\sigma(X) = \sum_{k=1}^{\nu}(1 - \alpha^{i_k}X), \tag{3.34}$$

hence the error-location polynomial for the modified received polynomial $r^*(X)$ can be defined as

$$\gamma(X) = \sigma(X)\beta(X), \tag{3.35}$$

where $\beta(X)$ is known. The resulting key equation is equal to [46]

$$\sigma(X)\beta(X)S(X) \equiv Z_0(X) \mod X^{2t}, \tag{3.36}$$

where the known terms can be combined to

$$T(X) \triangleq \beta(X)S(X) \mod X^{2t}, \tag{3.37}$$

thus

$$\sigma(X)T(X) \equiv Z_0(X) \mod X^{2t}. \tag{3.38}$$

The decoder has to find the solution $(\sigma(X), Z_0(X))$ such that $\sigma(X)$ has minimum degree $\nu$ and degree $Z_0(X) < \nu + e$ via the Berlekamp-Massey algorithm or the Euclidean algorithm, where $T(X)$ is used instead of $S(X)$ in these algorithms.

When the error-location polynomial $\sigma(X)$ is found, the term $\gamma(X)$ can be evaluated. The error-value evaluator $Z_0(X)$ is also obtained from the Berlekamp-Massey algorithm or the Euclidean algorithm. At last the error values are given

by [46]

$$e_{i_k} = \frac{-Z_0(\alpha^{-i_k})}{\gamma'(\alpha^{-i_k})} \tag{3.39}$$

for $1 \leq k \leq \nu$, and the values of the erasures are [46]

$$f_{i_l} = \frac{-Z_0(\alpha^{-j_l})}{\gamma'(\alpha^{-j_l})} \tag{3.40}$$

for $1 \leq l \leq f$, where $\gamma'(X)$ is the derivative of $\gamma(X)$.

### 3.2.3 LDPC Codes

Low-density parity-check (LDPC) codes were first invented by Gallager in 1962 [25]. In 1981 Michael Tanner published his work about codes on graphs [72]. After many years of being ignored, LDPC codes on graph and iterative decoding were interested and intensively investigated by researchers in the 1990. LDPC codes with long block length and iterative message passing decoding achieve very low error rate just a fraction of decibel away from Shannon limit [46].

#### 3.2.3.1 Encoding

LDPC codes are linear block codes, so they can be specified by a generator matrix **G** or a parity-check matrix **H**. The parity-check matrix **H** is of low-density which means the number of non-zero elements in the parity check matrix is small compared to the number of rows and columns in the check matrix. To design the LDPC codes, the parity-check matrix is firstly determined, then the generator matrix can be obtained through Gauss-Jordan elimination of the parity-check matrix (details see [39]). The parameters for designing the parity check matrix **H** are code length $n$, number of message bits $k$, row weight $W_r$ and column weight $W_c$. There can be any number of check equations (but only $n-k$ of them are linearly independent), which correspond to the rows in the parity-check matrix.

LDPC codes can be visualized by a *Tanner graph*, where the check nodes and bit nodes are connected through the edges. The check nodes correspond to the parity check equations and the bit nodes correspond to the codeword bits. An

edge connects the check node $i$ to the bit node $j$ if the bit $j$ takes part in the check equation $i$ and so on. A *cycle* is defined as a path in the graph which starts and ends at the same node; besides the nodes in the path must not be repeated. The smallest cycle in the graph is called a *girth*. To design good LDPC codes suitable for iterative decoding, any 4-cycle in the graph should be avoided because they often prevent decoding from converging to the correct codeword.

#### 3.2.3.2 Combinatorial Design of LDPC Codes

We are interested in high-rate transmission of Staircase codes, thus the LDPC component codes built into Staircase blocks also need to have high rate. The high-rate LDPC codes can be constructed by combinatorial design [40], which is algebraically constructed with flexibility in selecting code length and rate online. Moreover it has specified properties such as regularity, minimum distance, girth and rate. While the random construction for high-rate LDPC codes has difficulty in removing cycles and requires more storage for $\mathbf{H}$.

Combinatorial design arranges a set of $v$ points into $b$ blocks. The design parameters are the number of points in each block $k$, and the number of blocks that contain each point $r$. The design is regular if $k$ are the same for every block and $r$ are the same for every point. The number of blocks that two points $x$ and $y$ are joined is $\lambda_{x,y}$. A regular $t$ design is denoted as $t-(v,b,r,k,\lambda)$ with every $t$-subset of points is contained in exactly $\lambda$ blocks. The design can be described by a $(b \times v)$ incident matrix $\mathbf{N}$ where each row in $\mathbf{N}$ represents a block $B_j$ and each column represents a point $P_j$ [40]

$$N_{i,j} = \begin{cases} 1, & \text{if } P_j \in B_i \\ 0, & \text{otherwise.} \end{cases} \tag{3.41}$$

The transpose of the incident matrix is used as the parity-check matrix $\mathbf{H}$ of LDPC codes. The Steiner design has $\lambda = 1$. The Steiner 2-design has $t = 2$ which guarantees no existence of cycle-4 codes.

A class of Steiner 2-design is called *Kirkman triple systems* (KTS), of which blocks can be arranged into $r$ groups called *resolution classes*, so that $v/k$ blocks of each resolution class are disjoint, and each class contains every point exactly

once. The construction method for KTS($3q$) with $v = 3q$ presented by [40] is as follows: "Let $q = 6m + 1$ be a prime power, $m$ an integer and take $\theta$ a primitive element of GF($q$), so that $\theta^{6m} = 1$, $\theta^{3m} = -1$ and $\theta^{2m} + 1 = \theta^m$. The point set is $\mathcal{H} = \mathrm{GF}(q) \times Z_3$[1] and the mixed difference system[2] consists of the sets

$$\begin{aligned} A &= \{0_1, 0_2, 0_3\} \\ B_{i,j} &= \{\theta_j^i, \theta_j^{i+2m}, \theta_j^{i+4m}\}, \qquad 1 \le i \le m \\ C_{i,j} &= \{\theta_j^{i+m}, \theta_{j+1}^{i+3m}, \theta_{j+2}^{i+5m}\}, \qquad 1 \le i \le m \\ D_{i,j} &= \{\theta_j^i, \theta_{j+1}^{i+2m}, \theta_{j+2}^{i+4m}\}, \qquad 1 \le i \le m \end{aligned}$$

for $1 \le j \le 3 \pmod 3$, where $\theta_j^i = (\theta^i, j) \in \mathcal{H}$. The sets $A, B_{i,j}$ and $C_{i,j}$ of the mixed different system make up the blocks of one *resolution class* of a design, and each translate[3] of these sets gives a further resolution class. Next, each set $D_{i,j}$ with its translate gives a resolution class, so we obtain $r = 9m+1$ resolution classes.

For example, take $\mathcal{H} = \mathrm{GF}(7) \times Z_3$, $m = 1$, $q = 7$, and $v = 21$. Choose $\theta = 3$ and the mixed different system is

$$\begin{aligned} A &= \{0_1, 0_2, 0_3\} \\ B &= \{3_1, 6_1, 5_1\}, \{3_2, 6_2, 5_2\}, \{3_3, 6_3, 5_3\} \\ C &= \{2_1, 4_2, 1_3\}, \{2_2, 4_3, 1_1\}, \{2_3, 4_1, 1_2\} \\ D &= \{3_1, 3_2, 3_3\}, \{6_2, 6_3, 6_1\}, \{5_3, 5_1, 5_2\} \end{aligned}$$

The set $A, B$, and $C$ make up the blocks of the first resolution class of the design and the six translation of these sets make up the blocks of the next six resolution classes. The blocks in the second resolution class with $g = 1$ are $\{1_1, 1_2, 1_3\}$,

[1]For an Abelian (commutative) group $\mathcal{G}$ with a number of elements $v$, $\mathcal{H} = \mathcal{G} \times Z_t$ consists of $tv$ elements, $t$ copies of each element of $\mathcal{G}$ with $(a, i) \in \mathcal{H}$ is the $i$th copy of the elements $a$ in $\mathcal{G}$

[2]For $\mathcal{H} = \mathcal{G} \times Z_t$, the $k$-elements subsets $D_1, ..., D_s \in \mathcal{H}$ form a mixed difference system, if there exists an integer $\lambda$ such that for every $i, j \in 1, 2, ..., t$, every element $g \in \mathcal{G}$ occurs $\lambda$ times as the difference $(x, i) - (y, j)$, where $g = x - y$ and $(x, i), (y, j)$ are among the elements of $D_1, ..., D_s$. If $i = j$ we have $g$ as a pure difference of class $i$, and if $i \neq j$ we have $g$ as a mixed difference of class $ij$ [64].

[3]Translate of the set $D_l$ are the set $D_l + g := \{(x + g, i) : (x, i) \in D_l\}$ for all $g \in \mathcal{G}$.

$\{4_1, 0_1, 6_1\}$, $\{4_2, 0_2, 6_2\}$, $\{4_3, 0_3, 6_3\}$, $\{3_1, 5_2, 2_3\}$, $\{3_2, 5_3, 2_1\}$, $\{3_3, 5_1, 2_2\}$. Next the translate of each block in $D$ make up a resolution class; for the first block, $D_{1,1}$, the blocks in this resolution class are $\{3_1, 3_2, 3_3\}$, $\{4_1, 4_2, 4_3\}$, $\{5_1, 5_2, 5_3\}$, $\{6_1, 6_2, 6_3\}$, $\{0_1, 0_2, 0_3\}$, $\{1_1, 1_2, 1_3\}$, $\{2_1, 2_2, 2_3\}$. There are totally 10 resolution classes, each with 7 blocks to give KTS(21,70,10,3,1). Each block defines a row of the binary incident matrix $\mathbf{N}$, which has the dimension of $(70 \times 21)$, and is constructed as in (3.41). Then the transpose of $\mathbf{N}$ is a parity-check matrix $\mathbf{H}$ for LDPC code." [40]

#### 3.2.3.3 Bit-Flipping Decoding Algorithm

We investigate only the performance of hard-decision decoding by using a bit-flipping algorithm, because soft-decision decoding that uses a sum-product algorithm is too complex and inefficient for high-rate transmission.

The bit-flipping algorithm is a hard-decision iterative form of decoding. The decoding algorithm is given in [46] as follows. Firstly, it computes the parity check sum (syndrome bits) from $\mathbf{z} = (z_0, z_1, ..., z_{n-1})$ the binary hard-decision received sequence as

$$\mathbf{s} = (s_1, s_2, ..., s_J) = \mathbf{z} \cdot \mathbf{H^T}, \tag{3.42}$$

where

$$s_j = \mathbf{z} \cdot \mathbf{h_j} = \sum_{l=0}^{n-1} z_l h_{j,l}. \tag{3.43}$$

If all the parity-check sums are zero, then stop decoding. Otherwise find the number of failed parity-check equations for each bit denoted as $f_i$ whereas $i = 0, 1, ..., n-1$. Thereafter identify the set of bits which $f_i$ is the largest and flip those bits. The algorithm repeats from computing the syndrome bits again until the parity-check sums are equal to zero or the preset maximum number of iterations is reached.

# Chapter 4

# Performance of Staircase Codes

## 4.1 G.709 Compatible Staircase Codes

The G.709 [36] recommendation of interfaces for optical transport networks by the International Telecommunication Union (ITU) defines high-rate optical transmission with code rate $R = 239/255 = 0.9373$. Thus, [6] defines a G.709 compatible Staircase code with rate $R = 0.9373$, $m = 510$, $r = 32$, where the BCH component codes have parameters $(n = 1023, k = 993, t = 3)$ with the modified generator polynomial adding two-bit error detection $(X^2 + 1)$

$$\begin{aligned} g(X) &= (X^{10} + X^3 + 1)(X^{10} + X^3 + X^2 + X + 1) \\ &\quad (X^{10} + X^8 + X^3 + X^2 + 1)(X^2 + 1) \end{aligned} \tag{4.1}$$

resulting in BCH component codes with $r = 32$ parity bits. The modified Staircase block $B_i$ has dimension 512 rows each with 510 bits $(512 \times 510)$ resulting in 261120 bits per block. The frame $B_{i-1}^T$ has dimension $(512 \times 512)$ where the two top missing rows are filled with zeros (this is to achieve compatibility with ITU G. 709). The encoding of the block of BCH component codes $(1023, 993, 3)$ is carried out with one bit shortening, so the rows of the Staircase code have dimension $512 + 510 = 1022$. The resulting rate of this modified code is equal to $R = 1 - 32/510 = 0.9373$ which is exactly the same as the rate of the code in the G.709 recommendation and indicates high-rate code. This code is the baseline code in the research of this thesis as high-rate Staircase codes on wireless channels

are investigated.

## 4.2 Performance Analysis of the Baseline Staircase Code

The performance analysis of Staircase codes is split into three parts: performance for high input bit error probability, iterative decoding threshold analysis, and performance at low bit error probability. At high input bit error probability the baseline Staircase code follows the performance of the BCH component codes, since iterative decoding does not perform; thus the decoding is equivalent to each component code that is decoded independently. The output bit error probability of Staircase codes $p_O$ can be given as [73]

$$p_O \geq \sum_{i=t+1}^{n} \frac{i}{n} \begin{pmatrix} n \\ i \end{pmatrix} p_E^i (1-p_E)^{(n-i)}, \tag{4.2}$$

where $p_E$ is the input bit error probability, $t$ is the error correction capability of the component codes, and $n$ is codeword length.

To find the decoding threshold of iterative decoding, above which the decoding is likely to fail, peeling decoding analysis is done [75], which is a simulation on pseudo decoding [41] to estimate the iterative decoding threshold of Staircase codes based on an error graph argument. The all-zero codeword with random-error patterns is processed in each row and in each column, removing the errors if the weight is at most $t$. Thus, the highest input bit error probability $p_E$ where no more errors exist in the first block of the decoding window is the estimated iterative threshold, which also depends on the size of the decoding window and the maximum number of iterations. After the input bit error probability $p_E$ is less than the iterative decoding threshold, then the tangent of the output bit-error curve is getting steep according to the product codes property, which corresponds to the waterfall region in the performance curve. In [75] the authors derive an upper bound for the iterative decoding threshold based on information transfer functions.

Density evolution (DE), which tracks the evolution of the probability density

functions through iterative decoding, can be used to find the iterative decoding threshold of an ensemble. Density evolution for generalised LDPC (GLDPC) codes and spatially-coupled GLDPC codes is proposed by [38], where they primarily define an ensemble of spatially-coupled GLDPC as a Tanner graph with ensemble $(\mathcal{C}, m, L, w)$, where $\mathcal{C}$ is a binary linear code with $(n, k, d_{min})$ with $d_{min} \geq 2t+1$ and $t$ is the error correction capability, $L$ is the number of positions of bit nodes contained in spatially-coupled GLDPC, $m$ is a selected number such that $mn$ is divisible by 2 and $w$. There are $N = \frac{mn}{2}$ degree-2 bit nodes and $m$ degree-$n$ code-constraint nodes at each position, thus there are totally $mn$ output connections from all bit nodes and $mn$ output connections from all code-constraint nodes at each position, which are called sockets. "The $mn$ sockets are separated to $w$ groups of $\frac{mn}{w}$ sockets via uniform random permutation $\pi$ at each bit position and code-constraint position. The $j$-th group of the bit nodes at the $i$-th position is called $\mathcal{S}^b_{i,j}$ with $j \in \{0, 1, ..., w-1\}$, and the $j$-th group of the code-constraint nodes at the $i$-th position is called $\mathcal{S}^c_{i,j}$ with $j \in \{0, 1, ..., w-1\}$. The Tanner graph is constructed by connecting $\mathcal{S}^b_{i,j}$ to $\mathcal{S}^c_{i+j,w-j-1}$" [38]. This annotation can be illustrated as the yellow lines in Figure 4.1. The iterative decoding of Staircase codes with the bound minimum distance decoder can be interpreted as removing the "error graph", which has the code-constraint node of each component code connecting (by edges) to its erroneous bit nodes. In iterative decoding process, we remove those code-constraint nodes, their edges, and their connected erroneous (but correctable) bit nodes from the error graph, for which the number of the connected edges to a check node is not bigger than the error correction capability $t$ of the component codes. The error graph has a high probability to be a tree, which is necessary for a message passing algorithm to guarantee the correctness of the result, when $m \to \infty$, thus the iterative decoding can be analysed through density evolution. In [32] the density evolution is used to predict the performance curve at the iterative decoding threshold, and the waterfall region for Staircase codes by assigning $w = 2$, since each position of the Staircase codes are connected to 2 code-constraint positions as illustrated in Figure 4.1. The message from bit node $i$ to constraint node $j$ is in error, when the bit node $i$ is in error and the incoming messages from other constraint nodes $j' \neq j$ are in error, which happens when there are at least $t$ errors in those constraint nodes. The error probability of

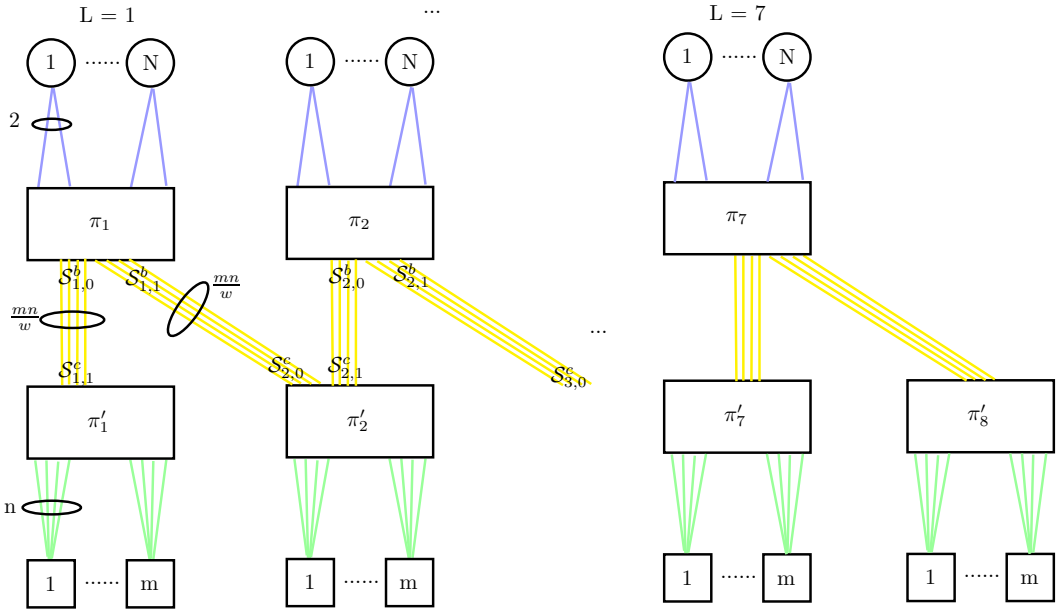



Figure 4.1: Tanner graph of Staircase code with $L = 7, w = 2$ (derived from [38])

the bit nodes at position $i$ in the $(l+1)$-th iteration for ideal component decoders without *error decoding* can therefore be given as [38]

$$x_i^{(l+1)} = p\hat{f}_n(x_i^{(l)}), \tag{4.3}$$

where $x_i^{(l)} = 0$ for all $i \notin \{1, 2, ..., L\}$, $l \geq 0$, $p$ is the bit error probability of the BSC, and

$$f_n(x) \triangleq \sum_{i=t}^{n-1} \frac{i}{n} \begin{pmatrix} n-1 \\ i \end{pmatrix} x^i (1-x)^{(n-i)}. \tag{4.4}$$

From the spatially-coupled GLDPC codes ensemble definition, the average of input bit error probability at position $i$ is given as [38]

$$y_i^{(l)} = \frac{1}{w} \sum_{j=0}^{w-1} x_{i-j}^{(l)}, \tag{4.5}$$

therefore the bit error probability of the bit nodes at position $i$ in the $(l+1)$-th

iteration is given as

$$x_i^{(l+1)} = p\left(\frac{1}{w}\sum_{k=0}^{w-1} f_n\left(\frac{1}{w}\sum_{j=0}^{w-1} x_{i-j+k}^{(l)}\right)\right). \tag{4.6}$$

We track the evolution of the bit error probability of the baseline Staircase code ensemble (BCH $(1023, 993, d_{min} = 7)$, m=512, L=7, w=2) with $x_i^{(0)} = 0.0051$ for $i \in L$ and plot it in Figure 4.2. The evolution starts at the upper right corner and attempts to reach the origin, where the bit error probability is zero. The minimum input bit error probability $x^{(0)}$ for which the average bit error probability $\bar{x}^{(l)}$ with $l \to \infty$ gets stuck at the bisectrix is the estimated iterative decoding threshold. For this ensemble the iterative threshold is found to be at $\bar{x}^{(0)} = 0.0051$. Figure 4.3 shows the case with $\bar{x}^{(0)} = 0.0050$ that the iterations succeed in arriving at the origin after so many iterations. It can be observed that the decoding trajectory slows down when it enters a bottleneck or tunnel region [22] near the bisectrix line [45].

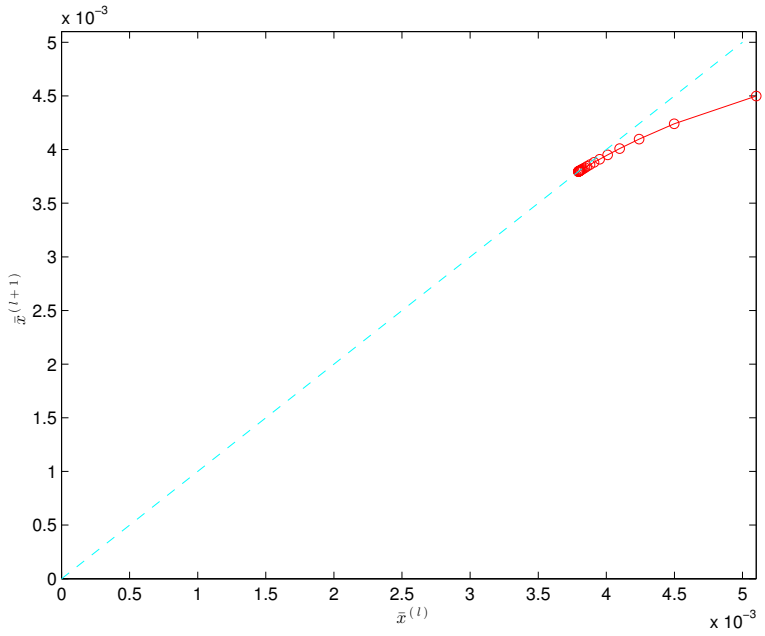



Figure 4.2: Evolution of the bit error probability with $\bar{x}^{(0)} = 0.0051$

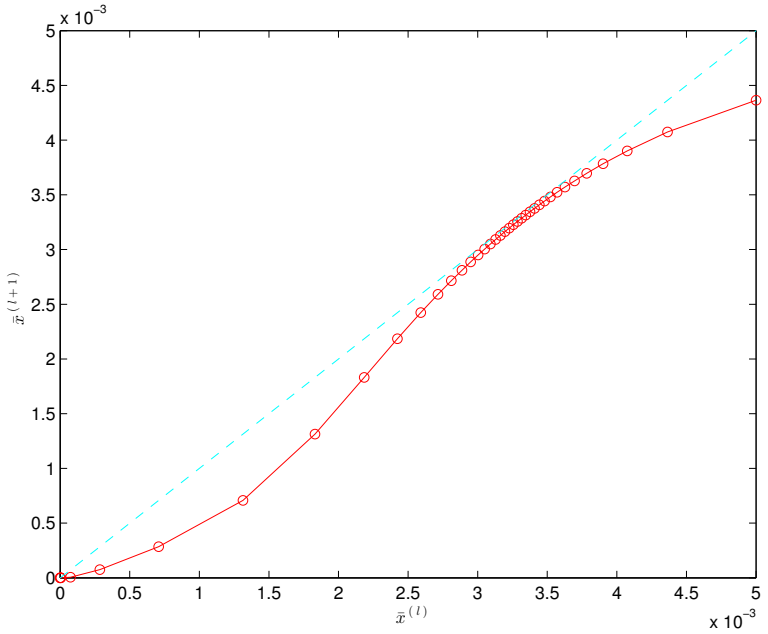



Figure 4.3: Evolution of the bit error probability with $\bar{x}^{(0)} = 0.0050$

At last the performance curve in waterfall region can be extrapolated to reach the error floor at a very low output bit error probability, which depends on the probability that an error pattern contains a stall pattern. The stall pattern is "a set $s$ of codeword positions, for which every row and column involving positions in $s$ has at least $t+1$ position in $s$, and thus the decoder gets locked in $s$ state in which no updates are performed" [6]. This definition is equivalent to the $(t+1)$-core pattern of product codes (see Section 2.8.1); however the analysis of Staircase codes involves two consecutive blocks $B_i$ and $B_{i+1}$ instead of one block $B$ in the product codes analysis.

The error floor of the baseline Staircase codes can be estimated using the *Union Bound Technique* [6]. The error floor equals the probability that a stall pattern is assigned to block $B_i$, i.e., and possibly $B_{i+1}$ given

$$\mathrm{BER}_{\mathrm{floor}} \leq \sum_{s \in \mathcal{S}_i} \Pr[\text{bits in } s \text{ in error}] \cdot \frac{|s|}{510^2}, \tag{4.7}$$

where each block has 510 rows, 510 columns, and the component codes have error correction capability $t = 3$. To enumerate the set $\mathcal{S}$, [6] gives the bounding contribution due to the minimal stalls, in which only $t+1$ rows have positions in $s$ and only $t+1$ columns have positions in $s$. Thus the multiplicity of minimal stall patterns for $B_i$ is given as

$$M_{min} = \begin{pmatrix} 510 \\ 4 \end{pmatrix} \cdot \sum_{m=1}^{4} \begin{pmatrix} 510 \\ m \end{pmatrix} \cdot \begin{pmatrix} 510 \\ 4-m \end{pmatrix}, \tag{4.8}$$

where the first term $\left(\binom{510}{4}\right)$ is for the two consecutive blocks sharing the same $t+1$ rows (or columns) in $s$. The sum term is for $t+1$ columns (or rows) in $s$, where $m$ columns (or rows) of the $t+1$ columns (or rows) are in the first block and $t+1-m$ columns (or rows) of the $t+1$ columns (or rows) are in the other block. When $p$ is the probability that the received bit is in error, and $\zeta$ is the erroneous bit flips that occur due to incorrect decoding, the upper bound of the probability that a particular minimal stall $s$ occurs is [6]

$$\sum_{l=0}^{16} \begin{pmatrix} 16 \\ l \end{pmatrix} p^{16-l}\zeta^{l} = (p+\zeta)^{16}. \tag{4.9}$$

Then the error floor contribution due to *minimal stall patterns* can be estimated as [6]

$$\text{BER}_{\text{floor}} \approx \frac{16}{510^2} \cdot M_{min} \cdot (p+\zeta)^{16} \tag{4.10}$$

where $\zeta = 5.8 \times 10^{-4}$ when $p = 4.8 \times 10^{-3}$. For details in calculating the error floor's contribution due to *non minimal stalls* see [6]. The error floor, which the dominant contribution is due to minimal stall patterns, is estimated to occur at $4.0 \times 10^{-21}$ [6], which is very hard to get to by simulation; thus the extrapolation of the performance curve is usually done for these kinds of codes.

## 4.3 Performance Simulations of the Baseline Staircase Code

In Figure 4.4 the performance simulation of the baseline Staircase code is compared with the performance simulation of a Reed-Solomon code of the same rate and with the theoretical performance of the BCH (1023, 993, t = 3) component codes from Equation 4.2. At high input bit error probability with $p_E > 0.008$ the performance simulation of the Staircase code follows the theoretical performance of the BCH component code, as the BCH component codes cannot correct more errors than the error correction capability of the BCH component codes, and for a bad channel such error patterns occur often. It can be seen that at input bit error probability $p_E = 0.0049$, the Staircase code performs with the output bit error probability $p_O = 2.79 \times 10^{-7}$, which is much better than the theoretical performance of the BCH component codes and the RS code of the same rate. At $p_E = 0.0048$ the $p_O$ is marked at $10^{-8}$ because the simulation finds no errors in $10^9$ data bit realizations[1]. The abrupt improvement of the output bit error probability from the theoretical performance of the BCH component code is known as the water fall region. This is due to the iterative decoding of Staircase codes that reduces the number of errors in each iteration processed (for channel better than the iterative decoding threshold). The estimated iterative decoding threshold from density evolution in the last section results at $p_E = 0.0051$ for the baseline Staircase code ensemble, which defines the upper bound, above which, the iterative decoding will definitely fail. Our Monte Carlos simulation with MATLAB can only run to have a confidence value of $p_O$ above $10^{-8}$. For $p_O$ smaller than $10^{-15}$ as required for high-rate optical transport networks, hardware simulations, e.g. by an FPGA realization, would be required, but this was out of scope of this

[1] According to [44, Theorem 2.4] the confidence interval for a probability $p$ of "success" (a bit error in our case) when we observe $z = 0$ successes in an experiment of $n$ independent trials is $[0; p_0(n)]$. For $n$ large and for a confidence of 0.95 the upper limit $p_0(n)$ is approximately given by $p_0(n) \approx \frac{3.689}{n}$, with the relative error smaller than 10% for $n \geq 20$. A condition for this result to hold is, however, that the trials must be independent, and, as we are considering bit-error probability also within code words, independence can not be guaranteed. Therefore, well knowing that the error probability is not zero, we have decided to mark the fact that no error has been found in $10^9$ trials by marking the point at $10^{-8}$, which is more conservative than the bound $p_0(n) \approx 3.689 \cdot 10^{-9}$ one would obtain from the theorem in [44].

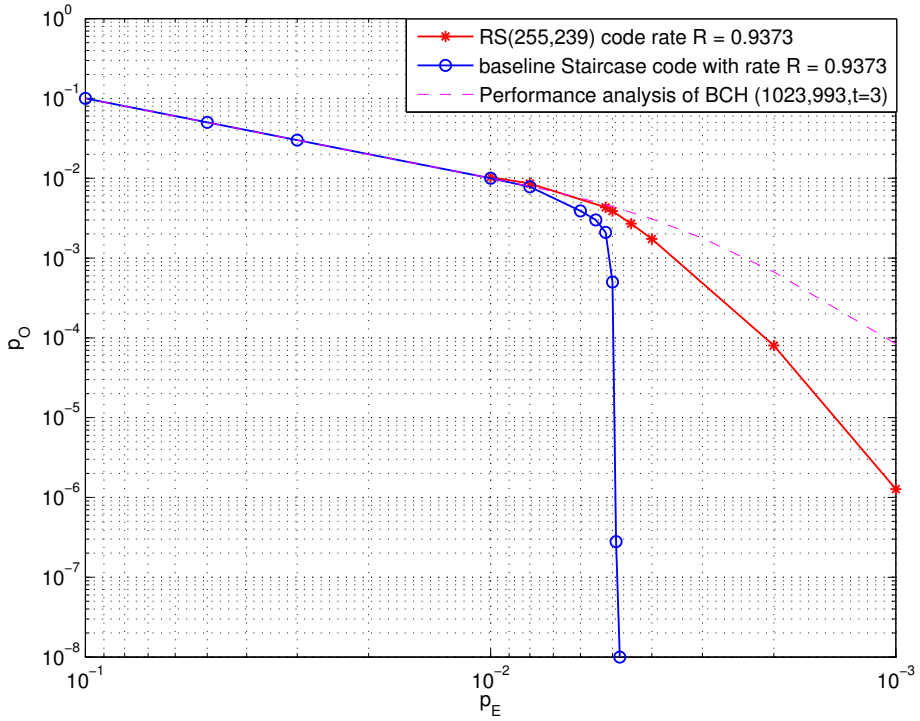



Figure 4.4: Simulation of the baseline Staircase code compare with RS code from G.709 and the theoretical performance of BCH component codes.

research.

Figure 4.5 shows the number of decoding iterations that some random error sequences decoded by the baseline Staircase code require to converge to a stable state. The curves are shown for different $p_E$ with a decoding window of size 7. Here we define the stable state as when the decoding of the next iteration does not change the number of errors decoded in a codeword and the iterations can be stopped. It can be seen that at $p_E = 0.0052$ and $p_E = 0.008$, which are greater than the successful decoding probability at $p_E = 0.0048$ shown in Figure 4.4 with the output bit error probability $p_O \leq 10^{-8}$, the codewords require fewer iterations (about 2 to 5) because the iterations stop early on the uncorrectable error patterns. At $p_E = 0.0048$ the number of iterations required varies between 5 to 13. At $p_E = 0.0045$ which is below the successful decoding probability, the number of iterations required is between 4 to 10, which is less than the iterations required at the $p_E = 0.0048$ because there are less errors to be corrected. Later on we fix the number of decoding iterations in every simulation to 7 such that the input bit error $p_E$ below 0.0048 are still correctable and the decoding time is moderate.

## 4.4 High Error Floor of Staircase Codes with Small-$t$ Component Codes

The probability $P_E$ of *decoding error* for $t$-error-correcting RS codes is given as [51]

$$P_E = \sum_{u=0}^{n} P_E(u) q_u \tag{4.11}$$

where $q_u$ is the probability that an error pattern has weight $u$, $P_E(u)$ is the conditional probability of decoder error given $u$ channel errors, and $n$ is codeword length. The upper bound on $P_E(u)$ assuming all error patterns of the same weight are equiprobable, is given as [51]

$$P_E(u) \leq \frac{1}{t!} \quad \text{for all} \quad u \geq t+1. \tag{4.12}$$

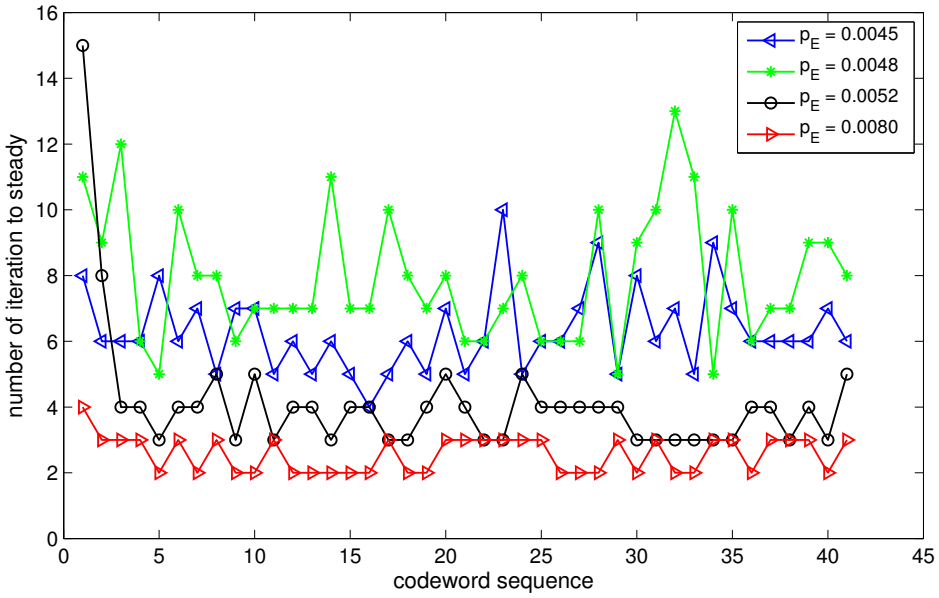



Figure 4.5: Number of decoding iterations required for random sequences of baseline Staircase code on different input bit error probability.

Therefore, when there are more than $t$ errors in a received codeword, the probability of decoder error, which means the decoder selects a wrong codeword, is upper bounded by $\frac{1}{t!}$: this can also be adopted for BCH codes that are a subcode of RS codes.[1] Accordingly, for small $t$-error correcting codes the probability of decoder error is higher than for large $t$-error correcting codes. The probability of decoder error of RS/BCH component codes of product codes can be neglected for moderate to large value of $t$ as it is quite small, e.g., for $t = 5$ the probability of decoder error is upper bounded by 0.0083. However, small values of $t$, where $t = 2 - 4$, e.g., for $t = 4$ with the probability of decoder error 0.0416, contribute more to additional events that increase the probability of decoder errors of the RS/BCH component codes. Therefore the iterative decoding of product codes with small-$t$ component codes may stop with a stall pattern, which is not part of the original pattern, thus causing higher error floor in the product codes [41]. The statement is also valid for Staircase codes, because Staircase codes are a kind of product code and also are decoded iteratively. Based on this, [41] suggested using binary codes of even weight, or introducing extra parity symbols, or decoding only up to $t - 1$ errors, to decrease decoding error of the component codes, such that no more error bits are inserted in to the blocks of product codes, which worsen the iterative decoding. The error floor of the product codes, therefore, is lowered. The main purpose is to avoid decoding the most unconfident words located between two nearest-neighbour codewords. Using block codes of even weight, of which $d_{min}$ is even, in BMD decoder that has error correction capability $t = \lfloor \frac{d_{min}-1}{2} \rfloor$, the decoder leaves words of distance $d_{min}/2$ between two nearest-neighbour codewords undecided; because they locate outside of the decoding bound of the BMD decoder. In contrast to using block codes of odd weight, of which $d_{min}$ is odd, of the same error correction capability $t = \lfloor \frac{d_{min}-1}{2} \rfloor$, the BMD decoder leaves no words between two nearest-neighbour codewords undecided. Hence the error decoder probability of codes with even weight is lower than codes with odd weight for the same error correction capability $t$. Similarly decoding only up to $t - 1$ leaves some words that is further away with distance

[1] According to [34] let $\mathcal{C}'$ be an RS code over the extension field GF($q^m$) that has a set of codewords over GF($q^m$) of length $N = q^m - 1$. Thus the codewords $c'$ are in $[\mathrm{GF}(q^m)]^N$, but some of them are in $[\mathrm{GF}(q)]^N$, which form the BCH code $c$. The BCH code $c$ is, thus, a set of all RS codewords that are in $[\mathrm{GF}(q^m)]$.

$t$ from codewords undecided, such that the decoder error probability is lowered. Using extra parity check symbols controls the correctness of the decoded component codewords. If they are not correct, the initial words are left unchanged, such that no more error bits are inserted into the array of product codes or Staircase codes during iterative decoding.

We investigate by performance simulation whether adding a Cyclic Redundancy Check (CRC) to the component code lowers the error floor of Staircase codes. The baseline Staircase code has BCH codes ($n = 1023, k = 993, t = 3$) as component codes, which are small-$t$ component codes, and it has the component codes extended by 2 bit CRCs. We simulated a Staircase code with BCH ($n = 1023, k = 993, t = 3$) component codes without the additional 2 bit CRCs for comparison that has the generator polynomial

$$\begin{aligned} g(X) &= (X^{10} + X^3 + 1)(X^{10} + X^3 + X^2 + X + 1) \\ &\quad (X^{10} + X^8 + X^3 + X^2 + 1) \end{aligned} \tag{4.13}$$

with the overall rate equal to 0.9412.

For smaller $t$, the Staircase code with BCH ($n = 1023, k = 1003, t = 2$) component codes was simulated, of which the generator polynomial can be given as

$$g(X) = (X^{10} + X^3 + 1)(X^{10} + X^3 + X^2 + X + 1) \tag{4.14}$$

with the overall rate of 0.9608. For the Staircase code with BCH ($n = 1023, k = 1003, t = 2$) component codes that have additional 2 bit CRCs and an overall rate equal to 0.9569, the generator polynomial of the component codes can be given as

$$\begin{aligned} g(X) &= (X^{10} + X^3 + 1)(X^{10} + X^3 + X^2 + X + 1) \\ &\quad (X^2 + 1). \end{aligned} \tag{4.15}$$

Moreover the Staircase code with BCH ($n = 1023, k = 983, t = 4$) component

codes was simulated, of which the generator polynomial can be given as

$$\begin{aligned} g(X) &= (X^{10}+X^3+1)(X^{10}+X^3+X^2+X+1) \\ &\quad (X^{10}+X^9+X^8+X^7+X^6+X^5+X^4+X^3+1) \\ &\quad (X^{10}+X^8+X^3+X^2+1) \end{aligned} \tag{4.16}$$

with the overall rate of 0.9216. For the Staircase code with BCH $(n = 1023, k = 983, t = 4)$ component codes that have additional 2 bit CRCs and the overall rate equal to 0.9176, the generator polynomial of the component codes can be given as

$$\begin{aligned} g(X) &= (X^{10}+X^3+1)(X^{10}+X^3+X^2+X+1) \\ &\quad (X^{10}+X^9+X^8+X^7+X^6+X^5+X^4+X^3+1) \\ &\quad (X^{10}+X^8+X^3+X^2+1)(X^2+1). \end{aligned} \tag{4.17}$$

It can be observed in Figure 4.6 that the Staircase code with BCH (1023, 993, t = 3) without CRC shows a higher error floor at $p_O = 2 \times 10^{-7}$ with $p_E = 0.002$ in contrast to the baseline Staircase code with 2 bit CRCs, which shows steep a waterfall region at $p_E = 0.0048$. The Staircase code with BCH (1023, 1003, t = 2) without 2 bit CRCs shows a higher error floor at $p_O = 2 \times 10^{-6}$ with $p_E = 0.001$, but the Staircase code with BCH (1023, 1003, t = 2) with 2 bit CRCs shows a steep waterfall region and is even better than the Staircase code with BCH (1023, 993, t = 3) without CRC, which has a smaller rate, at $p_E < 0.00285$. The Staircase code with BCH (1023, 983, t = 4) with 2 bit CRCs has a waterfall region at $p_E = 0.007$, where the Staircase code with BCH (1023, 983, t = 4) without CRC has an error floor at $p_O = 1.43 \times 10^{-6}$ with $p_E = 0.0061$. Even though the BCH component codes of two Staircase codes have the same error correction capability $t$, the insertion of 2 bit CRCs improves the performance significantly and eliminates the high error floor. This is because when the CRCs detect errors in the decoded codewords, the initial bits are left unchanged; hence error decoding, which causes more flipped bits in iterative decoding, is avoided. Nevertheless, errors may have patterns that are sometimes undetectable by CRCs; thus it has impact on the performance of Staircase codes. The *undetected error*

*probability* $P_{ud}$ for BSC can be given as [3], [81]

$$P_{ud}(\mathcal{C}, \epsilon) = \sum_{i=d}^{n} A_i \epsilon^i (1-\epsilon)^{n-i}, \tag{4.18}$$

where the bit error probability $\epsilon \in [0, 1/2]$, $A_i$ is the weight distribution of the code $\mathcal{C}$, $d$ is the minimum distance of the code $\mathcal{C}$, and $n$ is the code length. This formula requires the knowledge of the weight distribution of the code, which is unknown for most codes except Hamming codes, maximum distance codes, e.g., the Reed-Solomon-Codes [34]. A simple approximation of undetected error probability for BSC when $\epsilon = 1/2$ is given as [81]

$$P_{ud} < 2^{-r}, \tag{4.19}$$

where $r$ is the number of parity check bits. In our simulation, the undetected errors for each row of BCH codeword with 2 bit CRCs have $P_{ud} < 2^{-2}$.

For a Staircase code with RS (n = 255, k = 247, t = 4) component codes, which are small-$t$ component codes, we compared the performance to the Staircase code with RS (n = 255, k = 247, t = 4) component codes extended by 2 bit CRCs, which has the overall rate $R \approx 0.9352$. The component codes of these two Staircase codes are decoded to $t$ errors. Furthermore, we simulated the Staircase code with RS (n = 255, k = 247, t = 4) component codes decoded to $t-1$ errors, which means the decoding makes a correction to codewords that have up to $t-1$ errors, while codewords with $t$ errors are left unchanged.

As can be observed in Figure 4.7 the performances of the Staircase codes with component codes decoded to $t$ errors have the iterative decoding thresholds at $p_E > 0.003$, which are higher than that decoded to $t-1$ errors, and has the iterative decoding threshold at $p_E \approx 0.0027$. However the curve with component codes decoded to $t$ errors shows a higher error floor in contrast to those decoded to $t-1$ errors, where no error floor is observed in our simulations. The Staircase code with CRC has a waterfall region at a higher input bit error probability $p_E$ due to the lower rate, but the error floor is still able to be seen, nevertheless the error floor occurs at a lower output bit error probability $p_O$ than the RS Staircase code without CRC.

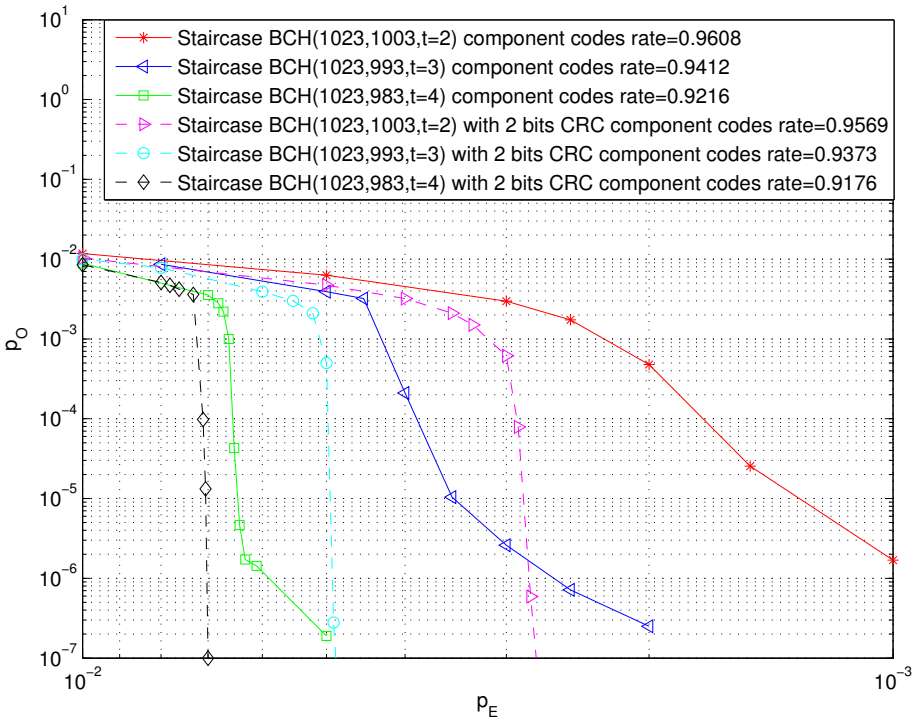



Figure 4.6: Performance comparison of Staircase codes with and without 2 bit CRCs.

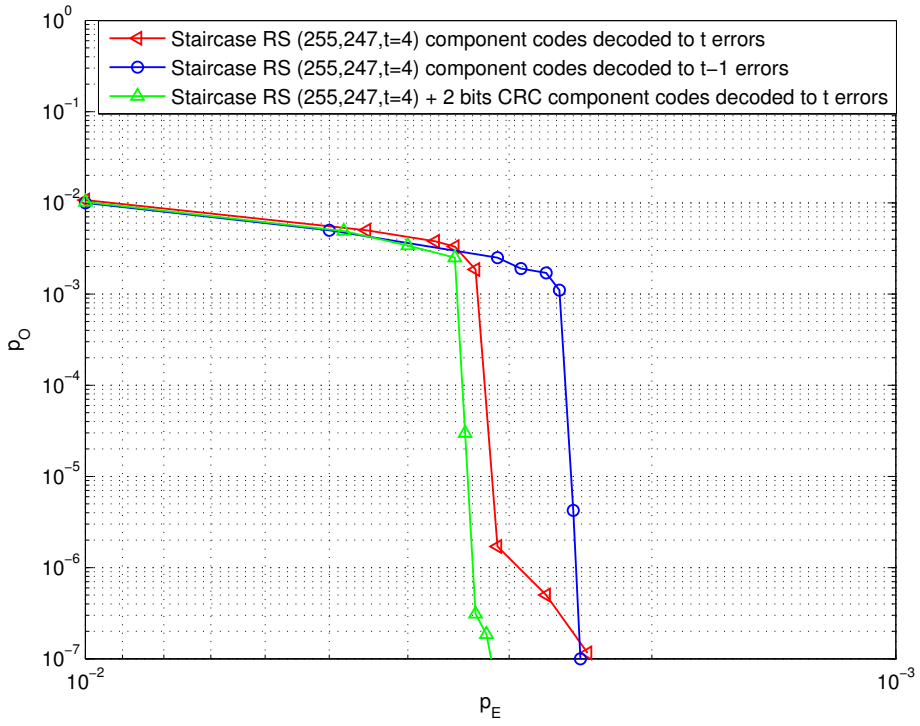



Figure 4.7: Performance comparison of Staircase code with RS component codes decoded to $t$ errors with and without 2 bit CRCs and decoded to $t-1$ errors.

## 4.5 Performance of Staircase Codes with LDPC Component Codes

LDPC codes with soft-decision decoding using the sum-product algorithm are capacity-approaching codes (for some code designs). However LDPC codes with hard-decision decoding using the bit flipping-algorithm do not perform very well. The Staircase code structure improves the performance of the component codes. Therefore we investigated the LDPC component codes in the Staircase code structure to determine whether they can compete with other hard-decision algebraic component codes.

The systematic LDPC codes are integrated into the block structure of the Staircase code. The selected LDPC (n=1023, k=930) codes with rate 0.9091 are constructed from the Kirkman triple system (KTS) which enables the construction with specified properties such as regularity, minimum distance, girth and rate of high-rate LDPC codes [39]. The decoding of LDPC component codes is based on bit-flipping algorithms, which are used for hard-decision decoding, because in the decoding of high-rate codes, the complexity is critical, and therefore hard-decision decoding is preferred.

As can be seen in Figure 4.8 the performance of the Staircase code with LDPC component codes is inferior to the baseline Staircase code, even though it has smaller code rate. The increase in the maximum number of decoding iterations of the LDPC component codes from 100 to 1000 does not help improving the performance of LDPC Staircase code. A high error floor can be observed at input bit error probability $p_E < 0.0025$, in contrast to the baseline Staircase code where no error floor can be seen. As a consequence the selected LDPC component codes are not suitable for high-rate Staircase codes using hard-decision decoding. We note here that the investigation is not exhaustive, where only one LDPC code design with one option of hard-decision decoding is performed.

Even though soft-decision decoding of the LDPC component codes by the sum-product algorithm can improve the performance of the LDPC Staircase codes, we do not consider it due to the complexity and latency of soft-decision decoding. Nevertheless LDPC codes with soft-decision decoding could be of interest, if the latency and the complexity were not restricted, e.g., [88] proposed a rate adaptive

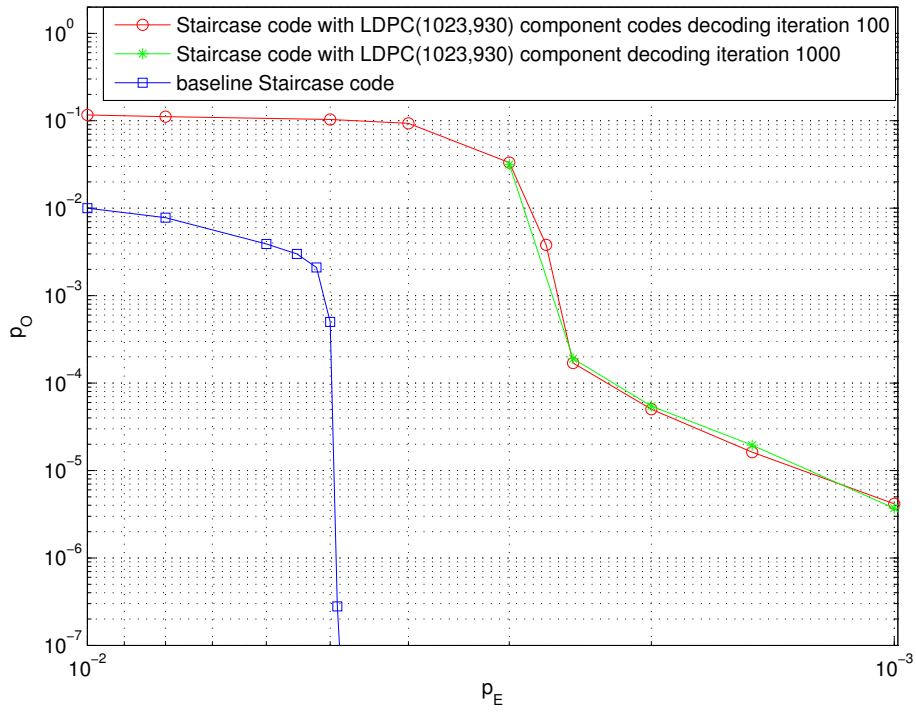



Figure 4.8: Comparison of the baseline Staircase code to the Staircase code with LDPC(1023,930) component codes.

LDPC in Staircase structure decoded by sum-product algorithm, this is beyond the scope of this research.

## 4.6 Conclusion

The high-rate G.709-compatible Staircase code with BCH component codes and the parameters for the iterative decoding of these codes were investigated in this chapter. The performance analysis started at high input bit error probability, where the performance of Staircase codes followed the performance of BCH component codes. When the input bit error probability falls below the iterative decoding threshold, the iterative decoding of Staircase code improves the performance significantly and can be seen as the water fall region in the performance

curve. The iterative decoding threshold can be obtained in many ways, such as, peeling decoding analysis, information transfer function or density evolution. At a low input bit error probability, iterative decoding cannot correct the errors in stall patterns, even though the input bit error probability is decreased; hence this can be seen as the error floor in the performance curve. The performance simulation corresponds to the theoretical performance analysis. However, the error floor from the analysis yields very low values, which cannot be verified by simulation.

High error floors in the performance curves of some Staircase codes are due to small-$t$ errors correction capability of the component codes, which causes higher probability of error decoding of the component codes. This high error floor can be avoided by introducing extra parity symbols to the component codes, or decoding the component codes only up to $t - 1$ errors to prevent the erroneous decoding of the component codes and, hence, even more bit errors.

The implementation of high-rate LDPC codes to the block of Staircase codes, which are decoded using bit-flipping algorithms, shows inferior performance to the Staircase codes that use usual algebraic component codes. This is due to the poor performance of low-complexity hard-decision decoding of the LDPC component codes.

# Chapter 5

# Staircase Codes for High-Rate Wireless Transmission on Burst-Error Channels

[1] Staircase codes were initially designed for high-rate fiber optic transmission to correct errors in a BSC channel; however to use them on burst-error channels some aspects require close attention. The RS codes (which are known to have good performance for correcting burst errors [46], [62], [52] because the codewords are encoded and decoded symbol-wise) are implemented in the Staircase blocks and tested on channels with different burst length. Block interleaving of the codeword prior to transmission is one method to combat burst errors, thus it is implemented and tested on burst-error channels. Lastly, complexity and decoding latency of both types of component codes are given which have direct influence on decoding latency of Staircase codes that can be achieved on a given hardware.

## 5.1 Gilbert-Elliott Model for Burst-Errors

Gilbert [29] and Elliott [24] defined a two-state Markov model for a burst-noise binary channel, which depends on the channel states "good (G)" and "bad (B)"

[1] Part of this Chapter has been published in the IEEE Wireless Communications Letters [43]

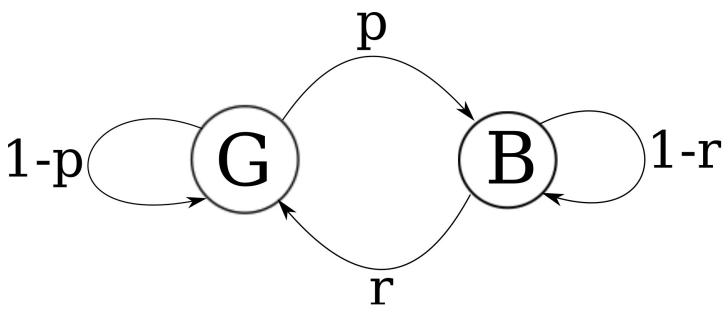



Figure 5.1: Gilbert-Elliott model generating burst errors.

as can be seen in Figure 5.1. In the Gilbert model, errors are generated with probability 0 in the "good state" and probability $b$ in the "bad state", whereas, for the Elliott model, errors can also occur in the "good state" with probability $g$. The probability of transition from the "good state" to the "bad state" is

$$p = P(Q_t = B \mid Q_{t-1} = G), \tag{5.1}$$

and the probability of transition from the "bad state" to the "good state" is

$$r = P(Q_t = G \mid Q_{t-1} = B). \tag{5.2}$$

The unconditional ("steady state") probability of the "good state" is [33]

$$\pi_G = \frac{r}{p+r}, \tag{5.3}$$

and the probability of the "bad state" is

$$\pi_B = \frac{p}{p+r} \tag{5.4}$$

with the resulting error rate equal to

$$p_E = g\pi_G + b\pi_B. \tag{5.5}$$

As the time spent in one state until leaving for the other state does *not* depend on the probability of transition into that state, the probability distribution of how

long the channel stays in a particular state is a geometric random variable. The probability to stay in the "bad state" for $T = \tau$ time instances is, therefore, [85]

$$P_T(\tau) = \begin{cases} r(1-r)^{\tau-1} & \text{for } \tau = 1, 2, ... \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

The average burst length is the expected value of the time staying in the "bad state" which is (with (5.6)) calculated as

$$\Delta_B = E[T] = 1/r. \tag{5.7}$$

The average length of an "error free" interval is derived the same way resulting in

$$\Delta_G = 1/p. \tag{5.8}$$

To specify all parameters of the model as functions of the desired error probability $p_E$ and the desired average burst-error length $\Delta_B$, hence $g = 0$ (i.e., no errors in the good state) and $b = \frac{1}{2}$ (50% error probability in the bad state) are assigned such that the total error probability depends only on the probability $\pi_B$ to be in the "bad state" [54]:

$$p_E = \frac{1}{2}\pi_B. \tag{5.9}$$

The probability of changing from "bad state" to "good state" is with (5.7)

$$r = \frac{1}{\Delta_B}. \tag{5.10}$$

Using (5.4) in (5.9), the probability of changing from "good state" to "bad state" is given as

$$p = \frac{2p_E}{\Delta_B(1-2p_E)}. \tag{5.11}$$

With the parameter settings according to (5.10) and (5.11) and with $g = 0$ and $b = \frac{1}{2}$ all channel model parameters are now determined by the desired average

burst length $\Delta_B$ and the desired error probability $p_E$.

Figure 5.2 shows the distribution of the simulated burst length with average burst lengths of 2, 10, 30 and 80 bits with parameter $p_E = 0.0045$ compared to values computed from the Equation in 5.6. The simulation results accurately match the probability distribution equation. It can be observed that bursts with shorter burst lengths than the average value occur more often than bursts with longer burst lengths.

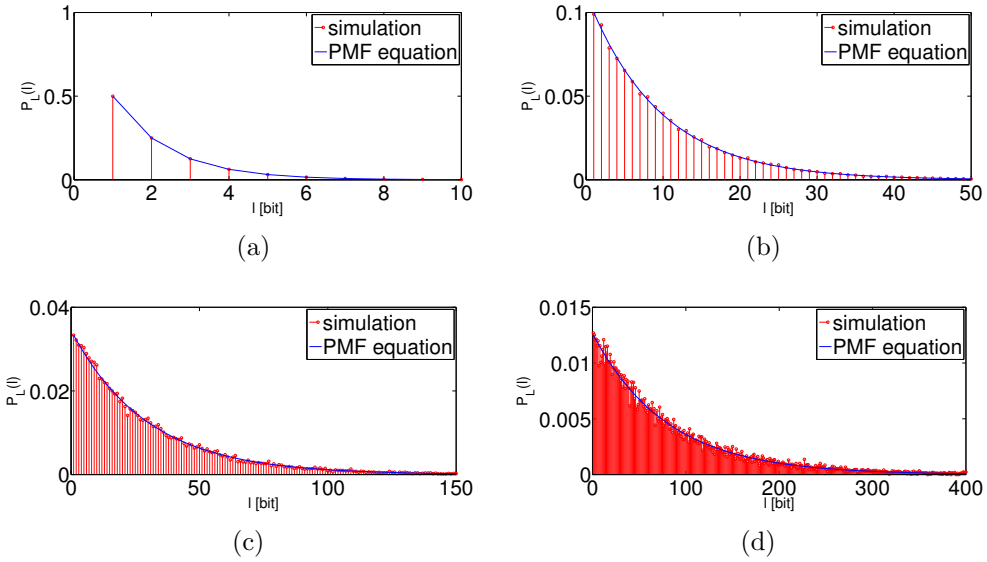



Figure 5.2: Burst length distribution with average burst lengths (a) 2 bits (b) 10 bits (c) 30 bits (d) 80 bits with $p_E = 0.0045$.

## 5.2 Capacity of Gilbert-Elliot Channel

As stated in [53] the capacity $C_\mu$ of the channel with memory $\mu \triangleq 1 - p - r$ is lower bounded by the channel without memory, it increases monotonically with $\mu$ and converges asymptotically to the capacity of the channel where the side information about its instantaneous state is available at the receiver. The capacity in terms of the input sequences $\mathbf{x}_l$ and the output sequences $\mathbf{y}_l$ of length

$l$ is given by

$$C = \lim_{l\to\infty} \frac{1}{l} \max_{P(\mathbf{x_l})} I(\mathbf{x}_l, \mathbf{y}_l). \tag{5.12}$$

Besides, the capacity of the Gilbert-Elliot channel in bits per channel use in terms of error, the process $\mathbf{z}_l$ is given by [53]

$$C = 1 - \lim_{l\to\infty} \frac{1}{l} H(\mathbf{z}_l), \tag{5.13}$$

and also

$$C = 1 - \lim_{l\to\infty} E[h(q_l)], \tag{5.14}$$

where $H(\cdot)$ is the entropy rate, $z_l = x_l \oplus y_l$ and $h(\cdot)$ is the binary entropy function given as

$$h(q) \triangleq -q \log q - (1-q) \log(1-q). \tag{5.15}$$

The random variable $q_l$ is the probability of channel error at the $l$th use of the channel, conditioned on the channel errors at its previous uses given as

$$q_l = \Pr(z_l = 1 \mid \mathbf{z}_{l-1}). \tag{5.16}$$

The random variable $q_l$ can be computed recursively as [53]

$$q_{l+1}(z_l) = v(z_l, q_l(z_{l-1})) \tag{5.17}$$

with the function $v(\cdot,\cdot)$ which is defined as follows:

$$v(0,q) \triangleq \begin{cases} p_G + p(p_B - p_G) + \mu(q - p_G)\frac{(1-p_B)}{(1-q)} & \text{if } p_B \neq 1, \\ (1-p)p_G + p, & \text{if } p_B = 1, q \neq 1, \end{cases} \tag{5.18}$$

and

$$v(1,q) \triangleq \begin{cases} p_G + p(p_B - p_G) + \mu(q - p_G)(p_B/q) & \text{if } p_G \neq 0, \\ (1-r)p_B, & \text{if } p_G = 0, q \neq 0, \end{cases} \tag{5.19}$$

where $p$ and $r$ correspond to the probability of transition from good state (G) to bad state (B) and vice versa. The notation $p_G$ is the probability of generating errors in the good state and corresponds to $g$ in Section 5.1. The notation $p_B$ is the probability of generating errors in bad state and corresponds to $b$ in Section 5.1.

The initial value for the recursion is given after [63] as

$$\begin{aligned} q_0 &= \Pr(z_0 = 1) \\ &= \Pr(q_0 = G)p_G + \Pr(q_0 = B)p_B \\ &= \pi_G p_G + \pi_B p_B, \end{aligned} \tag{5.20}$$

which is the error probability $p_E$ of the steady state, such that the stationarity of the process is guaranteed.

To compute the capacity from Equation 5.14 the complexity is increased exponentially with $l$, thus [63] proposed a method called "coin tossing method" to evaluate the capacity of the burst-error channel using Equation 5.13 and the property of convergence in probability according to

$$\frac{1}{l}\log p(\mathbf{z}_l) \mapsto \lim_{k\to\infty} \frac{1}{k} H(\mathbf{z}_k). \tag{5.21}$$

There the term $\lim_{k\to\infty} \frac{1}{k} H(\mathbf{z}_k)$ is approximated by generating a long process of sequence $\mathbf{z}_l$ and evaluating $\frac{1}{l}\log p(\mathbf{z}_l)$. The sequence $\mathbf{z}_l$ can be generated recursively as a Bernoulli$(q_i)$ process using Equation 5.17. Therefore the capacity can

be computed as

$$\begin{aligned} C &\triangleq 1 + \frac{1}{l} \log p(\mathbf{z}_l) \\ &= 1 + \frac{1}{l} \sum_{i=1}^{l} \log p(z_i \mid \mathbf{z}_{i-1}) \\ &= 1 + \frac{1}{l} \sum_{i=1}^{l} [z_i \log(q_i) + \bar{z}_i \log(1 - q_i)], \end{aligned} \tag{5.22}$$

where $\bar{z} = 1 - z$ and it was shown in [63] to converge to the same value as the computationally demanding method of evaluating the probability distribution in each iteration.

In Figure 5.3 the capacity of the Gilbert-Elliot channel computed by the coin tossing method with channel parameters $\Delta_B = 10$, $p_G = 10^{-20}$, $p_B = 0.5$ and $p_E$ ranging from 0.001 to 0.5 is shown in comparison to the capacity of the BSC. The computation of coin tossing method is averaged over 100 realizations. It can be observed that channel capacity decreases with increasing input bit error probability $p_E$. The capacity of Gilbert-Elliot channel is higher than the capacity of BSC for the same input bit error probability.

## 5.3 Simulation Set Up of Staircase Codes

Due to the iterative decoding of the Staircase codes, it is hard to analytically find the output bit error probability on a burst-error channel. Therefore, we perform Monte Carlo simulations to investigate the performance of these codes on burst-error channels.

The G.709-compatible Staircase code from [6] is our baseline code. Its component codes are shortened binary BCH codes ($n = 1022, k = 990; t = 3$) extended by 2 bit CRCs[1], where $n$ is the block length, $k$ is the number of data bits and $t$ is the error correction capability. In one codeword block $B_i$ (see Figure 3.1) there

[1]The insertion of CRC bits to the Staircase codes is quite essential for small-$t$ RS or BCH component codes (which have probability of decoding error of at most $1/t!$ if more than $t$ errors occur [51]), because the CRC prevents the component codes to contribute to additional events that increase the error probability of product-like codes [41], which is the cause of high error floors.

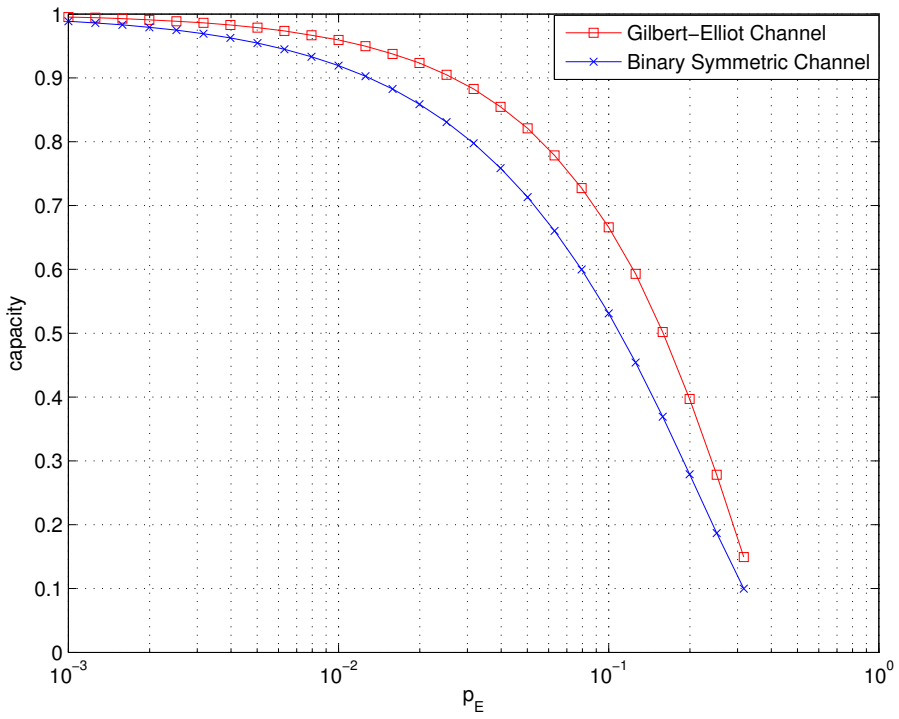



Figure 5.3: The capacity of the Gilbert-Elliot channel with parameter $\Delta_B = 10$, $p_G = 10^{-20}$, $p_B = 0.5$ compared to the capacity of the BSC channel.

are 512 rows and 510 columns, which corresponds to 261120 bits. The code has overall rate $R = 0.9373$, which indicates high-rate transmission. Due to the fixed structure of Staircase codes, it is difficult to leverage the same block length and rate for different component codes, when RS codes are to be used. The best we can do is to design a code with approximately the same rate. However, the block lengths are smaller and bigger than those of the baseline Staircase code, such that the performance of the baseline Staircase code is expected to be in-between.

The RS component codes built into the Staircase structure yield high rates approximately equal to that of the baseline code as follows:
1) RS $(n = 255, k = 247; t = 4)$ in GF$(2^8)$, $R \approx 0.9370$, each $B_i$ has 128 rows and 127 columns of symbols resulting in 130048 bits.
2) RS $(n = 255, k = 247; t = 4)$ in GF$(2^8)$ extended by 2 bit CRCs with $R \approx$

0.9352, each $B_i$ has 128 rows of symbols and 127 symbols plus 2 bits columns resulting in 130304 bits. The 2 bit CRCs are implemented by multiplying the polynomial $X^2 + 1$ to each row of the encoded RS component codes.
3) RS $(n = 511, k = 495; t = 8)$ in GF$(2^9)$, $R \approx 0.9373$, each $B_i$ has 256 rows and 255 columns of symbols resulting in 587520 bits.
For all simulations we decode with a sliding window of size 7 and the maximum number of iterations is set to 7.

## 5.4 Staircase Codes with Block Interleaving

When a code is transmitted on a channel with burst errors, one method to reduce the effect of burst errors is to deploy an interleaver; such that bundles of errors are distributed over many codewords.

Row-column interleaving is a well-known simple type of block interleaving. The input codeword is firstly arranged into the block interleaver row-by-row. Then the output is read from the block column-by-column. At the de-interleaver the received codeword is arranged into the block column-by-column and then the output is read from the block row-by-row. At the end the output symbols have the same sequence as the input symbols and can be decoded with the deployed channel decoder.

The diagonal interleaving for product codes proposed by [86] interleaves the code bits or symbols of a block of a product code in diagonal direction. The pseudo code for arranging the product code symbols with $n_2$ rows and $n_1$ columns assuming $n_2 \geq n_1$ is given in Algorithm 1, where $i$ is the index of the $i$th row, and $j$ is the index of the $j$th column of the product code array, whose element is read out as the $v$th element. An example of diagonal interleaving sequence of a product code array with 6 rows and 4 columns is illustrated in Figure 5.4, where the number in each box corresponds to the read out sequence $v$. The red numbers indicate that the read out sequence is done in diagonal direction.

After each block of a Staircase codeword is encoded, it is interleaved with an array of dimension equal to the dimension of one Staircase codeword block; thus the size of an interleaving block is defined. The symbols of BCH component codes are in bits as they are from GF(2); therefore the baseline Staircase code is in bit-

**Algorithm 1** Diagonal Interleaver [86]

$i \leftarrow 0$, $j \leftarrow 0$, $f \leftarrow 0$, $g \leftarrow 0$
**for** $v = 1$ to $n_1 n_2$ **do**
  $i \leftarrow ((v - 1 + f \cdot g) \mod n_2) + 1$
  $j \leftarrow ((v - 1) \mod n_1) + 1$
  **if** $ij = n_1 n_2$ **then**
    $f \leftarrow 1$
  **end if**
  **if** $f = 1$ **and** $j = n_1$ **and** $v_2 \mod n_2 = 0$ **then**
    $g \leftarrow g + 1$
  **end if**
**end for**

| | | | |
|---|---|---|---|
| 1 | 18 | 7 | 24 |
| 13 | 2 | 19 | 8 |
| 9 | 14 | 3 | 20 |
| 21 | 10 | 15 | 4 |
| 5 | 22 | 11 | 16 |
| 17 | 6 | 23 | 12 |

Figure 5.4: Diagonal interleaving sequence of a product code array with 6 rows and 4 columns.

wise interleaved. Symbol-wise interleaving performs interleaving for each symbol of RS Staircase codes such that the burst error correction capability of the RS component codes is maintained. At the receiver, each block is de-interleaved and then decoded with an iterative Staircase-code decoder as illustrated by Figure 5.5. The drawback is that extra memory is required and interleaving causes additional delay due to computing time for rearranging the symbols [61].

For the baseline Staircase code, we simulated with bit-wise row-column interleaving and diagonal interleaving, both with an interleaving array of 512 rows and 510 columns with a size of 261120 bits equal to one codeword block $B_i$.

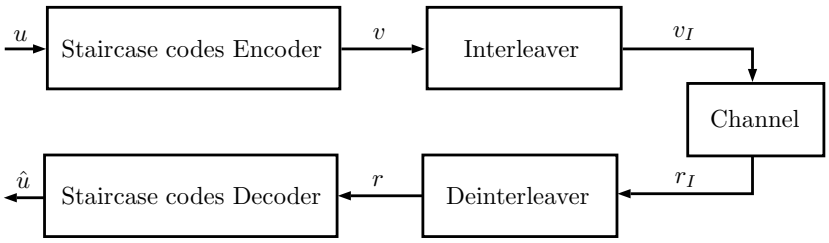



Figure 5.5: Diagram of Staircase codes with interleaving.

For the RS Staircase code, the symbol-wise diagonal interleaving for every 2 blocks of the Staircase RS code ($n = 255, k = 247; t = 4$) with and without CRC component codes is performed. The interleaving array has 128 rows and 254 columns for the RS ($n = 255, k = 247; t = 4$) Staircase code without CRC, which results in an interleaving block size of 260096 bits. The interleaving array for the 2 blocks of RS ($n = 255, k = 247; t = 4$) Staircase code with CRC has 128 rows and 256 columns with an interleaving block size of 260608 bits, where the 2 bit CRCs are filled with zeros before transformed into symbols. Accordingly, the interleaving block sizes in bits are close to the interleaving block size of the BCH baseline code.

## 5.5 Simulation Results for High-Rate Staircase Codes on a Burst-Error Channels

Figure 5.6 shows the performance of those high-rate Staircase codes on a *random-error* channel. The baseline Staircase code, the baseline Staircase code with diagonal interleaving, and the baseline Staircase code with row-column interleaving have the same performance, because the channel errors have a random distribution, hence the interleaving does not improve anything. It can be seen that the Staircase codes with RS component codes have a worse performance than the baseline Staircase code on the random-error channel due to symbol-wise encoding of the RS component codes such that they are not suitable for correction of random errors as many symbols are corrupted with single erroneous bits. High error floor can be noticed for the RS $(255, 247; 4)$ Staircase code without

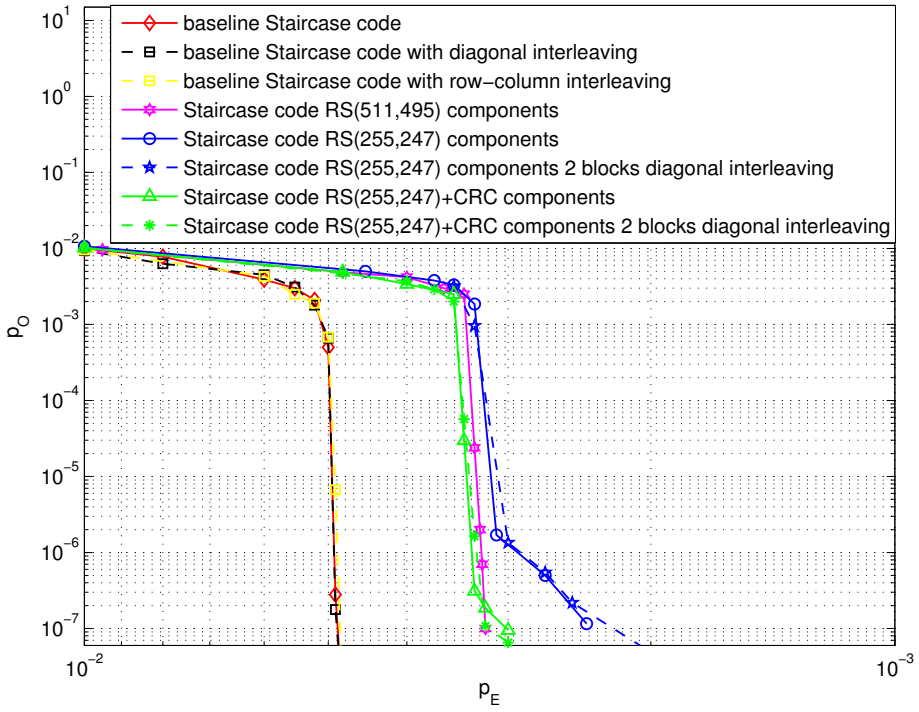



Figure 5.6: Performance of Staircase codes with different component codes on a random-error channel.

CRC due to small error correction capability $t$ of the component codes, while the RS $(255, 247; 4)$ Staircase code with 2 bit CRCs has a lower error floor and has a little bit higher iterative decoding threshold. The diagonal interleaving on both RS $(255, 247; 4)$ Staircase code with and without CRC brings no improvement again, because the random errors are "randomized" which does not change the error characteristics. The RS $(511, 495; 8)$ Staircase code has a little bit higher iterative decoding threshold than the RS $(255, 247; 4)$ Staircase code without CRC, but still lower than the RS $(255, 247; 4)$ Staircase code with CRC; however no error floor can be observed at $p_O = 10^{-7}$ of our simulation.

Figure 5.7 shows the performance of these high-rate codes on a *burst-error* channel with average burst length of 10. It can be observed that the baseline Staircase code without interleaving and the baseline Staircase code with row-

column interleaving have similar and the worst performance of all curves. The baseline Staircase code with diagonal interleaving has significantly better performance than that with row-column interleaving. Although both interleaving methods spread the burst errors over multiple component codewords, the row-column interleaving with the array of size equal to that of the Staircase code block does not improve the burst-error correction capability, because it spreads the errors over rows or columns, which the iterative decoding of Staircase codes has achieved to correct. The RS Staircase codes have better performance than on the random-error channel in Figure 5.6 and also have much better performance than the three curves of the BCH Staircase codes on burst-error channels, because the RS component codes are encoded and decoded symbol-wise, thus they are suitable for burst-error channels. All RS Staircase codes perform much better than the baseline Staircase code to correct burst errors at higher input bit error probability $p_E$; however the Staircase codes with RS $(255, 247; 4)$ component codes show high error floor, so that the curves intersect with the baseline Staircase code with diagonal interleaving at low measured bit error rate $p_O$. Thus, diagonal interleaving for the baseline Staircase code is better for burst error channels with short burst lengths. The RS $(255, 247; 4)$ Staircase code without CRC shows higher error floor and worse performance than the RS $(255, 247; 4)$ Staircase code with CRC. The symbol-wise diagonal interleaving improves the performance of both RS $(255, 247; 4)$ Staircase codes. The RS $(511, 495; 8)$ Staircase code has a better performance than the RS $(255, 247; 4)$ Staircase code because the RS component codes are longer and have higher error correction capability $t$. The performance of RS $(511, 495; 8)$ Staircase code with symbol-wise diagonal interleaving is expected to be better than that without symbol-wise diagonal interleaving; however, we omit its simulation, since the simulations of diagonal interleaving of the RS $(255, 247; 4)$ Staircase codes have indicated the results of using diagonal interleaving on RS Staircase codes.

We simulate these high-rate Staircase codes for different average error-burst lengths $\Delta_B$ on the channel and for error probabilities $p_E = 0.0030$ and $0.0045$, for which the simulation of our baseline Staircase code finds no errors in $10^9$ data bit realizations on a random-error channel. In our graphs the marks[1] at $p_O = 10^{-8}$

[1] According to [44, Theorem 2.4] the confidence interval for a probability $p$ of "success" (a

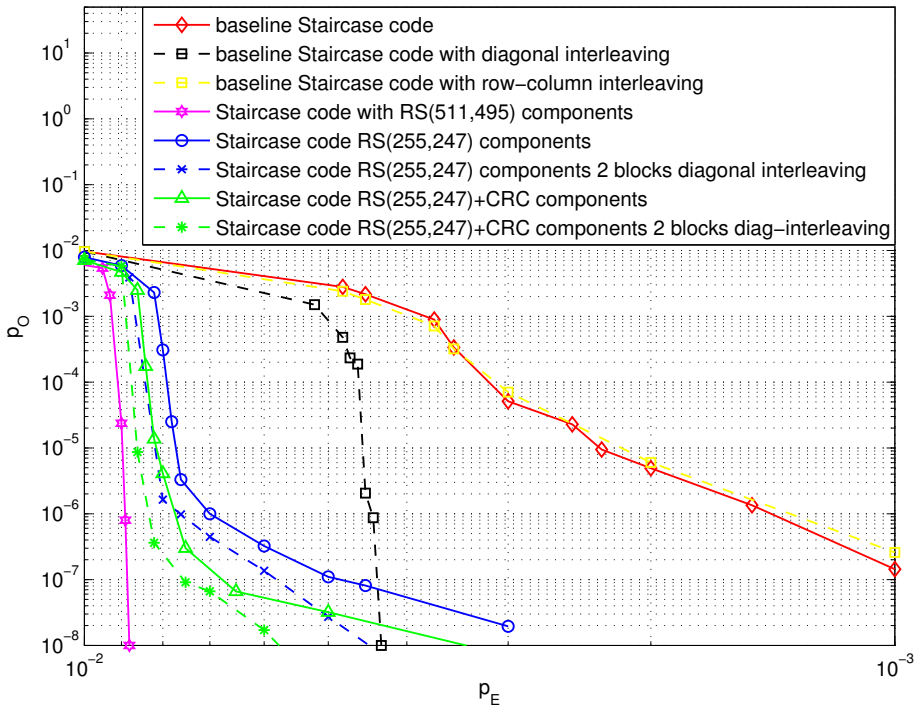



Figure 5.7: Performance of Staircase codes with different component codes on a burst-error channel with average burst length of 10.

indicate that no error has been found in $10^9$ realizations of our simulation.

Figure 5.8 shows the performances of different Staircase codes for different average burst length at $p_E = 0.0030$. We can observe that the baseline Staircase code shows no error only up to $\Delta_B = 2$, while the baseline Staircase code with diagonal interleaving shows no error up to $\Delta_B = 100$. The RS $(255, 247; 4)$ Staircase code without CRC component codes has an output bit error proba-

bit error in our case) when we observe $z = 0$ successes in an experiment of $n$ independent trials is $[0; p_0(n)]$. For $n$ large and for a confidence of 0.95 the upper limit $p_0(n)$ is approximately given by $p_0(n) \approx \frac{3.689}{n}$, with the relative error smaller than 10% for $n \geq 20$. A condition for this result to hold is, however, that the trials must be independent, and, as we are considering bit-error probability also within code words, independence can not be guaranteed. Therefore, well knowing that the error probability is not zero, we have decided to mark the fact that no error has been found in $10^9$ trials by inserting a point at $10^{-8}$ in Figures 5.8, 5.9, which is more conservative than the bound $p_0(n) \approx 3.689 \cdot 10^{-9}$ one would obtain from the theorem [44].

bility of $p_O = 10^{-6}$ on a random-error channel, which is higher than for the RS $(255, 247; 4)$ Staircase code with CRC component codes, which has $p_O = 10^{-7}$. For $2 \leq \Delta_B \leq 12$ we find no error for the RS $(255, 247; 4)$ Staircase code with CRC, while for the RS $(255, 247; 4)$ Staircase code without CRC we still find some errors. The symbol-wise diagonal interleaving on both RS $(255, 247; 4)$ Staircase codes improves the performance such that no error is found from $\Delta_B = 5$ in the case that the code is without CRC, and from $\Delta_B = 2$ in the case that the code is with CRC, up to $\Delta_B = 400$. For the RS $(510, 495; 8)$ Staircase code we find no error both on a random-error channel and on a burst-error channel up to $\Delta_B = 100$. The output bit error probability $p_O$ of RS $(255, 247; 4)$ Staircase codes at first decreases with the burst length $\Delta_B$ (at fixed bit error probability $p_E$) and later increases again, while the output bit error probability $p_O$ of RS $(511, 495; 8)$ Staircase codes is small even for small burst length and it only starts to increase at relatively large $\Delta_B$.

Figure 5.9 shows the performances of different Staircase codes for different average error-burst length at $p_E = 0.0045$, which is slightly below the iterative decoding threshold of the baseline Staircase code. The baseline Staircase code shows no error up to $\Delta_B = 2$. The diagonal interleaving improves the performance such that no error is found up to $\Delta_B = 7$. All of the RS Staircase codes cannot correct errors on the random-error channel at this input bit error probability, where the output bit error probability is equal to input bit error probability. The RS $(255, 247; 4)$ Staircase code without CRC component codes can correct with $p_O < 10^{-6}$ for $3 < \Delta_B < 15$, while the RS $(255, 247; 4)$ Staircase code with CRC component codes shows a better performance with $p_O < 10^{-7}$ in the same $\Delta_B$ interval. The symbol-wise interleaving also helps to extend the error correction to around $\Delta_B = 200$ with $p_O$ close to $10^{-8}$ for both RS $(255, 247; 4)$ Staircase codes. The RS $(510, 495; 8)$ Staircase code shows no error from $\Delta_B = 2$ up to $\Delta_B = 60$. The symbol-wise diagonal interleaving of RS $(511, 495; 8)$ Staircase code is expected to extend the burst-error correction capability to a longer burst length than $\Delta_B = 60$; however, we omit its simulation, since the simulations of symbol-wise diagonal interleaving of the RS $(255, 247; 4)$ Staircase codes have indicated that using diagonal interleaving on RS Staircase codes can extend the burst-error correction capability to a longer burst length.

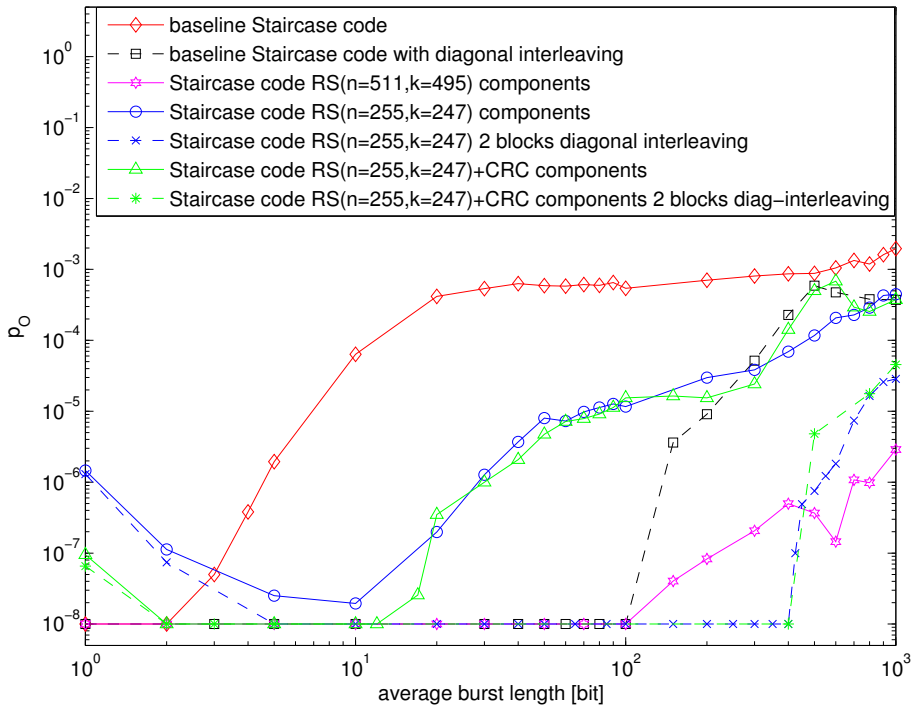



Figure 5.8: Performance of Staircase codes with RS and BCH component codes vs. average error-burst length $\Delta_B$ for a bit error probability of $p_E = 0.0030$.

## 5.6 Complexity Comparison of the Component Codes

The analysis in [6] discovered that Staircase codes have smaller decoding data flow than message passing decoding for LDPC codes, because the decoding of component codes using lookup-table-based decoding has relative low data flow. Due to the structure of Staircase codes, which enables parallel decoding of many component codes in one decoding window at the same time, we compare the software-decoding complexities of the component codes, which mainly affect the decoding latency of Staircase codes. The decoding of high-rate codes using syndromes is efficient and has small complexity [17]; therefore we analyze syndrome-

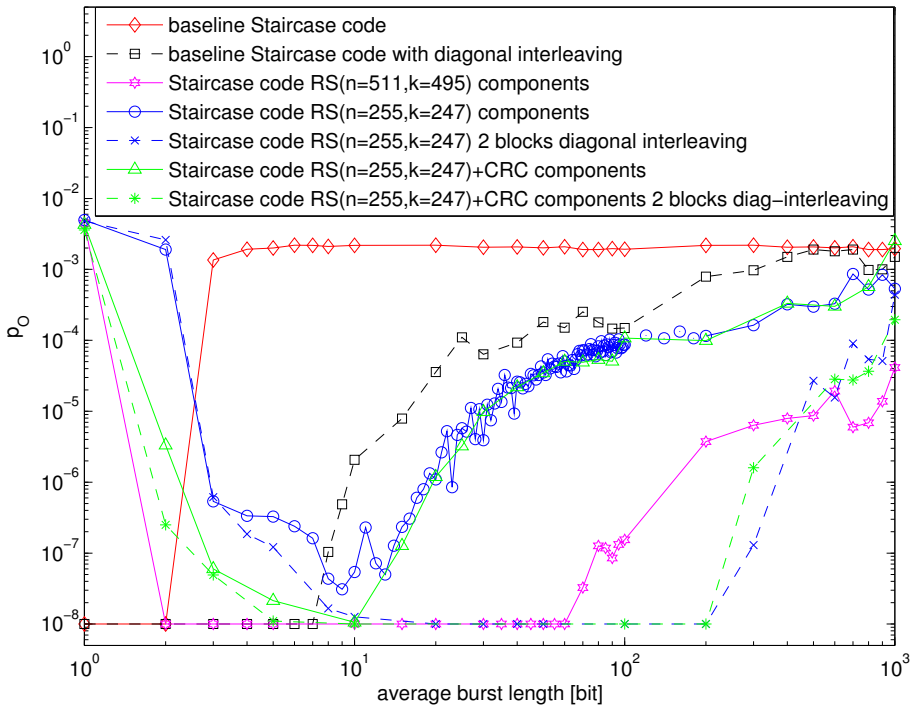



Figure 5.9: Performance of Staircase codes with RS and BCH component codes vs. average error-burst length $\Delta_B$ for a bit error probability of $p_E = 0.0045$.

based decoding here.[1]

The implementation complexities for decoding RS codes are given in Table 5.1 (with references). For BCH codes, which have symbols in GF(2), the complexity of computing syndromes reduces to half of the computation than for RS codes, as well as the complexity of finding the error-location polynomial using the Berlekamp-Massey algorithm, due to the fact that for computing a syndrome we can exploit the property $[f(X)]^{2i} = f(X^{2i})$ for any polynomial in $X$ in GF(2) (for details, see [46]). Furthermore there is no need for Forney's formula for codes over GF(2) as knowledge of the error-location is sufficient to correct by flipping the bit. The complexities of decoding BCH codes are given in Table 5.2. The

[1]We would like to thank Prof. Martin Bossert for help with the computation of the complexities of hard decision decoding of BCH and RS codes.

Table 5.1: Complexity of RS Decoding

| Decoding step | Multiplication | Addition | Inversion |
|---|---|---|---|
| Syndrome Comp. [17] | $2t(n-1)$ | $2t(n-1)$ | 0 |
| Massey Algorithm [46] | $4t^2$ | $4t^2$ | 0 |
| Chien Search [17] | $n(t-1)$ | $nt$ | 0 |
| Forney's Formula [17] | $2t^2$ | $t(2t-1)$ | $t$ |
| Total | $3nt+6t^2-n-2t$ | $3nt+6t^2-3t$ | $t$ |
| Total per bit | $\frac{3nt+6t^2-n-2t}{qn}$ | $\frac{3nt+6t^2-3t}{qn}$ | $\frac{t}{qn}$ |

$n$: code length in symbols, $q$: bits per symbol, $t$: symbol error correction capability

Table 5.2: Complexity of BCH Decoding

| Decoding step | Multiplication | Addition | Inversion |
|---|---|---|---|
| Syndrome Comp. [46] | $t(n-1)$ | $t(n-1)$ | 0 |
| Massey Algorithm[46] | $2t^2$ | $2t^2$ | 0 |
| Chien Search[17] | $n(t-1)$ | $nt$ | 0 |
| Total | $2nt+2t^2-n-t$ | $2nt+2t^2-t$ | 0 |
| Total per bit | $\frac{2nt+2t^2-n-t}{n}$ | $\frac{2nt+2t^2-t}{n}$ | 0 |

$n$: code length in bits, $t$: bit error correction capability

decoding complexity of the additional $x$ bits CRC of a codeword of length $n$, which can be implemented using the shift register from Figure 2.9, requires $xn$ additions and $(x-1)n$ multiplications. Table 5.3 shows numbers for the decoding complexities per component codeword of the Staircase codes with code rate of approximately 0.93.

Even though the complexities of decoding all codes are of the same order and directly proportional to $n$ for the high-rate regime where $n \gg t$, the number of arithmetic operations per code bit are different as shown in Table 5.4 for specific examples. We observe that the BCH $(1023, 993; 3)$ code requires the most multiplication and addition operations, while the RS $(255, 247; 4)$ code re-

Table 5.3: Complexities per code word for decoding the component codes

| Component codes | Multiplication | Addition | Inversion |
|---|---|---|---|
| BCH (1023,993) | 5130 | 6153 | 0 |
| RS (255,247) | 2893 | 3144 | 4 |
| RS (511,495) | 12121 | 12624 | 8 |

Table 5.4: Comparison of the complexities per code bit

| Component codes | Multiplication | Addition | Inversion |
|---|---|---|---|
| BCH (1023,993) | 5.0147 | 6.0147 | 0 |
| RS (255,247) | 1.4181 | 1.5412 | 0.0020 |
| RS (511,495) | 2.6356 | 2.7449 | 0.0017 |

quires the most inversion operations per code bit. Nevertheless, the complexity of each arithmetic function depends on the algorithm used, and the cost of the arithmetic functions are different for symbols from different fields. We therefore measure the software decoding time per code bit of those component codes operated on our computer as given in Table 5.5. Furthermore we measure the additional latency per code bit of a diagonal de-interleaving function with respect to no interleaving as shown in Table 5.6. We note that for other hardware the values will vary, but generally RS decoding latency will also remain smaller than that of BCH $(1023, 993; 3)$ component codes on other configurations. The software decoding latency for one sliding window of a Staircase code is equal to the number of component codes in one block multiplied by the decoding time of one component codeword, since the decoding is usually serially operated. However in hardware simulations e.g., by an FPGA realisation, the decoding of one sliding window can be done in parallel, such that the decoding latency depends only on the decoding time of one component codeword; moreover, the additional latency due to transformation between bit and symbols (for RS Staircase codes) can be neglected, as for an FPGA circuit, the line for symbols can be preallocated under construction.

The Staircase code with RS $(511, 495; 8)$ component codes has high performance in decoding long burst errors at the given $p_E$, and the time required for decoding the component codes per code bit is 4 times smaller than the decoding time required for decoding the BCH $(1023, 993; 3)$ component codes of the baseline Staircase code. The Staircase code with RS $(255, 247; 4)$ can correct short burst errors at the given $p_E$ and the decoding time of the component codes per code bit is 5 times smaller than the decoding time of the BCH $(1023, 993; 3)$. The de-interleaving time per code bit for decoding 2 blocks of RS $(255, 247; 4)$ Staircase code is 8.8 times less than that for one block of the baseline Staircase

Table 5.5: Software decoding time per code bit [$\mu s$]

| BCH (1023,993) | RS(255,247) | RS(511,495) |
|---|---|---|
| 0.3093 | 0.0586 | 0.0739 |

on Dell PowerEdge Rack 410 Dual Xeon X5650 2.66GHz (2011)

Table 5.6: De-interleaving time per code bit [$\mu s$]

| 1 block baseline Staircase code | 2 blocks Staircase RS(255,247) code |
|---|---|
| 0.1624 | 0.0184 |

on Dell PowerEdge Rack 410 Dual Xeon X5650 2.66GHz (2011)

code, in spite of approximately equal bit block interleaving sizes, because the interleaving is done symbol-wise.

## 5.7 Conclusion

We investigated high-rate Staircase codes for transmission over burst-error channels. The Staircase codes with RS component codes have much better performance in correcting burst errors than the baseline Staircase code with BCH component codes. For Staircase codes with the same overall rate, the BCH component codes of the baseline Staircase code require the largest decoding time per code bit, while the selected RS component codes require less decoding time per bit; therefore the decoding time of the baseline Staircase code is higher than the decoding time of the selected RS Staircase codes for the same number of bits. The larger the error correction capability $t$ of RS component codes, the longer burst errors they can correct; however this increases decoding latency and the complexity per code bit. Besides, the complexity per bit is inversely proportional to the symbol size $q$: the number of multiplications per bit is proportional to $\frac{3t-1}{q}$, and the number of additions per bit is proportional to $\frac{3t}{q}$ for $n \gg t$.

The baseline Staircase code with diagonal interleaving, despite additional latency, shows satisfactory performance on burst-error channels with short burst lengths at the given input bit error probability, in contrast to the well-known row-column interleaving, which does not bring about the improvement, because the decoding of Staircase codes is processed iteratively row- and column-wise, thus

the burst-error distribution stays widely the same. The symbol-wise diagonal interleaving of RS $(255, 247; 4)$ Staircase code extends the burst-error correction capability to longer burst lengths. The additional latency in de-interleaving of the RS Staircase codes is lower than that of the BCH Staircase codes due to symbol-wise interleaving.

Therefore, Staircase codes with RS component codes are suitable for high-rate transmission on burst-error channels without increasing decoding latency. The symbol-wise interleaving of the RS Staircase codes extends the burst-error correction capability to longer burst length with inherently increasing decoding latency. However if there are both random errors and (not too long) burst errors in the channel, the baseline Staircase code with diagonal interleaving is a better solution as it can correct burst errors and additionally random errors without error floor (at the price of more decoding latency). All in all, a compromise between the burst-error correction capability and the decoding latency must be reached, depending on the burst-nature of the channel.

# Chapter 6

# Rate Compatible Staircase Codes for High-Rate Wireless Transmission

[1] Wireless transmission normally suffers from varying channels, which arise from multipath propagation that can cause severe amplitude fading, or from dispersion that causes inter symbol interference. Due to high bit rate, even slow movements of objects change the channel on a time scale of many bit periods. To transmit data efficiently under varying channel conditions an adaptation of the channel code rate is crucial. By channel estimation at the receiver, the measured channel SNR or the decoder input bit-error probability ($p_E$) [28] are used to determine the maximum code rate that can be supported. There are different algorithms for different error correction codes. For example LDPC codes [37], [42], [14], Convolutional Codes [31], RS codes [48], [19], and Turbo codes [27], [55], [87], Staircase codes with LDPC component codes [88] and GLDPC-Staircase codes [50] adapt the code rate at the transmitter, according to the channel quality (which must be returned from the receivers). The general method of adapting the code rate of a "mother code" is shortening or puncturing it; code specific details have to be accounted for, such as cycle length in LDPC codes [14], good convergence properties of the convolutional component codes of Turbo codes [55],

[1] Part of this Chapter has been accepted for publication in the Transactions on Emerging Telecommunications Technologies

large free distance $d_{free}$ and a small information error weight $c_d$ on all paths with $d \geq d_{free}$ for convolutional codes [31]. An important common point in designing rate-adaptive channel codes is to keep low complexity at the encoder and the decoder and rate-compatibility, meaning that lower-rate code words contain higher-weight code words.

One practical and effective approach to increase the throughput in time-varying wireless channels is to combine error correction codes with automatic repeat request (ARQ) schemes known as "hybrid ARQ": the decoder first tries to decode the received high-rate codeword. In case the decoder detects no failure it sends an acknowledgement (ACK). If, however, there *is* a decoding failure, a retransmission is required through a negative acknowledgement (NACK). There are two well known types of hybrid ARQ. For Type-I hybrid ARQ [21], upon receiving NACK the encoder re-encodes the data block and sends the codeword again, while the decoder attempts to decode the newly received codeword independently of the previously received one. This scheme merely increases the probability of successful transmission at the cost of lower throughput. The optimal approach is called Chase Combining (CC) [15], which combines multiple received codewords before decoding, corresponding to maximal ratio combining (MRC). For Type-II hybrid ARQ [68] the transmitter, upon receiving NACK, sends additional parity bits to the receiver (that have not been sent before), and the decoder tries to decode the received codeword using the word received from the previous transmission assisted by the newly received parity bits; this is called an "Incremental Redundancy (IR)" scheme. Type-II has significantly better error correction performance than Type-I for a given throughput, but also higher complexity and it requires additional signaling as well as larger buffer size at the encoder and the decoder.

Our contribution is the implementation of rate-adaptive Staircase codes suitable for wireless communication, which requires high-rate transmission and hard-decision decoding. The high performance high-rate Staircase codes [6] were previously of fixed rate for fiber optic communication. But the proposed extension with RS codes in an ARQ-framework rate-adaptive encoders are enabled so that the code can be successfully decoded in a wider range of bit error probability, and therefore throughput is increased.

## 6.1 Component Codes with Variable-Rate by Puncturing

We briefly discuss variable-rate block codes that can be used as component codes in the Staircase concept. Assume that we are given a $(N, K)_q$ block code in GF($2^q$) with generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}$ as a "mother code" with rate $R = K/N$. We also assume that the code is systematically encoded, i.e., the $K$ data symbols are mapped to $N$ code symbols by appending $N - K$ parity symbols.

Consider the popular and simple concept of puncturing [31]: the number $K$ of data symbols is kept constant (i.e., the same as in the mother code) and by puncturing, the number $N - K$ of redundancy bits is decreased, resulting in a higher code rate. Therefore, the mother code is of low code rate and the concept is that the rate-adaptive encoder sends punctured codewords through the channel, which corresponds to a code rate determined by the channel capacity. The puncturing pattern is known at the decoder, and the received punctured code word is treated as a code word of the mother code with erased symbols. We use this concept to adapt the rate of the RS assist part for rate-adaptive Staircase codes in the following section.

## 6.2 Rate-Adaption of Staircase Codes

Because RS codes allow for a flexible number of punctured bits for Incremental Redundancy (IR) transmission schemes [46], a wide range of rate-adaptivity is achieved with only one additional encoder and decoder; hence low complexity at the encoder and decoder is retained. Moreover, RS codes have burst-error correction capability to combat burst errors that can occur in wireless transmission. Therefore, we use punctured RS codes as extra component codes for the rate-adaptive Staircase codes we propose below. At the decoder, the punctured symbols of the RS component codes are treated as "erased" when decoding the component codes (see Section 3.2.2.3).

The RS codes are used additionally to the BCH codes of the standard Staircase

scheme, i.e., each Staircase block still consists of data bits and high-rate BCH component codes: the reason is to maintain high performance and low error floor in correcting random (non-burst) errors of the BCH Staircase code. The RS codes perform rate-adaptivity, and at the same time burst-error correction in addition to high-performance random error correction of BCH Staircase code without a requirement for an additional interleaving.

The code array of such a rate-adaptive Staircase code is shown in Figure 6.1. Each row of a Staircase block $B_i$ of length $m_1$ is additionally encoded with RS codes of low rate resulting in $m_2$ rows of RS codes, each of length $n$ and each with $m_1$ "data" bits. By variable-rate encoding (through puncturing of the parity symbols), the blocks of the RS component codes can be shortened to fit into the block size of the required rate; for this, the parity bits of the RS codes are partitioned into $m_2$ rows of blocks $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and so forth.

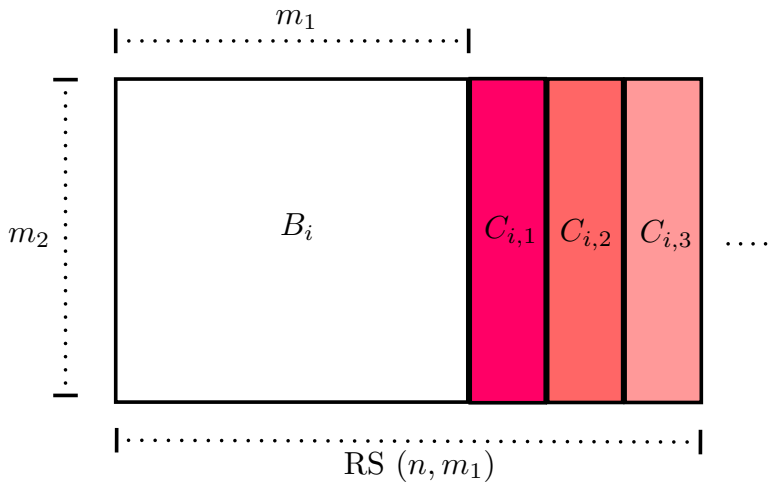



Figure 6.1: Rate-adaptive Staircase code: block array.

## 6.3 Rate-Adaption in Type-II hybrid ARQ

The rate-adaptive Staircase codes are deployed in a Type-II hybrid ARQ scheme with incremental redundancy, following the principle of rate-compatible punctured convolutional codes (RCPC) [31]: upon decoding failure, the transmission

of information or parity bits is *not* repeated; instead, previously punctured code bits are transmitted (forming a lower-rate code together with the information that has already been received), until the code is powerful enough to be decoded successfully. This way, a retransmission of the whole Staircase block is avoided. As depicted in Figure 6.2, the block $B_3$ (the same extension with RS blocks $C_{j,1}, C_{j,2}, C_{j,3}$... would apply to all other blocks $B_j$), only the block $C_{3,1}$ or additionally the block $C_{3,2}$ or additionally the block $C_{3,3}$ and so on are transmitted, when the transmitter keeps receiving the information "NACK" (unable to decode) from the decoder for block $B_3$ or $\{B_3, C_{3,1}\}$ or $\{B_3, C_{3,1}, C_{3,2}\}$ etc. In order to decide whether ACK or NACK is sent back to the encoder, the data bits are encoded with a Cyclic Redundancy Check (CRC). If no error is detected by the CRC, ACK is sent and NACK otherwise. The transmission of the ACK/NACK messages is assumed to be error-free. The required number of CRC bits depends on the desired reliability of the detection. $N_{\mathrm{CRC}} = 64$ redundancy bits will be sufficient for most applications. Note that the rate-loss relative to the data block size is rather small, as of the $m_1(m_2 - r) = 512(510 - 32) = 244736$ data bits in the standard Staircase code block just 64 are used for extra CRC redundancy.

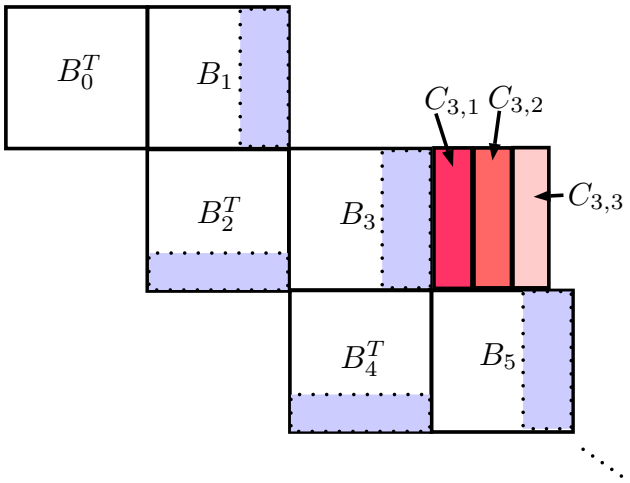



Figure 6.2: Rate-adaptive blocks: arrangement in the Staircase scheme, only shown for block $B_3$ for simplicity.

The flow-chart of rate-adaptive Staircase codes, deployed in a type-II hybrid ARQ scheme, is depicted in Figure 6.3. It has the following steps:

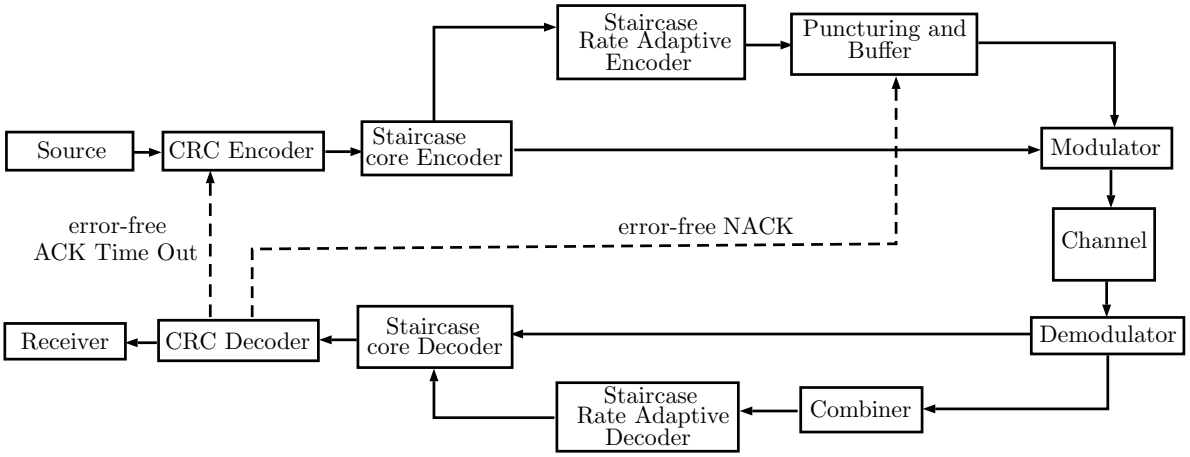



Figure 6.3: Block diagram of hybrid ARQ Staircase Codes.

1. The data bits in each Staircase block $B_1, B_2, B_3, ...$ are step-wise encoded with a Cyclic Redundancy Check (CRC) for error detection [46] to form ACK/NACK messages.

2. The blocks (including the CRC redundancy bits) from Step 1) are encoded with the high-rate Staircase code with BCH component codes extended by 2 bit CRCs[1]; this is the Staircase core code.

3. The core codewords from Step 2) are encoded with the rate-adaptive RS-encoder forming the blocks $C_{j,1}, C_{j,2}, C_{j,3}...$ that are kept in a buffer for the moment. If latency permits, one could also encode $C_{j,1}, C_{j,2}, C_{j,3}...$ only when those additional blocks are requested by the decoder (see later steps).

4. Transmit the blocks of the high-rate Staircase core code.

5. The Staircase core decoder attempts to decode the received word until the maximum number of iterations is reached. In the implemented version, the length of the decoding windows was $L = 7$; decoding would start at the last staircase block $B_{i+L-1}$ in the window and proceed back to the start of the

[1]To prevent error decoding of the small-$t$ component codes; thus high error floor is avoided

window at block $B_i$; from there, decoding would return in forward direction to block $B_{i+L-1}$ where the iteration started.

6. After core decoding in Step 5), the CRC error detection for the data symbols of block $B_i$ is evaluated. If errors are detected a NACK message is sent to the encoder and the scheme proceeds with Step 7).

   If the block $B_i$ is decoded error-free, this block is "decided" and removed from the decoding window. At the beginning of the decoding window the new block $B_{i=L}$ is accepted (in fact, the window slides one block forward) and the scheme proceeds with Step 1).

7. If for block $B_i$ a NACK-message was received, the transmitter sends the next block of incremental redundancy for each block in the decoding window (see also Figure 6.4 for an illustration that is further explained below). If there is no more redundancy available in the buffer, stop the process for block $B_i$, initiate failure management (e.g. complete re-transmission of $B_i$) and move the decoding window forward to decode the next block[1].

8. At the receiver, the incremental information is combined: the decoding iterations for the Staircase code are repeated, with RS decoding carried out first (to remove burst errors) for those blocks for which incremental redundancy is available; this is followed by standard BCH decoding in the sliding window fashion of the staircase scheme. After this decoding iteration the CRC of block $B_i$ is checked; if it fails again, go to Step 7). If the CRC detects no errors, block $B_i$ is decided, the window is moved forward, accepting the new block $B_{i+L}$, and the scheme proceeds with Step 1).

To further explain the process of rate-adaptive coding in the type-II hybrid ARQ scheme consider Figure 6.4. We assume that the current decoding window (dashed box) contains the blocks $B_3 ... B_9$. The check of the CRC of $B_3$ after the Staircase iterations detects errors in Figure 6.4 a), so incremental redundancy

[1] In the simulations below, this situation would be considered as packet loss which reduces throughput in Figure 6.14. For the bit-error performance simulation in Figure 6.5 we have considered this case by just using the erroneous data bits from the systematic encoding – this is the best we can do without assuming a re-transmission.

blocks are requested. This leads to correct decoding of block $B_3$ in part b) of Figure 6.4 and, hence, block $B_3$ is removed from the window and block $B_{10}$ is accepted in Figure 6.4 c). Note that the previously transmitted incremental redundancy packets are kept for $B_4 ... B_9$. Assuming that decoding of $B_4$ fails in the first Staircase iteration, more incremental redundancy blocks are requested (see Figure 6.4 d)) which finally leads to a correct decoding of $B_4$ which is removed from the window in Figure 6.4 e) and $B_5$ is decoded.

Of course one could evaluate all CRC within each decoding window and only request more incremental redundancy packets for those blocks that (still) fail to decode correctly. This would, however, require much more complicated ARQ signalling. Moreover, note that block $B_i$ (e.g. $B_3$ in Figure 6.4 a) is the "safest place" in the decoding window, as in the progress of sliding-window decoding, the most iterations have been spent on this block (within the current decoding window) and the most information from other blocks is concentrated there. Hence, when the decoding of block $B_i$ fails (indicated by the CRC) this is a clear message that the whole decoding window requires more redundancy to combat errors.

## 6.4 Rate-Adaptive Staircase Codes Analysis

### 6.4.1 Performance Analysis on Random-Error Channels

The performance analysis of rate-adaptive Staircase codes consists of 3 parts: the first is for high input bit error probability, the second part includes iterative decoding threshold analysis, and the third part is analysis for low bit error probability.

At high input bit error probability the Staircase code core parts alone cannot correct this high number of errors, because the number of errors is higher than the decoding threshold of iterated decoding, above which the decoded codeword is likely to fail. With the assistance by RS codes, the input bit probability of errors $p_E$ in each block is reduced to the acceptable intermediate bit probability of error $p_I$ for the iterative decoding of the core codes. Here in each row the shortened RS codes with error correction capability $t$ are decoded with the available parity bits using a bounded minimum distance decoder (BMD). Let $A(i)$ be the event

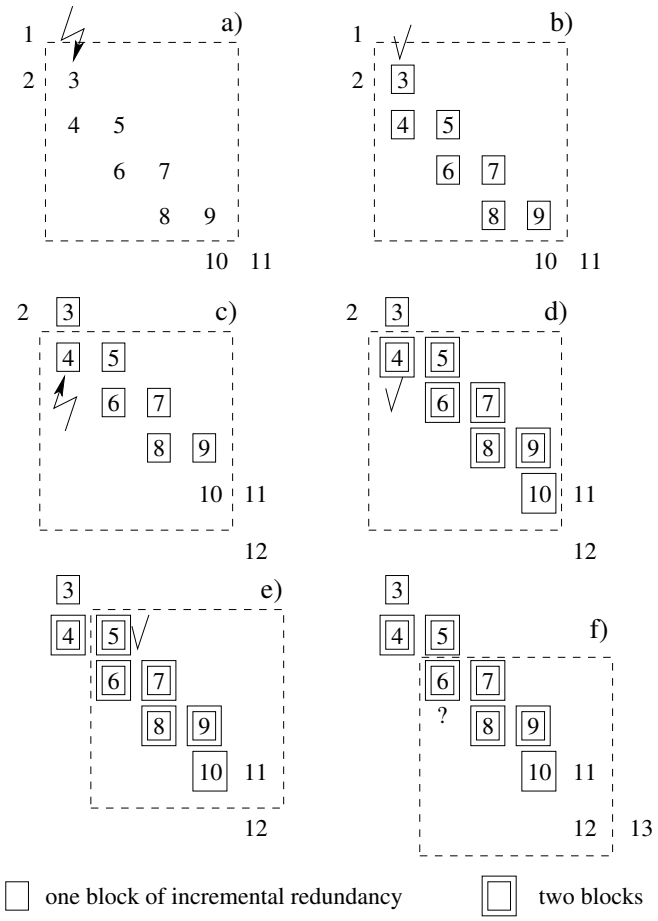



Figure 6.4: Illustration of Staircase decoding combined with incremental redundancy.

that $i$ symbol errors in a codeword of length $n$ occur at the decoder input, and its probability $P[A(i)]$ is given as [74]

$$P[A(i)] = \begin{pmatrix} n \\ i \end{pmatrix} P^i(1-P)^{(n-i)}, \tag{6.1}$$

where $P$ is the symbol error probability of the channel, $n$ is the codeword length in symbols. The term $\binom{n}{i}$ is the possibility of $i$ symbol errors occuring in $n$ symbols of a codeword, the term $P^i(1-P)^{(n-i)}$ is the error probability of a particular $i$-symbol error in a codeword of $n$ symbols. The BMD does not make a correction, when there are more than $t$ errors in a codeword; thus the average decoded symbol error probability is given by [74]

$$p_{SE} = \sum_{i=t+1}^{n} P[A(i)]\ p_{ISI}, \tag{6.2}$$

where $p_{ISI}$ is the average information-symbol error probability given $A(i)$. It is assumed in [74] that $p_{ISI} \approx p_{DSI}$ for $i = t+1, t+2, ..., n$, where $p_{DSI}$ is the average decoded-symbol error probability given $A(i)$, for $d \leq i \leq n$ the decoder does not change the number of errors so $p_{DSI} \approx i/n$, and for $t+1 \leq i \leq d$ the decoder produce a sequence of distance $d$ from the correct codeword so $p_{DSI} \approx d/n$; therefore, the average decoded symbol error probability is given by [74]

$$p_{SE} \approx \frac{d}{n} \sum_{i=t+1}^{d} \begin{pmatrix} n \\ i \end{pmatrix} P^i(1-P)^{(n-i)} + \frac{1}{n} \sum_{i=d+1}^{n} i \begin{pmatrix} n \\ i \end{pmatrix} P^i(1-P)^{(n-i)}. \tag{6.3}$$

Finally a tight lower bound on decoded symbol error probability over BSC is given by [60]

$$p_{SE} \geq \sum_{i=t+1}^{n} \frac{i}{n} \begin{pmatrix} n \\ i \end{pmatrix} P^i(1-P)^{(n-i)}\ , \tag{6.4}$$

where $P$ is the symbol error probability of the channel, $n$ is the codeword length in symbols (in our case the codewords are shortened to fit into one staircase block), and $t$ is the error correction capability (in symbols) of the RS assist codes.

In case of the transmission over a binary symmetric channel, the bit errors occur independently; thus the relation between the symbol error probability $P$ of the channel and the decoder input bit error probability $p_E$ is given as [60]

$$P = \sum_{i=1}^{m} \left( \begin{array}{c} m \\ i \end{array} \right) p_E^i (1-p_E)^{(m-i)}, \tag{6.5}$$

where $m$ is the length of binary sequence that represents one symbol in GF($q^m$). The term $\binom{m}{i}$ is for the number of possibility that $i$ bit errors occur in $m$ binary sequence that represent one symbol, while the term $p_E^i(1-p_E)^{(m-i)}$ is the error probability of a particular $i$-bit error in $m$ bits sequence. From the binomial theorem $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$, we have $a = 1-p_E$ and $b = p_E$; hence

$$\begin{aligned} P &= \sum_{i=1}^{m} \binom{m}{i} p_E^i (1-p_E)^{(m-i)} + \binom{m}{0} p_E^0 (1-p_E)^m - \binom{m}{0} p_E^0 (1-p_E)^m \\ &= (p_E + (1-p_E))^m - (1-p_E)^m \\ &= 1 - (1-p_E)^m. \end{aligned} \tag{6.6}$$

Since the symbol error probability $p_{SE}$ is related to the bit error probability $p_B$ by [60]

$$p_B = F p_{SE} \qquad \text{with} \qquad F = \frac{p_E}{P}, \tag{6.7}$$

scaling Equation 6.4 with $F$ and inserting $P$ with $1-(1-p_E)^m$ in the equation, we get the lower bound on intermediate bit error probability $p_I$ on the binary symmetric channel of the RS assist codes depending on the input bit error probability $p_E$ given as [60]

$$p_I \geq \frac{p_E}{1-(1-p_E)^m} \sum_{i=t+1}^{n} \frac{i}{n} \left( \begin{array}{c} n \\ i \end{array} \right) [1-(1-p_E)^m]^i [(1-p_E)^m]^{(n-i)}. \tag{6.8}$$

The intermediate bit error probability of the BCH component codes, which are

codes on GF(2), can be simply given as

$$p_I \geq \sum_{i=t+1}^{n} \frac{i}{n} \binom{n}{i} p_E^i (1-p_E)^{(n-i)} \ . \tag{6.9}$$

where the sum starts from $t+1$ errors that the BMD cannot correct, $\frac{i}{n}$ is for the number of errors relative to the number of codeword bits, $\binom{n}{i}$ is for the possibility of $i$ bit errors in $n$ codeword bits, and the term $p_E^i(1-p_E)^{(n-i)}$ is the error probability of a particular $i$-bit error in a codeword of length $n$ bits.

To find the iterative decoding threshold of the Staircase core code, which is a kind of product code, the error pattern in Staircase core code can be interpreted as an "error graph" [41] (see Section 2.8.1). Based on error graph statement, the peeling decoding analysis [75] can be deployed to find the iterative decoding threshold (see Section 4.2). The iterative decoding threshold can also be estimated using the density evolution (see Section 4.2) with the reduced bit error probability by RS assistance from Equation 6.8 as an input $x_i^{(0)}$ to the density recursion formulae [38] as follows

$$x_i^{(l+1)} = p\left(\frac{1}{w}\sum_{k=0}^{w-1} f_n\left(\frac{1}{w}\sum_{j=0}^{w-1} x_{i-j+k}^{(l)}\right)\right), \tag{6.10}$$

where $x_i^{(l)} = 0$ for all $i \notin \{1, 2, ..., L\}$, $L$ is the number of positions of bit nodes contained in a decoding window, $p$ is the reduced bit error probability after the RS assistance from Equation 6.8, $w$ is the number of groups that the sockets[1] at each bit/code-constraint position are separated, with $w = 2$ for Staircase core code, $l \to \infty$ is the number of iterations that $\bar{x}^{(l)}$ gets stuck at bisectrix, and $n$ is the codeword length of BCH component codes of the Staircase core code. The function $f_n(x)$ is defined as [38]

$$f_n(x) \triangleq \sum_{i=t}^{n-1} \frac{i}{n} \binom{n-1}{i} x^i (1-x)^{(n-i)}, \tag{6.11}$$

where $n$ is the codeword length of BCH component codes of the Staircase core

[1] output connections from bit nodes or from code-constraint nodes in the error graph

code, $t$ is the error correction capability of the BCH component codes of the Staircase core code.

After the intermediate error probability $p_I$ is less than the iterative decoding threshold, the tangent of the output bit-error curve is getting steep according to the product codes property, which corresponds to the waterfall region in the performance curve. This curve can be extrapolated to reach the error floor at very low output bit error probability, which depends on the probability that an error pattern contains a stall pattern[1] and was analysed in [6].

### 6.4.2 Throughput Analysis on Random-Error Channels

In wireless transmission, if a block cannot be decoded and a CRC detects errors, the packet is often dropped and a retransmission is required. The throughput is a figure of merit in this situation. The throughput of Rate-Adaption in Type-II hybrid ARQ Staircase codes under the assumption that all the blocks in a decoding window possess the same number of assist bits can be given as [31]

$$\eta = \frac{k}{k + l_{AV}}, \tag{6.12}$$

where $k$ is the number of information bits. The average number of parity bits $l_{AV}$, which is derived from [31], can be given as

$$l_{AV} = \begin{cases} \sum_{j=0}^{K} l_j (1 - P_{EF}(l_j)) \prod_{i=0}^{j-1} P_{EF}(l_i) & \text{if } \exists P_{EF}(l_j) < 10^{-7} \text{for } j \in \{0, 1.., K\} \\ \infty & \text{else,} \end{cases} \tag{6.13}$$

where $l_j$ is the number of parity bits used for decoding in step $j$, $l_0$ is the number of parity bits of the Staircase core codes, $K$ is the maximum number of assist steps. In our scheme, the packet is dropped after the maximum number of assist step is reached without success in decoding, which corresponds to when there

[1] "a set $s$ of codeword positions, for which every row and column involving positions in $s$ has at least $t+1$ position in $s$, and thus the decoder gets locked in $s$ state in which no updates are performed"[6], where $t$ is the error correction capability of the component codes. (details see Section 4.2)

is not any step $j$ that the packet error probability $P_{EF}(l_j)$ is lower than $10^{-7}$; therefore $l_{AV}$ is set to infinity to force the throughput to zero. The packet error probability $P_{EF}(l_j)$ at step $j$ is given by [31]

$$P_{EF}(l_j) = 1 - (1 - p_O(l_j))^n, \tag{6.14}$$

where $n$ is the packet length in bits, $p_O(l_j)$ is the output bit error probability of the rate-adaptive Staircase code from performance analysis depending on $l_j$. For a value of $l_j$, the output bit error probability $p_O(l_j)$ can be obtained from Equation 6.8; however when $p_I$ is lower than the iterative decoding threshold of the Staircase core code, $p_O(l_j)$ is set to a small value, e.g., $10^{-8}$ to account for the water fall region or the error floor.

### 6.4.3 Performance Analysis on Burst-Error Channels

In the analysis of the rate-adaptive Staircase codes on burst-error channels, where the Gilbert-Elliot model was used, we firstly consider the high input bit error probability region where the RS assist codes correct the errors in codewords. The probability of $m$ symbol errors in a sequence of $n$ symbols ($P(m,n)$), the probability of $m$ symbol errors in a block of length $n$ symbols where the channel is the "good state" at the first bit ($G(m,n)$), and the probability of $m$ symbol errors in a block of length $n$ symbols where the channel is the "bad state" at the first bit ($B(m,n)$), were first introduced in [24]. The recursion formulas for $P(m,n)$ are given as [24]

$$\begin{aligned}
P(m,n) &= \frac{r}{p+r}G(m,n) + \frac{p}{p+r}B(m,n), \\
G(m,n) &= G(m,n-1)(1-p)(1-g) + B(m,n-1)p(1-g) + \\
&\quad G(m-1,n-1)(1-p)g + B(m-1,n-1)pg, \\
B(m,n) &= B(m,n-1)(1-r)(1-b) + G(m,n-1)r(1-b) + \\
&\quad B(m-1,n-1)(1-r)b + G(m-1,n-1)rb,
\end{aligned} \tag{6.15}$$

with the initialisation $G(0,1) = 1-g$, $B(0,1) = 1-b$, $G(1,1) = g$, $B(1,1) = b$, and $G(m,n) = B(m,n) = 0$ when $m < 0$ or $m > n$. The parameters $r, p, b, g$ are

defined the same as in Section 5.1 as follows

$$p = P(Q_t = B \mid Q_{t-1} = G), \tag{6.16}$$

and

$$r = P(Q_t = G \mid Q_{t-1} = B). \tag{6.17}$$

At "good state (G)" errors are generated with probability $g$, while in "bad state (B)" errors are generated with probability $b$. These formulas are for the case of symbols from GF(2) where no state transition between good and bad state occurs in between one symbol transmission.

In case of RS codes, for which the state transitions between the good and bad state happen at times between one symbol transmission, the authors of [56] defined closed form expressions for this burst-error statistic. Firstly, the error sequence $E = \{E_k\}_{k=1}^{\infty}$, which is additive to the input sequence in GF(2) with $E_k = 1$ denoted as error at $k$th time, is defined. The matrices $\mathbf{P}(0)$ and $\mathbf{P}(1)$ have the $(i, j)$th entry as $P(E_k = e_k, S_k = j | S_{k-1} = i)$ which means the probability has the output symbol $e_k$ when the state changes from $i$ to $j$. Thus the sequence $\mathbf{e}_n \triangleq e_1 e_2 ... e_n$ has the probability [56]

$$P(\mathbf{e}_n) = \mathbf{\Pi}^T \left( \prod_{k=1}^{n} \mathbf{P}(e_k) \right) \mathbf{1}, \tag{6.18}$$

where $\mathbf{1}$ is a column vector with all entry ones, the distribution of the initial state $\mathbf{\Pi}$ of the channels assumed to be the stationary distribution given as the matrix

$$\mathbf{\Pi} = \begin{bmatrix} \pi_G \\ \pi_B \end{bmatrix} = \begin{bmatrix} \frac{r}{p+r} \\ \frac{p}{p+r} \end{bmatrix}, \tag{6.19}$$

where the parameters $p$ and $r$ are from Equation 6.16 and Equation 6.17 respectively. The transition matrices are defined as [56]

$$\mathbf{P} = \begin{bmatrix} 1-p & p \\ r & 1-r \end{bmatrix}, \tag{6.20}$$

$$\mathbf{P}(0) = \begin{bmatrix} (1-p)(1-g) & p(1-b) \\ r(1-g) & (1-r)(1-b) \end{bmatrix}, \tag{6.21}$$

$$\mathbf{P}(1) = \begin{bmatrix} (1-p)g & pb \\ rg & (1-r)b \end{bmatrix}, \tag{6.22}$$

where $g$ is the probability of generating errors at "good state (G)" and $b$ is the probability of generating errors at "bad state (B)". As an example, the probability of a sequence $\mathbf{e}_5 = 10010$ can be calculated by

$$P(10010) = \mathbf{\Pi}^T \mathbf{P}(1)\mathbf{P}(0)\mathbf{P}(0)\mathbf{P}(1)\mathbf{P}(0)\mathbf{1}. \tag{6.23}$$

Let $\mathcal{E}_n$ be an error event of length $n$, which is composed of all $w$ elementary error sequences $\mathbf{e}_n$ of length $n$. The generating series for such an error event is defined as [56]

$$F_{\mathcal{E}_n} = \sum_{i=1}^{w} x_{e_{i,1}} x_{e_{i,2}} ... x_{e_{i,n}}; \quad x_{e_{i,j}} \in \{x_0, x_1\}, j \in \{1, 2, ..., n\}, \tag{6.24}$$

which is in $R[[x_0, x_1]]$ the set of polynomials in non-commuting indeterminates $x_0$ and $x_1$, where $x_0$ marks an error bit equal to zero and $x_1$ marks an error bit equal to one. The probability of an elementary error sequence is not the same when the symbols are swapped; besides, it is related to the generating series, by which it can be obtained from the generating series. Firstly the generating series are used to determine the possible error sequences, and then $x_{e_i}$ of the generating series is replaced by $\mathbf{P}(e_i)$ and wrapped in $\mathbf{\Pi}$ and $\mathbf{1}$ as in Equation 6.18 to get the probability of the elementary error sequence. The probability of an error event of length $n$ can thus be written as [56]

$$P(\mathcal{E}_n) = \mathbf{\Pi}^T \left( \sum_{i=1}^{w} (\prod_{k=1}^{n} \mathbf{P}(e_{i,k})) \right) \mathbf{1}, \tag{6.25}$$

where $e_{i,k}$ is the $k$th symbol of the $i$th elementary error sequence. Using the

generating series property[1] for the error event with the set of all $\{0,1\}$ strings as follows [56]

$$\begin{aligned} F_{\mathcal{E}_n} &= [(\{0,1\}^*,\tau)] = (1-[(\{0,1\},\tau)])^{-1} = (1-x_0-x_1)^{-1} \in R[[x_0,x_1]] \\ &= [s^m z^n](1-x_0 z - x_1 sz)^{-1}, \end{aligned} \tag{6.26}$$

where the indeterminates $x_0$ marks the number of zeros, $x_1$ marks the number of ones, $z$ marks the length of the sequence, and $s$ marks the number of ones, then replacing $x_k$ with $\mathbf{P}(k)$, the probability of $m$ symbol errors in a codeword of length $n$ symbols is defined in terms of the generating series as [56]

$$\begin{aligned} P(m,n) &\triangleq P(\mathcal{E}_n) \\ &= [s^m z^n]\mathbf{\Pi}^T(\mathbf{I}-\mathbf{P}(0)z-\mathbf{P}(1)sz)^{-1}\mathbf{1}, \end{aligned} \tag{6.27}$$

where the indeterminate $z$ marks the length of the sequence, and the indeterminate $s$ marks the number of ones. For the RS code from GF($2^c$) the probability of $m$ error symbols in $n$ symbols is given as [56]

$$\begin{aligned} P(m,n) &= [s^m z^n]\mathbf{\Pi}^T(\mathbf{I}-\mathbf{P}(0)^c z-(\mathbf{P}^c-\mathbf{P}(0)^c)sz)^{-1}\mathbf{1} \\ &= [s^m z^n]H_P(s,z). \end{aligned} \tag{6.28}$$

The term $H_P(s,z)$ is the ratio of two polynomials in $s$ and $z$ known as *rational generating function*, which the denominator polynomial is for the recurrence relation, and the numerator polynomial is for the initial conditions. Thus $P(m,n)$ can be evaluated with a recurrence relation[2]. To evaluate the recurrence relation, we firstly give the coefficient of $z$ from Equation 6.27, 6.28 as a matrix $\mathbf{X1}$ with

[1] "$[(\mathcal{S}^*,w)] = 1+[(\mathcal{S},w)]+[(\mathcal{S}^2,w)]+... = (1-[(\mathcal{S},w)])^{-1}$, where $\mathcal{S}^* = \phi \cup \mathcal{S} \cup \mathcal{S}^2 \cup \mathcal{S}^3...$ is the set of sequences formed by concatenation any number of sequences in $\mathcal{S}$, the concatenation product of $\mathcal{A}$ and $\mathcal{B}$ is $\mathcal{AB} = \{ab | a \in \mathcal{A}, b \in \mathcal{B}\}$. $\mathcal{S}$ is a particular subset of the alphabet $\mathcal{N}_\mathcal{A} = \{0,1,..A\}$ to be enumerated, $w$ is the weight function recording the designed information about a sequence $\sigma \in \mathcal{S}$."[56]

[2] The recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_d a_{n-d}$ with generating function $a_0 + a_1 x^1 + a_2 x^2 + ....$ can be expressed by a rational generating function as: $a_n = a_0 + a_1 x^1 + a_2 x^2 + ... = \frac{b_0 + b_1 x^1 + b_2 x^2 + ... b_{d-1} x^{d-1}}{1 - c_1 x^1 - c_2 x^2 - ... - c_d x^d}$ [49], [5]

dummy elements $\mathbb{a}, \mathbb{b}, \mathbb{c}, \mathbb{d}$

$$\mathbf{X1} = \begin{bmatrix} \mathbb{a} & \mathbb{b} \\ \mathbb{c} & \mathbb{d} \end{bmatrix}, \tag{6.29}$$

and the coefficient of $sz$ from Equation 6.27, 6.28 as a matrix **X2** with dummy elements $\mathbb{e}, \mathbb{f}, \mathbb{g}, \mathbb{h}$

$$\mathbf{X2} = \begin{bmatrix} \mathbb{e} & \mathbb{f} \\ \mathbb{g} & \mathbb{h} \end{bmatrix}. \tag{6.30}$$

After some matrix operations, we get

$$H_P(s,z) = \frac{\begin{array}{r} \frac{r}{r+p}(1+(\mathbb{b}-\mathbb{d})z+(\mathbb{f}-\mathbb{h})sz) \\ +\frac{p}{r+p}(1+(\mathbb{c}-\mathbb{a})z+(\mathbb{g}-\mathbb{e})sz) \end{array}}{\begin{array}{r} 1-(\mathbb{a}+\mathbb{d})z-(\mathbb{e}+\mathbb{h})sz+(\mathbb{a}\mathbb{h}+\mathbb{d}\mathbb{e}-\mathbb{f}\mathbb{c}-\mathbb{b}\mathbb{g})sz^2 \\ +(\mathbb{a}\mathbb{d}-\mathbb{b}\mathbb{c})z^2+(\mathbb{e}\mathbb{h}-\mathbb{f}\mathbb{g})s^2z^2 \end{array}} \tag{6.31}$$

To find the initial conditions, let $H_P(s,z) = a_0 + a_1 z + a_2 sz + a_3 sz^2 + ...$, and multiply both sides of Equation 6.31 by its denominator, then equate the coefficients of each indeterminates $s$, $z$. As we will allow for a negative index in the recurrence relation, the condition $P(m,n) = 0$ for $m, n < 0$ must hold, and only $a_0, a_1$ and $a_2$ are required for the initial conditions [5] as $a_3, a_4, ...$ can be obtained from the recursion. Thus, we equate only the term with the indeterminates $1, z, sz$ as follows

$$\begin{aligned} a_0 \quad - \quad & (a_1 + (\mathbb{a}+\mathbb{d})a_0)z + (a_2 - (\mathbb{h}+\mathbb{e})a_0)sz = \\ & \frac{r}{r+p}(1+(\mathbb{b}-\mathbb{d})z+(\mathbb{f}-\mathbb{h})sz) + \frac{p}{r+p}(1+(\mathbb{c}-\mathbb{a})z+(\mathbb{g}-\mathbb{e})sz). \end{aligned} \tag{6.32}$$

Then we get the coefficients of the generating series that will be used as initial

conditions as follows:

$$\begin{aligned} a_0 &= 1 \\ a_1 &= \frac{r}{r+p}(\mathbb{a}+\mathbb{b}) + \frac{p}{r+p}(\mathbb{c}+\mathbb{d}) \\ a_2 &= \frac{r}{r+p}(\mathbb{e}+\mathbb{f}) + \frac{p}{r+p}(\mathbb{g}+\mathbb{h}). \end{aligned} \tag{6.33}$$

Finally the recurrence relation can be given as

$$\begin{aligned} P(m,n) &= (\mathbb{a}+\mathbb{d})P(m,n-1) + (\mathbb{e}+\mathbb{h})P(m-1,n-1) \\ &\quad -(\mathbb{a}\mathbb{h}+\mathbb{d}\mathbb{e}-\mathbb{f}\mathbb{c}-\mathbb{b}\mathbb{g})P(m-1,n-2) - (\mathbb{a}\mathbb{d}-\mathbb{b}\mathbb{c})P(m,n-2) \\ &\quad -(\mathbb{e}\mathbb{h}-\mathbb{f}\mathbb{g})P(m-2,n-2), \end{aligned} \tag{6.34}$$

with initial conditions

$$\begin{aligned} P(m,n) &= 0 \quad \text{for} \quad m,n<0 \;\; || \;\; m>n \\ P(0,0) &= a_0 = 1 \\ P(0,1) &= a_1 = \frac{r}{r+p}(\mathbb{a}+\mathbb{b}) + \frac{p}{r+p}(\mathbb{c}+\mathbb{d}) \\ P(1,1) &= a_2 = \frac{r}{r+p}(\mathbb{e}+\mathbb{f}) + \frac{p}{r+p}(\mathbb{g}+\mathbb{h}). \end{aligned} \tag{6.35}$$

When we insert the elements of **X1** with the elements of $\mathbf{P}(0)$, and insert the elements of **X2** with the elements of $\mathbf{P}(1)$ in Equation 6.34, 6.35 the result is a recurrence relation given in [56]. For RS codes, we insert the elements of **X1** with the elements of $\mathbf{P}(0)^c$ and the elements of **X2** with the elements of $\mathbf{P}^c - \mathbf{P}(0)^c$ in Equation 6.34, 6.35 to get the probability of $m$ symbol errors in a sequence of length $n$ symbols.

The output symbol error probability of a bound minimum distance decoder on burst-error channels is given as

$$p_{SE} = \sum_{m=t+1}^{n} \frac{m}{n} \cdot P(m,n), \tag{6.36}$$

where the summation starts from $t+1$ symbol errors that the BMD cannot correct,

$\frac{m}{n}$ is for the number of symbol errors relative to the codeword length $n$. Finally we scale the symbol error probability with $F = \frac{p_1}{P_1}$ according to Equation 6.7, where $p_1 = \Pi^T \mathbf{P}(1)\mathbf{1}$ accounts for probability of one bit error in the symbol, and $P_1 = 1 - \Pi^T \mathbf{P}(0)^c \mathbf{1}$ accounts for probability of symbol errors, therefore the intermediate bit error probability of the RS assist codes on burst-error channels can be given as

$$p_I = \frac{p_1}{P_1} \sum_{m=t+1}^{n} \frac{m}{n} \cdot P(m, n), \tag{6.37}$$

After the intermediate bit error probability $p_I$ is less than the iterative decoding threshold of the core BCH Staircase code on burst-error channels, the output bit error probability reduces due to the iterative decoding of the core BCH Staircase code; nevertheless it has not so steep slope as on random-error channels, because the core BCH Staircase code does not perform so well on burst-error channels.

## 6.5 Performance Simulation on Random-Error Channel

The G.709-compatible Staircase code from [6] is selected as the Staircase code core part because of its high performance in correcting random errors and its low error floor. To decide whether an ACK or NACK message is sent, we directly compare the data bits with the decoded bits in our simulation, thus in the simulation we do not have any failure in the CRC error detection; however in reality, a CRC is required for checking the correctness of the decoded bits as the data bits are not known to the decoder. Assuming a sufficient number of CRC redundancy bits, the probability not to detect any error is extremely small and, hence, we neglect this case. We select the RS code from GF($2^8$) as the assist codes, because it is from the smallest field that the number of bits in a row of one block of the core BCH Staircase code (510 bits) fits in the RS codeword such that the actual rate[1] of 0.5 is achievable (with symbols shortening).

[1] actual rate $= \frac{\text{the number of message bits in the core part}}{\text{the number of codeword bits in the core part+the number of the parity bits of the assist parts}}$

| $p_E$ | total rate | $p_O$ |
|---|---|---|
| 0.0078 | 0.8 | $1.39 \times 10^{-6}$ |
| 0.01315 | 0.7 | $2.58 \times 10^{-6}$ |
| 0.0185 | 0.6 | $3.195 \times 10^{-6}$ |
| 0.025 | 0.5 | $1.30 \times 10^{-6}$ |

Table 6.1: Performance of rate-adaptive Staircase codes assisted by RS codes on a random-error channel

For each row of the Staircase core part, the RS (n=255, k=199, shortened symbols = 135) component codes, which have $255 - 199 = 56$ parity symbols, $199 - 135 = 64$ symbols to be encoded, and an actual rate of 0.501, are encoded. These parity symbols are stored in a buffer for use in the ARQ scheme. If the transmitter receives NACK, it sends a block of parity symbols corresponding to the rate requirement, for example 11 parity symbols per row are sent for a rate requirement of 0.8, 22 parity symbols per row are sent for a rate requirement of 0.7, 36 parity symbols per row are sent for a rate requirement of 0.6, and all 56 parity symbols for each row are sent for a rate requirement of 0.5. At the receiver side, the decoder would succeed in decoding the rate-adaptive part using the Berlekamp-Massey Algorithm [46] with erasure symbols as long as for each row $2e + \rho \leq d_{min} - 1$, where $e$ is the number of error symbols, $\rho$ is the number of punctured symbols and $d_{min}$ is the minimum hamming distance of the RS mother code.

Figure 6.5 shows the performance of the rate-adaptive Staircase code for different input bit error probabilities on a *random-error* channel. With RS-assistance at an actual rate of 0.8, the Staircase code achieves an output bit error probability of $p_O = 1.39 \cdot 10^{-6}$ at an input bit error probability of $p_E = 0.0078$; this RS-assistance helps to increase the error correction capability of the Staircase code core part so that it is able to correct higher input error probability with the trade-off of a smaller rate. Table 6.1 gives an overview. We plot the analytical performance estimation derived from Section 6.4 for each rate of the RS assist codes in Figure 6.5. The estimated iterative decoding threshold of the core BCH codes obtained by peeling decoding analysis, which has a window size equal to 7, maximum number of iterations equal to 7, and averaging for 1000 decision blocks,

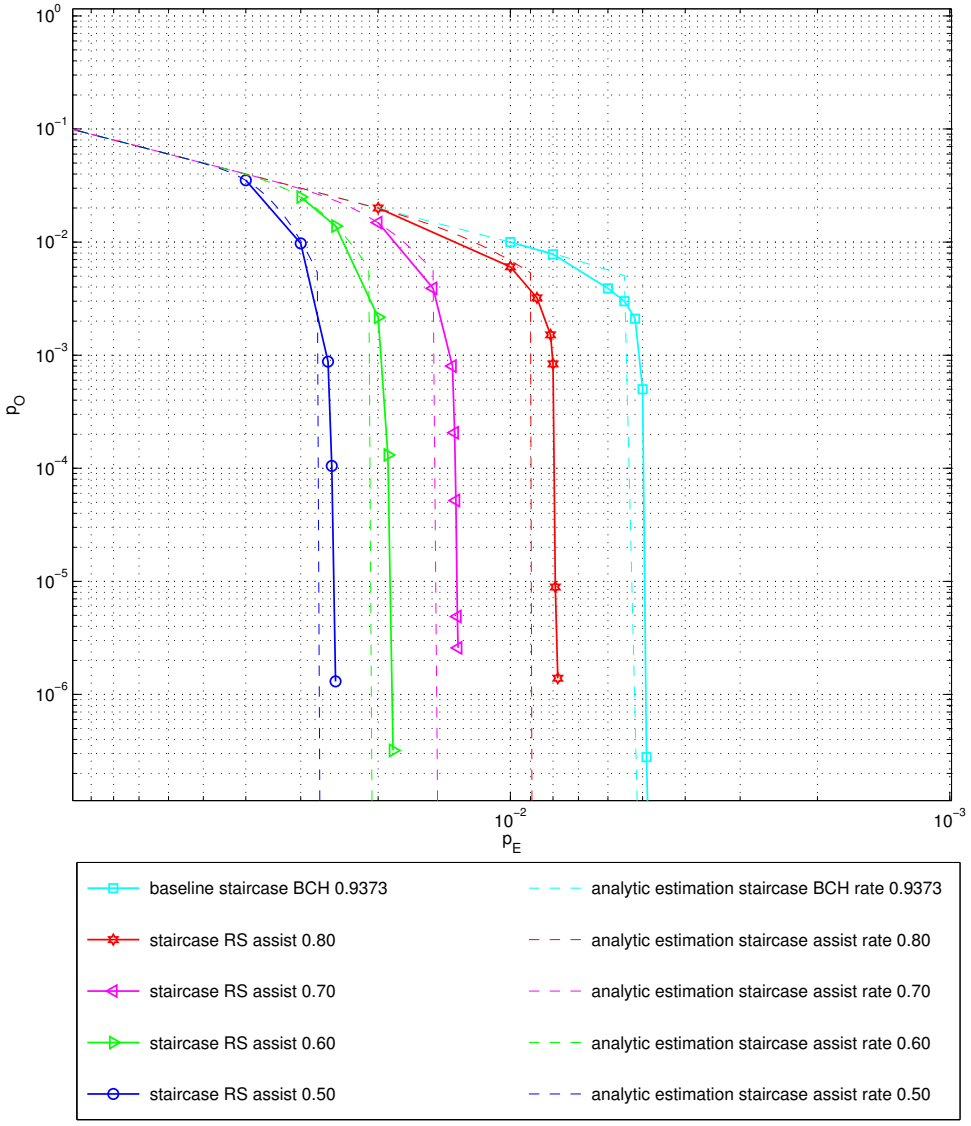


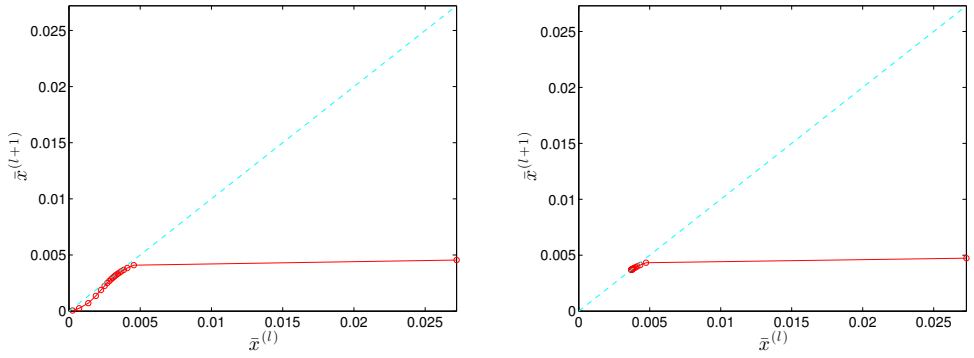

Figure 6.5: Performance of rate-adaptive Staircase codes with RS assistance, compared to analytic estimation on a random-error channel.

is at a bit error probability of 0.00533. The iterative decoding threshold of the BCH Staircase core code can be noticed in the Figure 6.5 at $p_E = 0.00533$ for the curve of the analytic Staircase BCH code, and at $p_O = 0.00533$ for the curves of the analytic Staircase assist codes where the sharp bends are shown. It can be seen that the curves of rate-adaptive Staircase codes coincide with the upper bound of the RS assist codes at high input bit error probability, the actual water-fall region occurs at a lower input bit error probability than the analysis curves, because the iterative decoding threshold gives the bound for unsuccessful iterative decoding, above which the iterative decoding will definitely fail; thus below the iterative decoding threshold, the iterative decoding is a possible success.

We track the evolution of bit error probability to estimate the iterative decoding threshold of the rate-adaptive Staircase codes via density evolution using Equation 6.10 as shown Figures 6.6-6.9. In each figure, the subfigure (a) shows density evolution of Staircase codes with rate assistance at input bit error probability $p_E$ below the iterative decoding threshold where the bit error probability converges to zero after many iterations; while the subfigure (b) shows density evolution of Staircase codes with rate assistance at input bit error probability $p_E$ at the adopted iterative decoding threshold where the bit error probability gets stuck at the bisectrix line after so many iterations and never converges to zero. The iterative decoding thresholds of the rate-adaptive Staircase codes are found to be at 0.0273 for rate 0.50 from Figure 6.6, at 0.0205 for rate 0.60 from Figure 6.7, at 0.0146 for rate 0.70 from Figure 6.8 and at 0.0088 for rate 0.80 from Figure 6.9. We plot these thresholds as vertical dash lines in Figure 6.10 and observe that the density evolution is indeed consistent with the performance simulation.

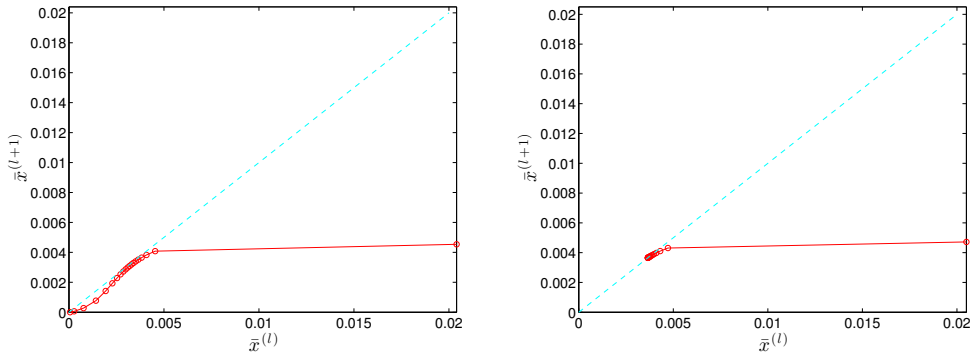
## 6.6 Performance Simulation on Burst-Error Channel

Figure 6.11 depicts performance of rate-adaptive Staircase codes on a burst-error channel with average burst length of 10. We observe that the baseline Staircase BCH code can correct fewer errors than on the random-error channel, so it can




(a) (b)

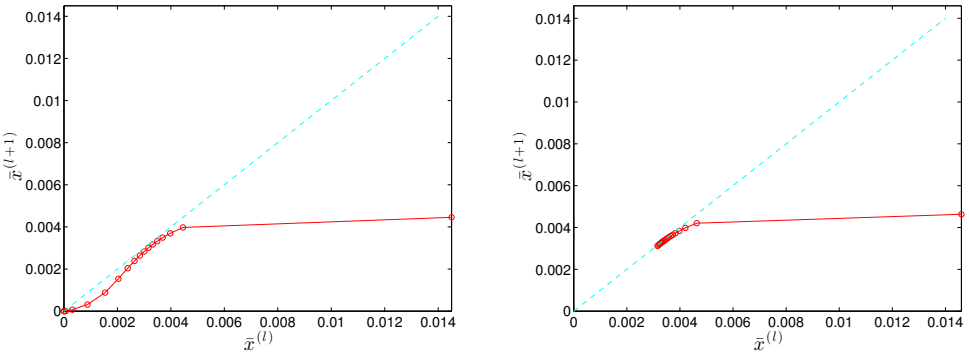
Figure 6.6: Evolution of the bit error probability of Staircase codes with assist rate of 0.5. (a) $p_E = 0.0272$, which is below the iterative decoding threshold; the graph converges to zero. (b) $p_E = 0.0273$, which is the adopted iterative decoding threshold; the graph does not converge to zero.




(a) (b)

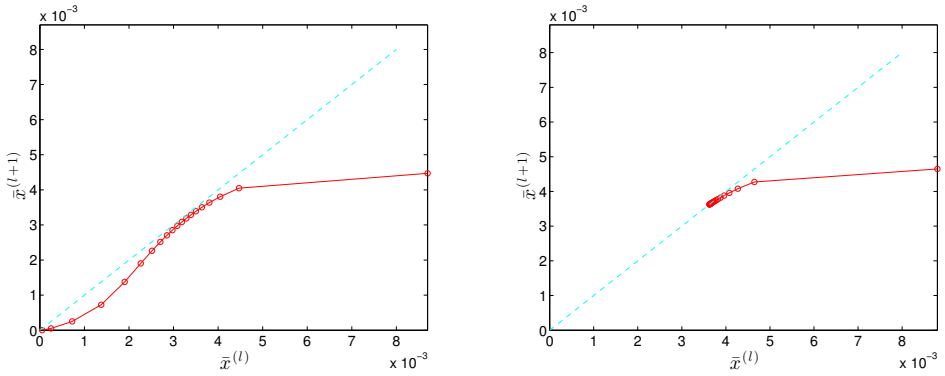
Figure 6.7: Evolution of the bit error probability of Staircase codes with assist rate of 0.6. (a) $p_E = 0.0204$, which is below the iterative decoding threshold; the graph converges to zero. (b) $p_E = 0.0205$, which is the adopted iterative decoding threshold; the graph does not converge to zero.

only correct for an input bit error probability of almost $p_E = 0.0014$ with an output bit error probability of $p_O = 10^{-6}$. However, the Staircase code with RS assistance at a rate of 0.8 can achieve the same output bit error probability at an input bit error probability of $p_E = 0.0047$. There is an inverse relationship between the rate of rate-adaptive Staircase codes and the input bit error prob-




(a) (b)

Figure 6.8: Evolution of the bit error probability of Staircase codes with assist rate of 0.7. (a) $p_E = 0.0145$, which is below the iterative decoding threshold; the graph converges to zero. (b) $p_E = 0.0146$, which is the adopted iterative decoding threshold; the graph does not converge to zero.




(a) (b)

Figure 6.9: Evolution of the bit error probability of Staircase codes with assist rate of 0.8. (a) $p_E = 0.0087$, which is below the iterative decoding threshold; the graph converges to zero. (b) $p_E = 0.0088$, which is the adopted iterative decoding threshold; the graph does not converge to zero.

ability, e.g., as the rate of the rate-adaptive Staircase code decreases the more input bit errors are corrected, thus allowing for the desired lower output bit error probability.

We plot the analytical performance estimation of the RS component codes from Equation 6.37 on a burst-error channel in Figure 6.11. The performance

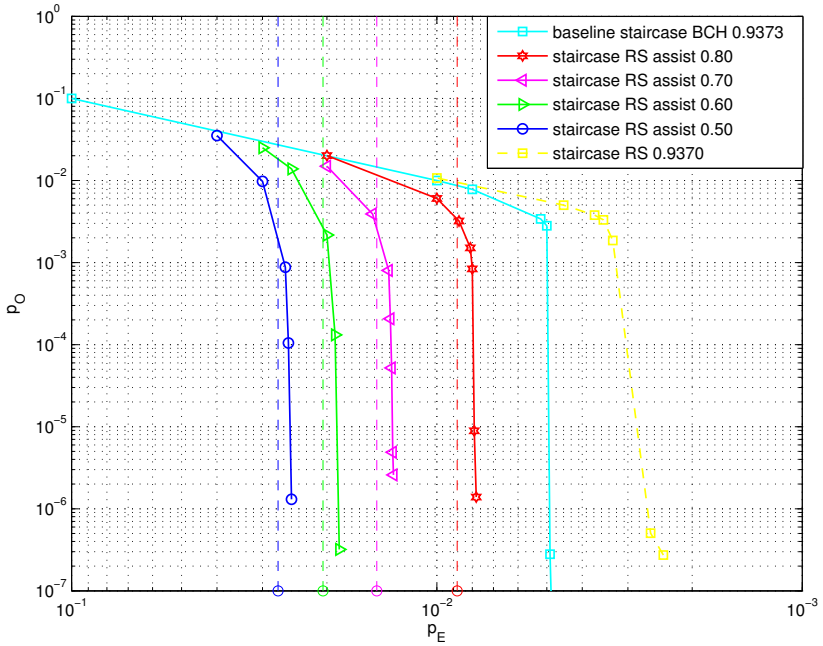



Figure 6.10: Performance of rate-adaptive Staircase codes with RS assistance on a random-error channel compared with iterative decoding thresholds from density evolution in dash lines. The performance curves for the baseline Staircase BCH code and the Staircase code with RS component codes are also included for reference

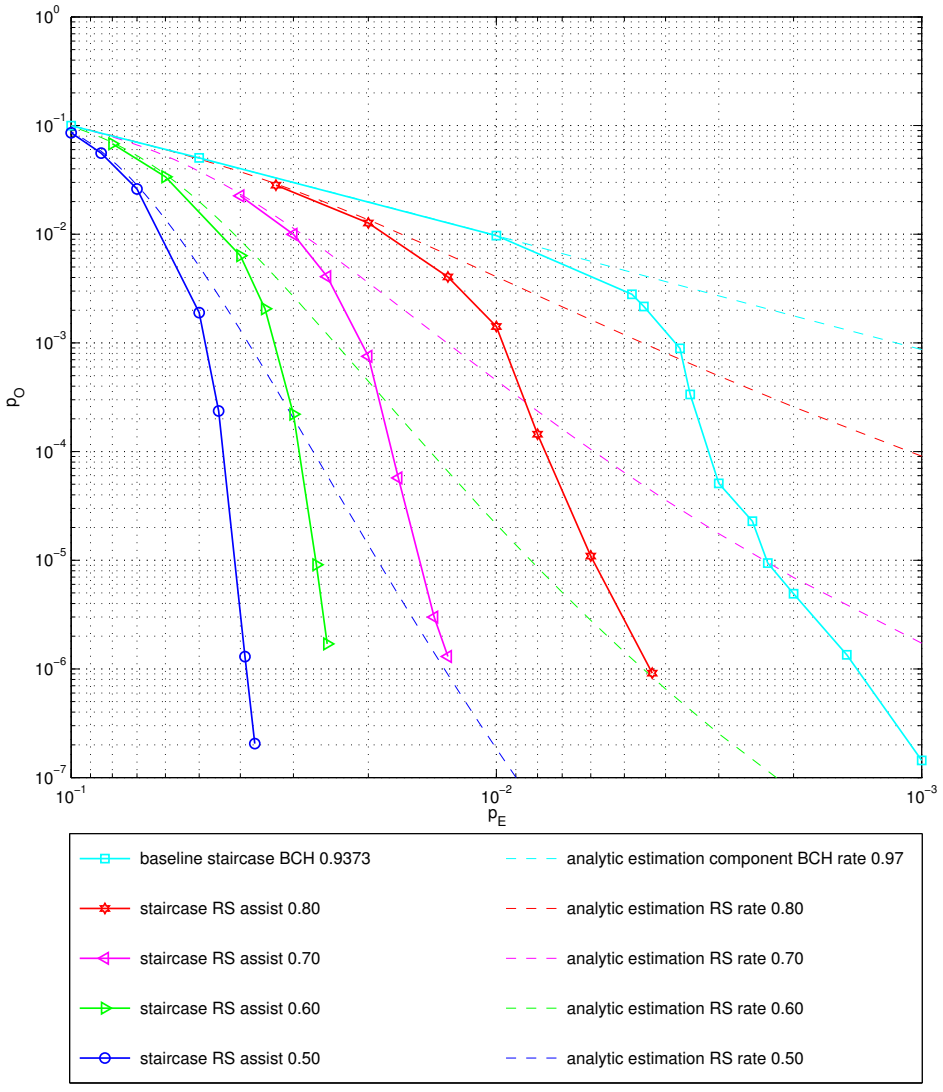



Figure 6.11: Performance of rate-adaptive Staircase codes with RS assistance on a burst-error channel with average burst length of 10.

of the Staircase with RS assistance coincides with the performance of the RS component codes at high input bit error probability. Thereafter, the smaller the input bit error probability, the larger the difference between the output bit error probability of the Staircase with RS assistance and their RS component codes, where the Staircase with RS assistance has smaller output bit error probability, e.g., the Staircase with RS assistance of rate 0.8 achieves $p_O = 10^{-6}$ at input bit error probability $p_E = 0.0044$, while the RS component codes achieves $p_O = 9 \times 10^{-4}$ at the same input bit error probability. This reduction of the output bit error probability of the Staircase with RS assistance is due to the iteration of the BCH Staircase core code, which starts to iterate after the number of errors are small enough, i.e., below the iterative decoding threshold of the BCH Staircase core code. Nevertheless, this reduction of the output bit error probability is not so fast as that on a random-error channel; thus there is no steep water fall region as for a random-error channel. This is because the BCH Staircase core codes do not perform very well on burst-error channels. The estimated iterative decoding threshold of the BCH Staircase core code is at 0.0037, which was found by peeling decoding analysis for a window of size 7 and the maximum number of iterations of 7, averaged for 1000 decision blocks on this burst-error channel. The iterative decoding threshold on the burst-error channel is naturally worse than the iterative decoding threshold for the random-error channel due to the worse performance of BCH codes on burst-error channels. Moreover, it does not guarantee error free decoding below the iterative decoding threshold, thus only the estimated bit error probability where iterations of the Staircase core code start can be determined from this value.

## 6.7 Decoding with RS assistance in each Iteration

We simulate the performance of rate-adaptive Staircase codes where the decoding of the RS component codes participates in every decoding iteration as the decoding window moves backwards, which means the decoding of the RS assist codes is done before the decoding of BCH component codes in every iteration

of decoding. This is different from the former simulations where the decoding of RS component codes is done only once, prior to the first iteration of the core decoding to remove the block errors.

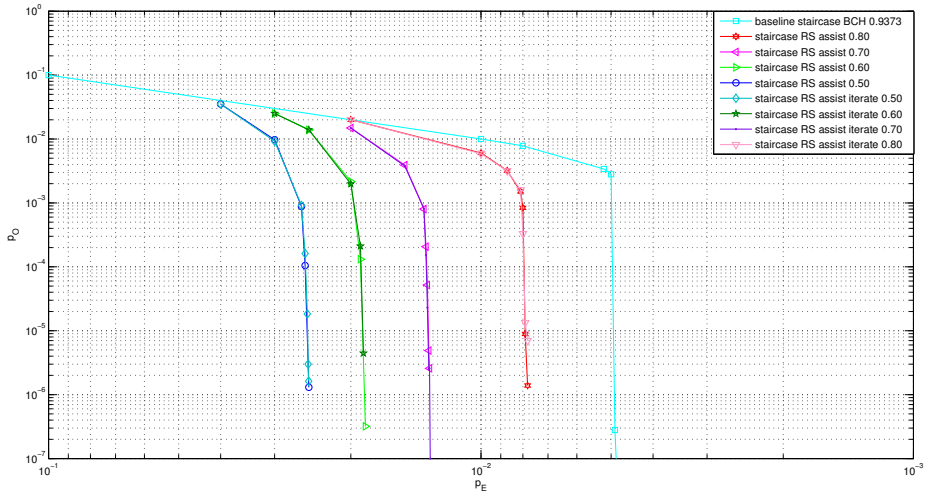



Figure 6.12: Performance comparison of the rate-adaptive Staircase codes on a random-error channel when RS decoding participates in every iteration.

We can see from Figure 6.12 and Figure 6.13 that the performances are not different from the former simulations on both random-error channels and burst-error channels. The reason is, that even though the RS assist codes have more error correction capability than the BCH core codes, they do not participate in concatenation to the neighbouring blocks; thus in the first decoding iteration nearly all of the correctable errors of the RS words have been corrected locally in each block, e.g., considering the Staircase core code with BCH component codes of $t = 3$ and the RS assist codes from GF($2^8$) with $t = 4$ (the minimum $t$ allowed for core codes of $t = 3$) at $p_E$ below the iterative decoding threshold of the core BCH codes: the maximum number of symbols changed per row is equal to 4 when decoding with the RS assist codes, and this is valid to all other blocks, thus the codewords are in the centre of each RS code decoding region with $t = 4$. Then after a decoding iteration of the BCH Staircase core code, the maximum number

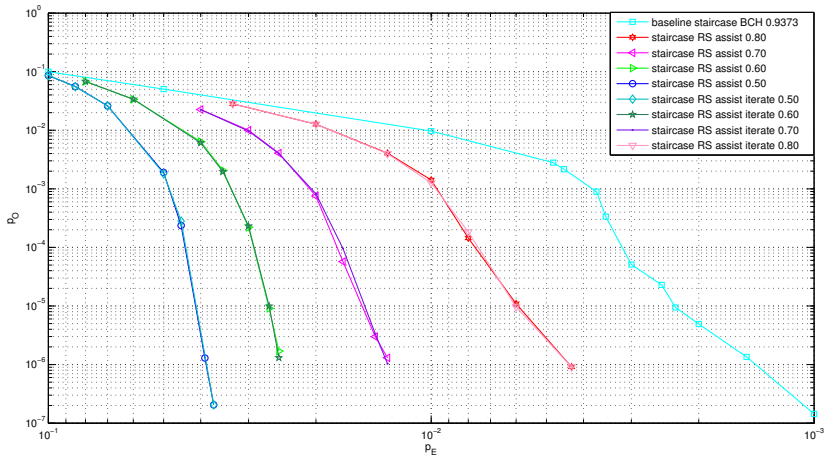



Figure 6.13: Performance comparison of the rate-adaptive Staircase codes on a burst-error channel with burst length 10 when RS decoding participates in every iteration.

of bits changed per row (altogether in the former block and the subsequent block) are only 3, because the iterative decoding of the core codes does not introduce more errors to the codewords. The change of these 3 bits per row yields the codewords, which are still in the same decoding region of the RS assist codes. As a consequence, to decode the RS assist codes in the next iterations does not bring about the improvement.

Hence, we persist with decoding the RS component codes only at the first iteration to save time of decoding (very significantly).

## 6.8 Throughput Simulation

We simulate the transmission with rate-adaption according to the Type-II hybrid ARQ scheme introduced in Section 6.3. It is assumed in the simulation that the transmission delay is small, the signal processing is fast enough not to cause any significant delay, and the feedback information (ACK and NACK) is error free, such that the "assist-part" still provides continuous flow of data. In the simulation, the baseline Staircase codes with BCH component of rate 0.9373 are

the core code. For each row of one assist block the assist part consists of 72 parity bits, therefore the resulting minimum rate for a maximum of 1 assist block is equal to 0.82. For a maximum of 2, 3, 4, 5, 6 assist blocks, the minimum rates are equal to 0.73, 0.66, 0.60, 0.55, 0.50 respectively. We define the throughput as

$$\eta = \frac{\text{accepted packets} \times \text{data bits in a packet}}{\text{transmitted packets} \times \text{code bits in a packet}} \tag{6.38}$$

$$= \frac{\text{number of data bits received}}{\text{total number of code bits transmitted}}. \tag{6.39}$$

where the data bits that are indicated as "received" are indeed correct (up to the extremely small failure rate of a CRC which is ignored).

Figure 6.14 shows the throughput of rate-adaptive Staircase codes in the Type-II hybrid ARQ scheme on a random-error channel. The curve "staircase RS 0.9370" is only for comparison of the Staircase code at (almost) the same rate of 0.9373 with RS components in the Staircase scheme: the latter clearly performs worse than the Staircase codes with BCH components on the random-error channel. We plot the estimated throughput analysis derived from Section 6.4 in Figure 6.14. The curve has a step shape due to the assumption that all windows in a decoding block have the same number of assist parts, which is contrary to the simulation where the number of assist parts is chosen as requested after the requirements in the first block of the decoding window. This also causes the analytic throughput curve to fall to zero at an input bit error probability smaller than the actual simulation curve. Each step corresponds to each code rate that matches the input bit error probability. It can be observed in Figure 6.14 that without the RS assistance, the throughput of the staircase BCH code already falls to zero at $p_E > 0.005$. With up to 6 RS-assist blocks, however, the throughput falls to zero at a much larger input bit error probability of $p_E = 0.02$. The curve "staircase ARQ max 1 assist block" shows that only one RS-assist block cannot help to increase the throughput. This is due to the fact that there are still not a sufficient number of parity bits of the RS component codes to correct the errors in the channel. For more than two RS-assist blocks, throughput can indeed be increased until the maximum available number of 6 assist-blocks is transmitted. The bold dashed curve shows the overall throughput with a maximum of 6 assist

blocks when the rate-*adaptive* scheme described in Sections 6.2 and 6.3 is applied.

Figure 6.15 shows the throughput of rate-adaptive Staircase codes in the Type-II hybrid ARQ scheme on a burst-error channel with average burst-length of 10. When the input bit error probability is larger than 0.0035, the Staircase code with BCH component codes has a throughput of zero, but when the rate adaption is applied, the throughput does not fall to zero until $p_E = 0.058$, which is also higher than on the random-error channel due to the better error correction capabilities of the RS assist parts for burst-error channels.

In Figure 6.16 and Figure 6.17 the throughput of rate-adaptive Staircase codes in the Type-II hybrid ARQ scheme on a burst-error channel with average burst-length of 30 and 50 are shown, respectively. It can be observed that the throughput of both figures are not as good as of the Figure 6.15 with average burst length of 10, and the figure for average burst length of 50 is the worst of all. This is because the RS assist codes from GF($2^8$) can correct better at the average burst length of 10 than the average burst length of 30 or 50.

## 6.9 Comparison to a Retransmission Scheme

To show the benefits of the rate-adaptive Staircase codes for Type-II hybrid ARQ schemes, we compare them with a retransmission scheme, in which the transmitter resends each block of Staircase codes successively after receiving each NACK from the receiver. It has the following steps:

1. The data bits for each Staircase block are encoded with a CRC for error detection to form ACK/NACK messages, then these blocks are encoded with the BCH Staircase code and are transmitted.

2. At the decoder, after receiving so many blocks as required for a decoding window, the Staircase code is decoded, and then the CRC error detection of the first block in the decoding window is evaluated. If errors are detected (as depicted in Figure 6.18 a)), a NACK message is sent to the encoder, otherwise this block is decided as "accepted" and an ACK is sent to acquire a new block.

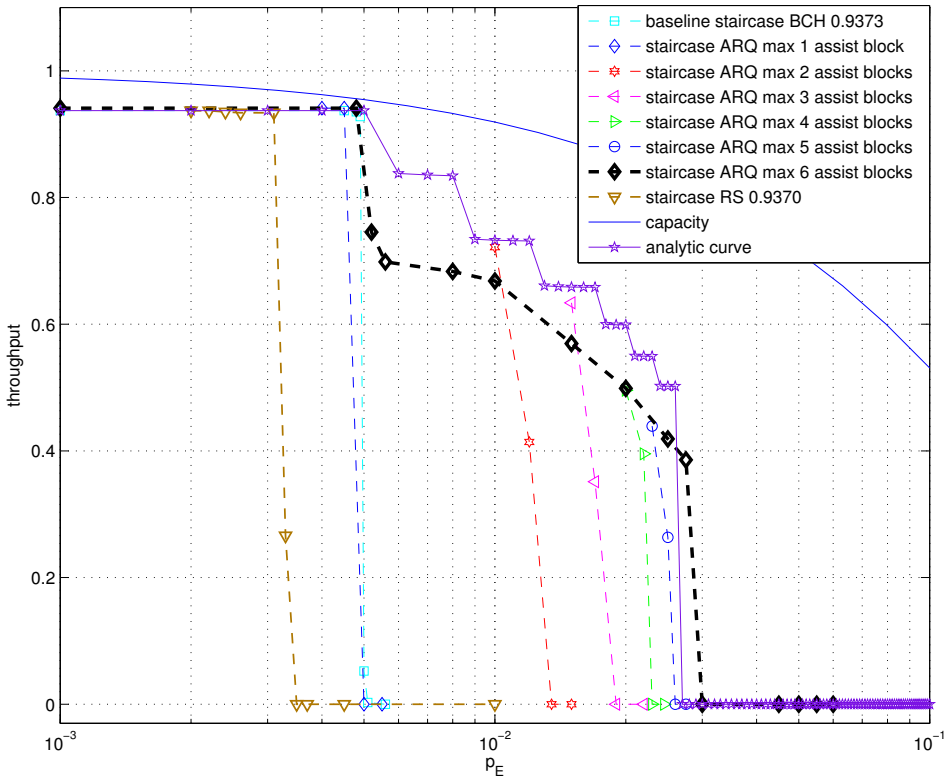



Figure 6.14: Throughput of the rate-adaptive Staircase code for different input bit-error probabilities on a random-error channel.

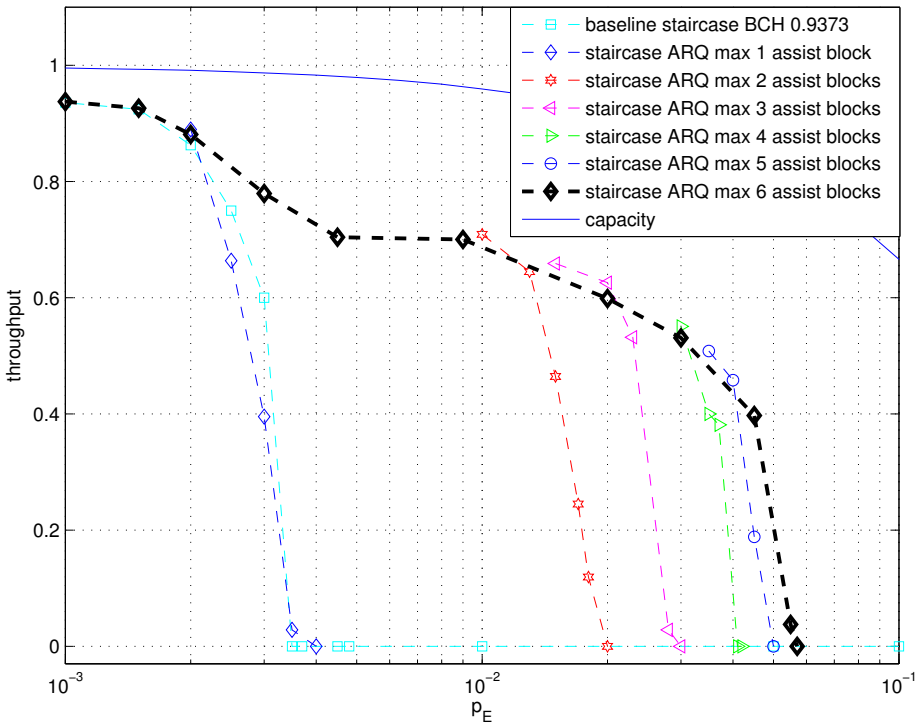



Figure 6.15: Throughput of the rate-adaptive Staircase code for different input bit-error probabilities on a burst-error channel with average burst length of 10.

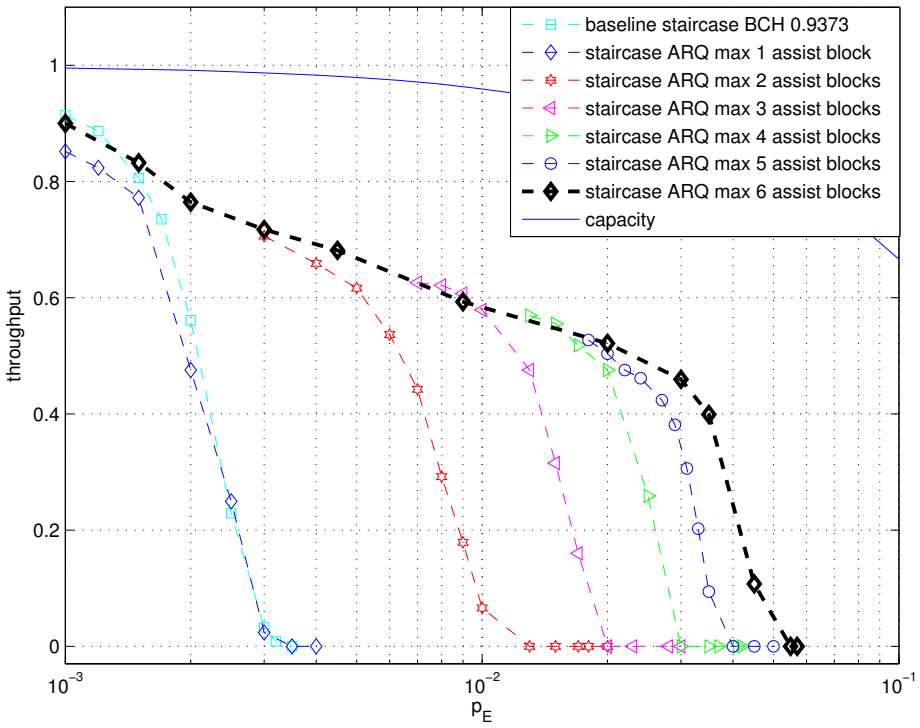



Figure 6.16: Throughput of the rate-adaptive Staircase code for different input bit-error probabilities on a burst-error channel with average burst length of 30.

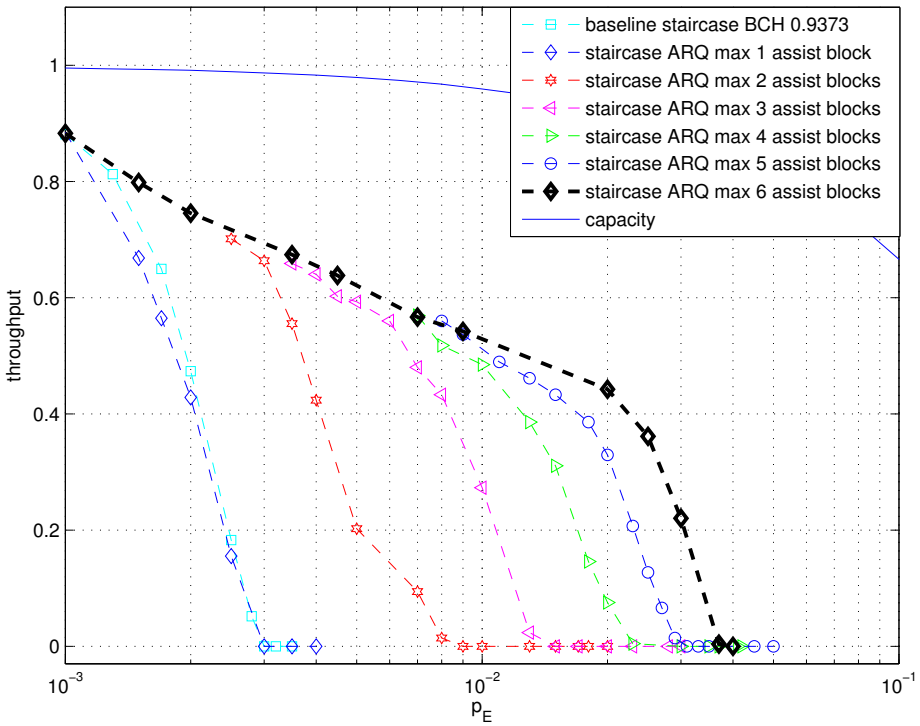



Figure 6.17: Throughput of the rate-adaptive Staircase code for different input bit-error probabilities on a burst-error channel with average burst length of 50.

3. When the transmitter receives an ACK, it transmits a new block; otherwise when NACK was received and the maximum number of retransmissions allowed is not reached, the transmitter resends the acquired block to the receiver.

4. At the decoder, after the decoder has received the retransmitted block, it replaces the former received block with the new received one (as depicted in Figure 6.18 b), c)), then the iterative Staircase decoding is processed and the CRC checks for errors. If there is no error in the first block of the decoding window (depicted in Figure 6.18 d)), this block is decided as "accepted" and an ACK is sent to acquire a new block. Then the decoding window is shifted forward to decode the new block (depicted in Figure 6.18 e)).

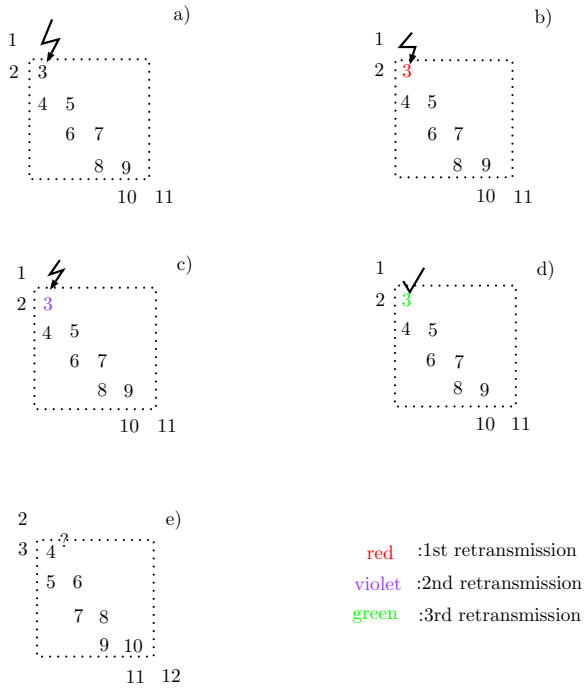



Figure 6.18: Illustration of Staircase code decoding with retransmission scheme.

We simulate both schemes on a randomly varying channel between input bit error probability $p_E = 0.004$, for which the Staircase core codes can certainly correct errors, and an upper value of input bit error probability $p_{UE}$. The maximum number of retransmissions per block allowed is bounded by the number

of maximum assist blocks of the ARQ scheme such that a fair comparison is achieved.

Figure 6.19 depicts the simulation results of rate-adaptive ARQ scheme compared to the retransmission scheme on varying channels. When the channel is randomly varying between $p_E = 0.004$ and $p_{UE} = 0.006$, the throughput of the ARQ schemes is equal to 0.78, whereas the throughput of the retransmission schemes is equal to 0.07. If the channel varies between $p_E = 0.004$ and a higher value of $p_{UE}$, the throughput of the ARQ schemes gets worse, but it is still better than the retransmission schemes, which always has the throughput below 0.1.

We note that even though the rate-adaptive scheme has a drawback that the extra RS decoding causes more decoding delay due to the computation of RS assistance than direct retransmission, it has a lot better throughput performance to cope with the varying channel conditions.

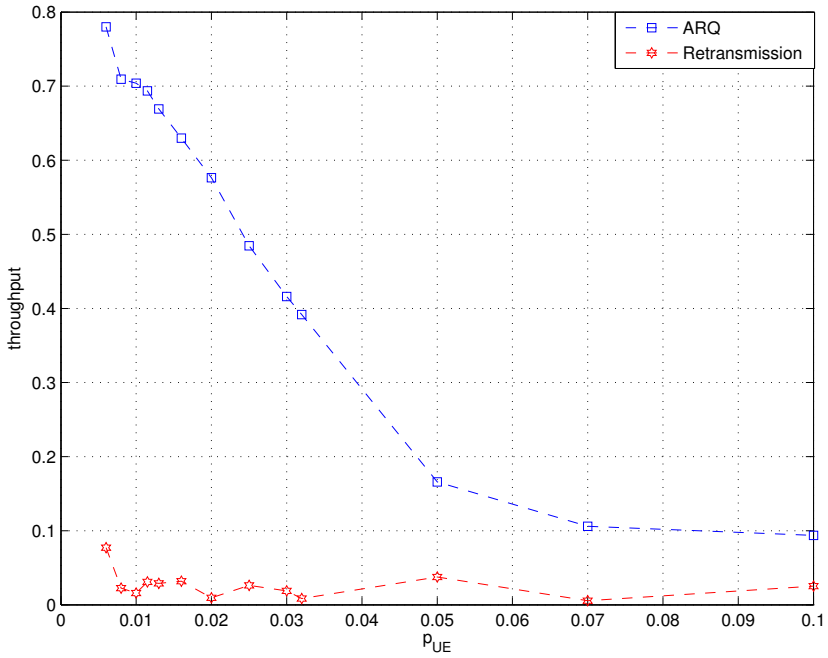



Figure 6.19: Throughput of the rate-adaptive Staircase code with ARQ compared to retransmission.

## 6.10 Conclusion

We proposed a rate-adaptive Staircase code for use on time-varying high-rate wireless channels. These rate-adaptive Staircase codes have RS codes as component codes to assist the BCH core codes used in the standard Staircase scheme. The advantages of using RS codes are burst-error correction capability and, importantly, flexibility in puncturing, therefore enabling their application in type-II hybrid ARQ schemes. The performance and throughput analysis of these rate-adaptive Staircase codes are derived. The simulations of the novel rate-adaptive Staircase codes show better performance and throughput on a wide range of input error probabilities in comparison to a standard Staircase code without rate-adaptation. Moreover the comparison of the proposed rate-adaptive Staircase code to a direct retransmission scheme on a varying channel shows that this rate-adaptive Staircase code has higher throughput on a wide range of input bit error probabilities. Consequently, the proposed scheme is suitable for high-rate wireless transmission on the channel that varies over time.

# Chapter 7

# Staircase Codes in Distributed Source Coding

Because Staircase codes have high performance on decoding codewords without knowledge of channel state information, together with the characteristic of a linear channel code that the generator matrix ($\mathbf{G}$) and the check matrix ($\mathbf{H}$) can be simply exchanged to get a system for linear source coding, we investigate the possibility of using Staircase codes in the framework of distributed source coding (DSC). Distributed source coding is based on the work by Slepian and Wolf [70], which gives the admissible rate region for lossless encoding of two correlated information sequences; the source encoder has no knowledge about this correlation whereas the decoder has both encoded message sequences at hand as depicted in Figure 7.1. The theory for lossy source coding that combines the results from Slepian and Wolf with quantization was derived later on by Wyner and Ziv [83]. The DSC concept is widely discussed for many applications such as wireless sensor networks [84] where the source data usually have correlation to the data from the nearby nodes; sending the compressed data instead of the whole bit stream can reduce the energy consumption of the nodes and extend the battery lifetime. Further application thereof is distributed video coding (DVC) [59], [2], [30] where the consecutive video frames have correlation to each other: sending the compressed frame using distributed source coding pushes the computational burden to the decoder that in some applications has more computational resources, and thus the

encoder requires less efficient hardware than in the conventional advanced video coding (AVC). Other applications are for example stereo video coding, spectrum sensing, multimedia streaming over heterogeneous networks etc.

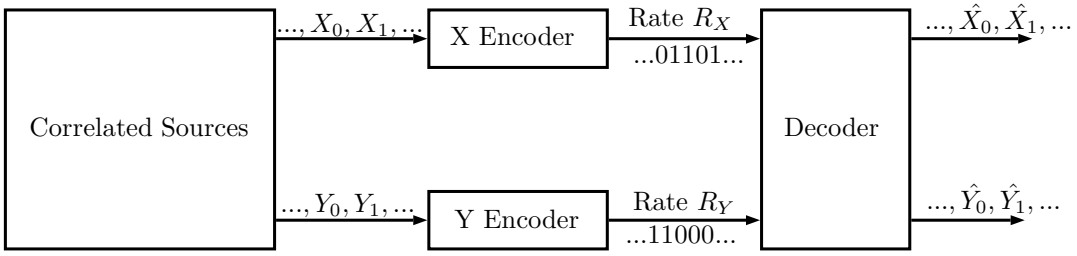



Figure 7.1: Correlated source coding configuration. [70]

## 7.1 Slepian-Wolf Coding and Code Designs

Shannon's source coding theorem [67] implies that the transmission of a single source $\mathbf{X} = (X_1, X_2, ...X_n)$ with independent realizations of $X$, taking values in the set $\mathcal{A} = \{1, 2, ...A\}$ must have rate of at least

$$R \geq H(X) \qquad \text{in } [\frac{\text{bits}}{\text{character}}] \tag{7.1}$$

$$\tag{7.2}$$

such that it can be recovered with arbitrary small error, and therefore R is said to be admissible. Equivalently, the upper limit of compression is equal to $1/H(X)$ measured in source letters per encoded binary digits [69]. The entropy of random variable $X$ is

$$H(X) = -\sum_{i=1}^{A} p_X(i) \log_2 p_X(i) \tag{7.3}$$

where $p_X(x) = \Pr[X = x]$ is the probability distribution of $X$.

For two correlated sources $X$ and $Y$ that have the joint probability distribution $p_{XY}(x, y) = \Pr[X = x \text{ and } Y = y]$, there exist the encoding rates $R_X$ and $R_Y$ such that the sequences $\hat{X}$ and $\hat{Y}$ after source decoding have arbitrary

small error probability: the rate pair $(R_X, R_Y)$ is called an admissible rate point. The associated information-theoretic parameters such as $H(X,Y)$, $H(X)$, $H(Y)$, $H(X|Y)$ and $H(Y|X)$ can be obtained from the joint probability distribution $p_{XY}(x,y)$ and can be found in [70]. The admissible rate region for the rate pair $(R_X, R_Y)$ is given by

$$R_X \geq H(X|Y) \tag{7.4}$$

$$R_Y \geq H(Y|X) \tag{7.5}$$

$$R_X + R_Y \geq H(X,Y) \tag{7.6}$$

and can be seen in Figure 7.2 which is located on the upper right of the blue line.

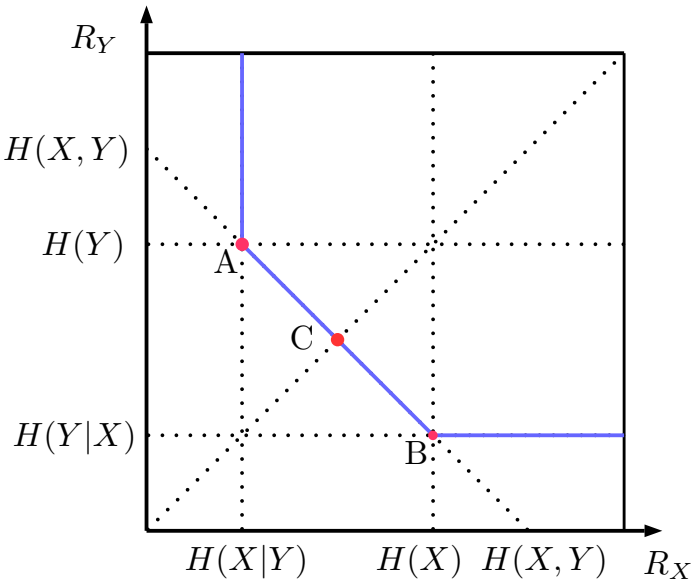



Figure 7.2: Slepian-Wolf rate region for two sources. [70]

The corner point A in Figure 7.2 corresponds to the problem of *source coding of $X$ with side information $Y$* as depicted in Figure 7.3, which implies that the decoder knows $Y$, which is compressed with rate $R_Y = H(Y)$, and the encoder knows both $X$ and $Y$ such that compression of X has rate $R_X = H(X|Y)$ following the equality

$$R_X + R_Y = H(X|Y) + H(Y) = H(X,Y). \tag{7.7}$$

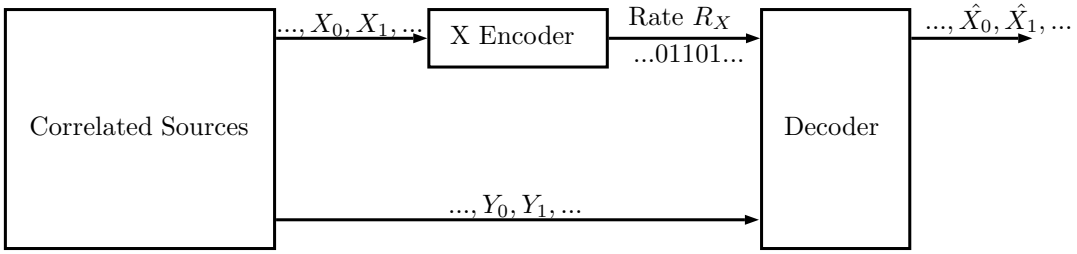



Figure 7.3: Lossless source coding with side information at the decoder. [84]

The parameter $X$ and $Y$ can be swapped resulting in the corner point $B$. This scheme is known as *asymmetric* SW coding. When the sources $X$ and $Y$ are compressed with the same rate corresponding to point $C$ the scheme is known as *symmetric* SW coding. All other points along the blue line between points A and B are then said to be *nonasymetric* SW coding, and they can be achieved by time sharing between points A and B.

We will consider here the special case of asymmetric SW coding with a *twin binary symmetric source* for that the sources $X$ and $Y$ are binary random variables defined as [80]

$$\begin{array}{rcrcl} p_X(0) & = & p_X(1) & = & \frac{1}{2}, \\ p_Y(0) & = & p_Y(1) & = & \frac{1}{2}, \\ p_{Y|X}(0|1) & = & p_{Y|X}(1|0) & = & p, \\ p_{Y|X}(0|0) & = & p_{Y|X}(1|1) & = & 1-p, \end{array} \tag{7.8}$$

with $H_2(p) = -[p\log_2(p) + (1-p)\log_2(1-p)]$. Therefore the entropies of twin binary symmetric sources are [80]

$$\begin{aligned} H(X) &= 1, \\ H(Y) &= 1, \\ H(Y|X) &= H_2(p), \\ H(X|Y) &= H_2(p). \end{aligned} \tag{7.9}$$

For asymmetric SW coding at corner point A we have $R_Y = H_2(Y) = 1$ and $R_X = H(X|Y) = H_2(p)$, on that account the correlation of $X$ and $Y$ at point A

is usually modeled by a *virtual* binary symmetric channel (BSC) with crossover probability $p$.

Considering the case of asymmetric SW coding, the constructive method achieving the SW bound proposed by Wyner [82] suggested the close relation of DSC to linear channel coding known as *syndrome approach* [23]. The source sequences $x$ of length $n$ are partitioned to cosets of a binary linear $(n, k)$ channel code that is defined by an $(n-k) \times n$ parity check matrix $\mathbf{H}$. The coset codes, which are equivalent to syndrome bits obtained from $s = x\mathbf{H}^T$, are transmitted by the encoder, and hence the compression rate is equal to $n : (n-k)$ for a full rank $\mathbf{H}$ matrix. The decoder thereafter receives the syndrome $s$ knowing the side information $y$ and then tries to decode the original sequence $\hat{x}$. In some applications, where the correlation between the sources are not constant, rate-adaptation is required. The *parity approach*, which can be easily constructed for rate-adaptivity by puncturing the parity bits, was proposed by [1] and [26]. In parity approach the source sequences are firstly systematically encoded with generator matrix $\mathbf{G}$ and then only the parity bits $x_p$ are sent such that the compression is achieved. The decoder, upon receiving the parity bits having the side information $y$ available, can estimate the source sequence $\hat{x}$.

There are many practical code designs based on channel codes. The first syndrome approach is known as DISCUS [58], where trellis based quantisation and coset construction were proposed. Designs using efficient Turbo codes based on the syndrome approach was proposed by [65] and based on the parity approach were proposed by [4], [1], and [26]. Because LDPC codes with a believed propagation decoder are near-capacity achieving, using LDPC codes for the DSC based syndrome approach also perform better than Turbo codes [47]. Due to the requirements for practical DSC that codes should be rate-adaptive, incremental and near the SW bound, a *serially-concatenated-accumulated code* (SCA) [16] was proposed. The SCA code generator is composed of an accumulator, an interleaver, and a set of base codes, which could be single parity check codes or extended Hamming codes. The decoder decodes the received syndrome together with the side information $y$ using the turbo decoding algorithm that iterates between the decoder of the accumulator code and the decoder of the base codes. The LDPC accumulate (LDPCA) and sum LDPC accumulate (SLDPCA) codes proposed for

high compression rate-adaptive DSC by [78] are efficient. The encoder consists of an LDPC syndrome former concatenated with an accumulator (additionally an adder in case SLDPCA codes), whereas the decoder can handle rate-adaptivity by modifying its decoding graph as it receives the additional syndrome bits. Lately BCH codes are proposed by [66] for application of rate-adaptive DSC with short to medium sequence length in a high correlation scenario.

## 7.2 Staircase Codes in DSC implementation

Due to better performance of Staircase codes over their component codes, which could be BCH, RS, etc., at higher input bit error probability of the BSC channels, which correspond to high correlation of the sources, together with the result from [66] stating that the BCH codes are more suitable to compress information sources at high correlation than LDPCA codes for moderate to short block length, we implemented the Staircase codes in an asymmetric SW framework with twin binary sources using the parity approach expecting that the concatenation of component codes and the iterative decoding of Staircase codes will improve the compression capability of the source codes in DSC schemes.

The implementation model is configured as shown in Figure 7.3. The source sequence $X_i$ of length $(m-r) \times m$, which has correlation to the source sequence $Y_i$ with correlated parameter $p$, is arranged in the block of the Staircase codes encoder as depicted in Figure 7.4, where $m$ is the number of rows or columns in Staircase codes array, $r$ is the number of columns of parity bits. The resulting parity block $X_{p,i}$ with length $r \times m$ generated from the Staircase codes encoder is sent through the ideal channel and arrives at the Staircase codes decoder error-free. The decoder decodes the received parity bits $X_{p,i}$ by placing them to the block of Staircase codes whereas the sequence of the side information $Y_i$ of length $(m-r) \times m$ is placed in the array in the position of $X_i$, and then the decoder decodes iteratively through the decoding window until the maximum allowable number of iterations is reached, thus the sequence $\hat{X}$ is recovered. The scheme has a normalized source encoding rate equal to

$$\text{source rate} = \frac{r}{m-r} \qquad \text{[bits per character]} \tag{7.10}$$

and a compression ratio equal to $\frac{m-r}{r}$. The encoder still has lower complexity in comparison to the decoder, which is crucial to DSC. We aim to push computational burdens to the decoder, i.e., the encoder only has to store the current block and the former block, while the decoder needs to store as many blocks as the predefined decoding window is long. Besides, the decoder needs to do iterative decoding which is composed of many steps of component codes decoding. For details of encoding and decoding of the Staircase codes see Chapter 3.

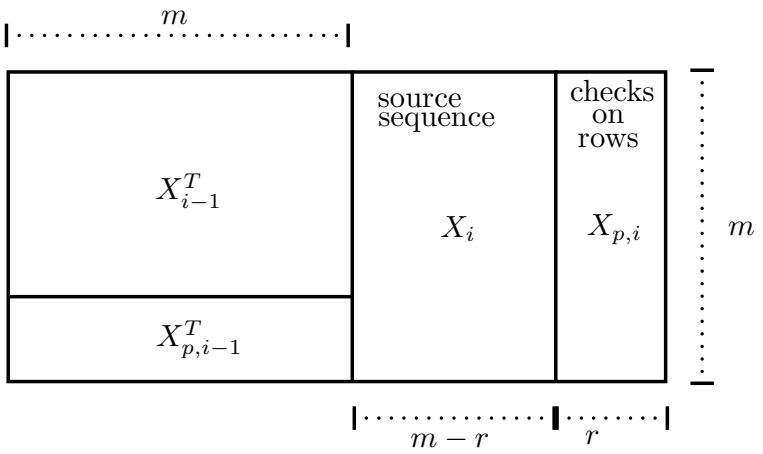



Figure 7.4: Staircase codes array for encoding in DSC.

If BCH component codes are constructed on another number field, the block length and the number of parity bits are different, which yields a different source rate of the Staircase codes. The BCH code from GF($2^q$) has the block length equal to $n = 2^q - 1$ and the number of parity check bits $r \leq qt$ where $t$ is error correction capability, thus the source rate is given as

$$\begin{aligned} \text{source rate} &= \frac{r}{m-r} \\ &\leq \frac{qt}{\frac{n}{2} - qt} \\ &= \frac{qt}{\frac{2^q-1}{2} - qt}. \end{aligned} \tag{7.11}$$

Figure 7.5 shows the estimated source rate for different $q$. It can be seen that Staircase codes from a smaller field have higher source rates; moreover the source

rates thereof increase faster with the increasing of $t$. Therefore, the application of BCH Staircase codes from a small field in DSC is quite limited as a small source rate cannot be achieved. Note that the source rate greater than one is not anymore source coding, because there are more encoded bits than data bits.

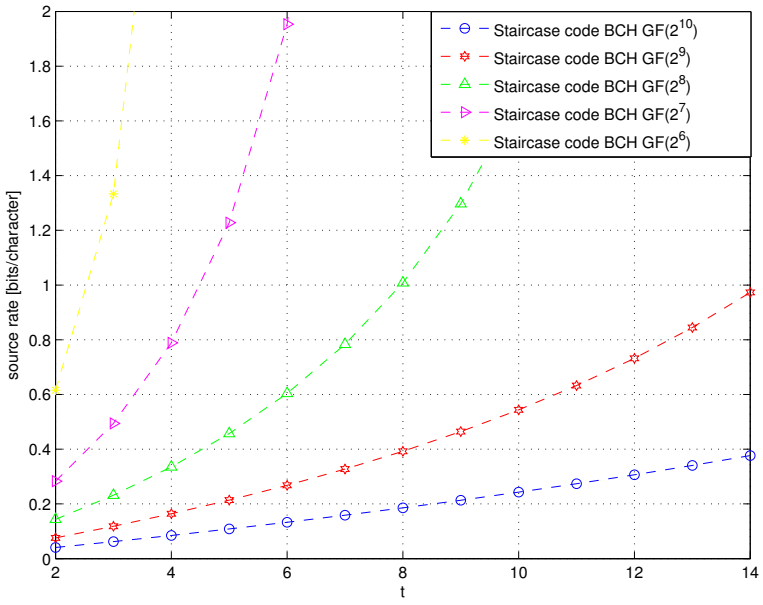



Figure 7.5: Estimated source rate of BCH Staircase codes

## 7.3 Performance Analysis of Staircase Codes in DSC

From the system set up, the block $X_{p,i}$ is transmitted error free through the ideal channel, whereas the block $Y_i$ is available at the receiver with crossover probability $p$ from the correlated source $X_i$. The same is true for the other block indices. In a decoding window as illustrated in Figure 7.6, each row of the 2 consecutive blocks is decoded with a bounded minimum distance decoder of the BCH component codes with error correction capability $t$. Thus, for the analysis

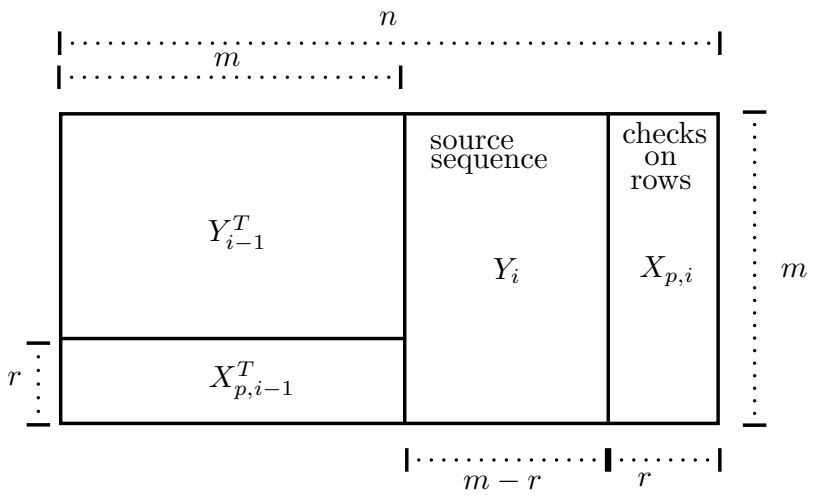



Figure 7.6: Staircase codes array for decoding in DSC.

at high $p$ above the iterative decoding threshold, the lower bound of word error probability of the first $m-r$ rows is given by (see Section 4.2)

$$P_{w1} \geq \sum_{i=t+1}^{n-r} \binom{n-r}{i} p^i (1-p)^{(n-r-i)} \,, \tag{7.12}$$

where $p$ is the crossover probability of the correlated sources. The term $\binom{n-r}{i}$ is for the number of possibility that $i$ bit errors occurring in $n-r$ positions, while the last $r$ positions have no error due to the error free transmission of the block $X_{p,i}$. The term $p^i(1-p)^{(n-r-i)}$ is the probability of a particular $i$-bit error in $n-r$ positions. The summation starts from $t+1$ where there are more errors than the error correction capability of the BMD to the possible number of errors occurring in those rows. The bit error probability of the first $m-r$ rows is thus

$$P_{b1} \geq \sum_{i=t+1}^{n-r} \frac{i}{n} \binom{n-r}{i} p^i (1-p)^{(n-r-i)}, \tag{7.13}$$

where $\frac{i}{n}$ is for the number of errors relative to the number of codeword bits. The lower bound of word error probability of the last $r$ rows is given by

$$P_{w2} \geq \sum_{i=t+1}^{m-r} \begin{pmatrix} m-r \\ i \end{pmatrix} p^i (1-p)^{(m-r-i)}, \tag{7.14}$$

where the term $\binom{m-r}{i}$ is for the number of possibility that $i$ bit errors occuring in $m-r$ positions while the other $m+r$ positions have no error due to the error free transmission of the block $X_{p,i}$, and $X_{p,i-1}^T$. The term $p^i(1-p)^{(m-r-i)}$ is the probability of a particular $i$-bit error in $m-r$ positions. The summation starts from $t+1$ where there are more errors than the error correction capability of the BMD to the possible number of errors occurring in those rows. The bit error probability of the last $r$ rows is given as

$$P_{b2} \geq \sum_{i=t+1}^{m-r} \frac{i}{n} \begin{pmatrix} m-r \\ i \end{pmatrix} p^i (1-p)^{(m-r-i)} \ , \tag{7.15}$$

where $\frac{i}{n}$ is for the number of errors relative to the number of codeword bits. The bit error probability of one decoding window can finally be estimated as

$$P_b \geq \frac{(m-r) \cdot P_{b1} + r \cdot P_{b2}}{m}, \tag{7.16}$$

which is the average of $m-r$ rows with bit error probability $P_{b1}$, and $r$ rows with bit error probability $P_{b2}$. After the iterative decoding threshold is reached, the bit error rate reduces dramatically and can be observed as a waterfall region in the performance curve. For the analysis of waterfall region and error floor see Section 4.2.

Figure 7.7 shows the performance of the Staircase codes BCH component codes from GF($2^{10}$) with $t = 2, 3, 4$ with 2 bit CRCs in the DSC scheme and the performance analytic curves of the BCH component codes from GF($2^{10}$) in the DSC scheme given by Equation 7.16. We simulated the Staircase codes with BCH component codes with 2 bit CRCs as we know that higher error floor of the small-$t$ component codes, which degrades the performance of DSC, can be avoided by adding error detection bits. It can be observed that the performances

of BCH Staircase codes in the DSC scheme coincide with the analytic curves of BCH component codes in the DSC scheme at high crossover probability $p$. When $p$ is low enough the bit error rate of the BCH Staircase codes are lower than the analytic curves of the BCH component codes due to the iterative decoding of the Staircase codes as can be seen as waterfall region in the curves.

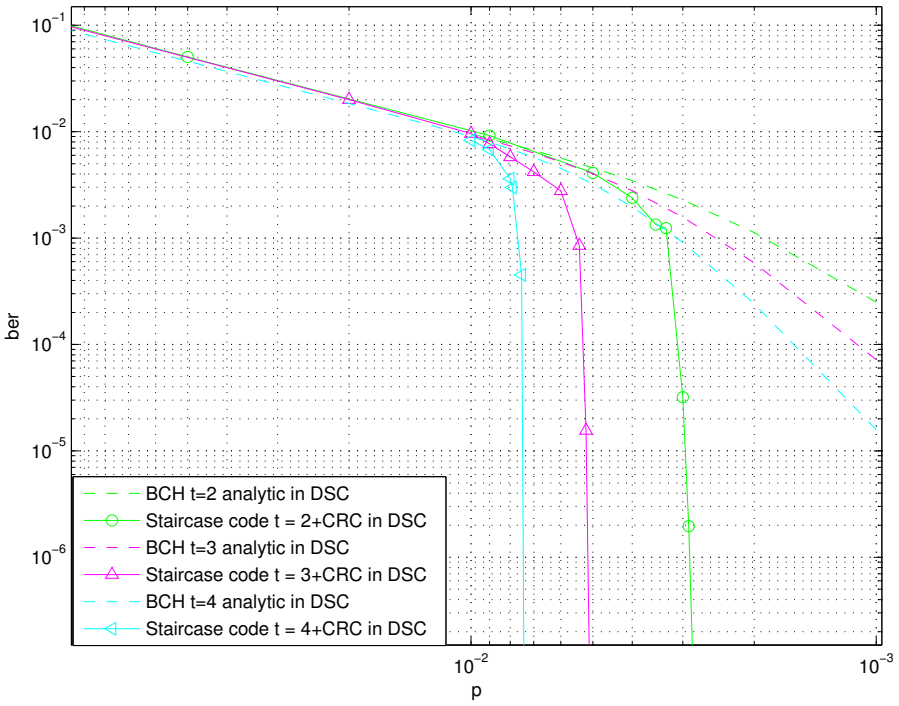



Figure 7.7: Performance of Staircase codes in DSC compared to the analytic BCH component codes in DSC.

## 7.4 Simulation Results

We are interested in Staircase codes from GF($2^{10}$) and GF($2^9$) which can achieve lower source rate, even though the block length of the Staircase codes with BCH component codes from GF($2^{10}$) is around $2.61 \times 10^5$ bits and from GF($2^9$) is around $6.52 \times 10^4$ which are quite long. We consider here the high correlation

scenario, where the conditional entropy $H(Y|X)$ between source $X$ and $Y$ is very low which corresponds to low crossover probability $p$; therefore $X$ and $Y$ are very similar. In this scenario the compression rate of the optimal LDPC codes are inferior to the compression rate of the optimal BCH codes from [66]. The decoded bit error rate over the crossover probability $p$ of the DSC using the Staircase codes with BCH component codes of different error correction capabilities $t$ are plotted in Figure 7.8. When a decoded bit error rate $\leq 10^{-5}$, we regard it as correct decoding and plot its source rate over conditional entropy $H(Y|X)$ as depicted in Figure 7.9. The same is done on Staircase codes from GF($2^9$) as depicted in Figure 7.10 and in Figure 7.11. These rate curves in Figure 7.9 and Figure 7.11 are compared to optimal rate-adaptive BCH codes from [66], which are rate-adaptive BCH codes of length 255 or 511 or 1023 with the lowest source rate at each value of $H(Y|X)$. The optimal rate-adaptive BCH from [66] are claimed to be better than LDPCA for length 1584 bits [78] in a high correlation scenario.

From Figure 7.8 and Figure 7.10 we can see that with the increasing of the crossover probability $p$, larger error correction capability $t$ is required for the BCH component codes, thus the source rate is higher according to Equation 7.11.

In Figure 7.9 and Figure 7.11 it can be observed that the bigger the error correction capability $t$ of the BCH component codes is, the wider is the gap from the SW-bound. In Figure 7.9, the Staircase codes with $t = 2, 3$ with CRC from GF($2^{10}$) have source rates close to the optimal rate-adaptive BCH codes from [66]. However, the source rates of the Staircase codes with higher $t$ are higher than those of the optimal rate-adaptive BCH codes from [66]. In Figure 7.11 the Staircase codes from GF($2^9$) have higher source rates than those of the optimal rate-adaptive BCH codes from [66] for all values of $t$. The reason for this inferior compression rate is due to the application of the parity approach in the Staircase codes, which is worse than the syndrome approach used by BCH codes in [66]. In spite of the iterative decoding of the Staircase codes, the BER has not been improved enough to cause the source rate smaller than the optimal rate-adaptive BCH codes from [66].

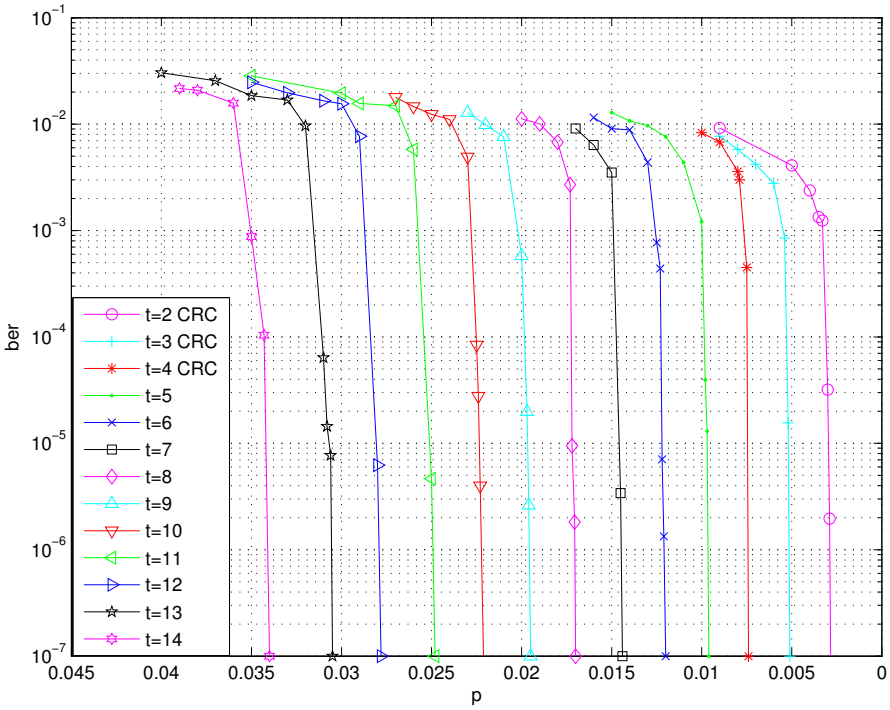



Figure 7.8: Bit error SW coding using a Staircase code with BCH component codes in GF($2^{10}$).

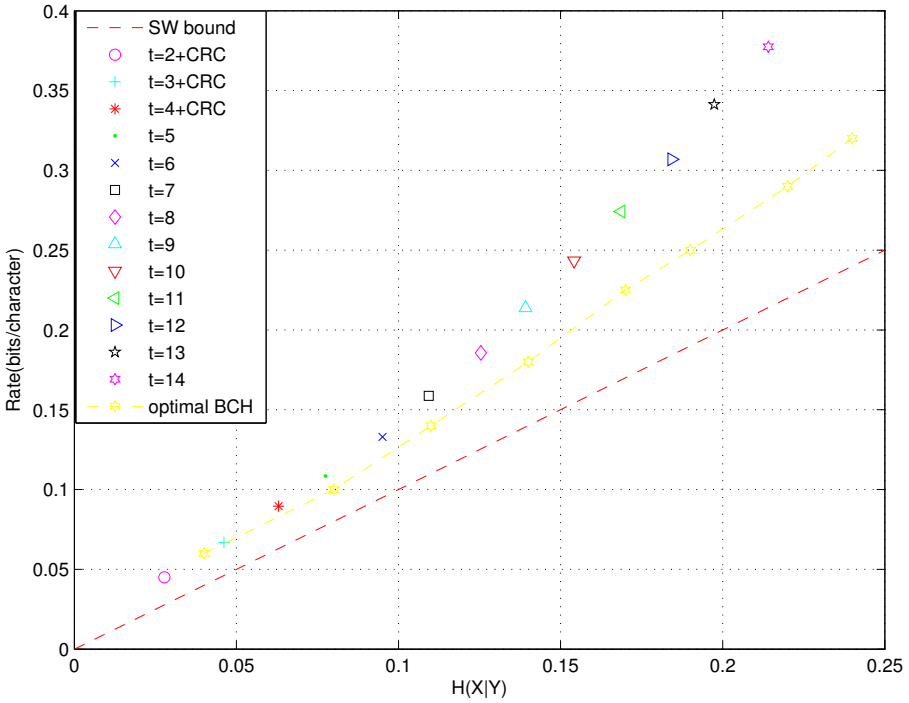



Figure 7.9: Rate curve of SW coding using a Staircase code with BCH component codes in GF($2^{10}$) compared to the optimal rate-adaptive BCH codes from [66].

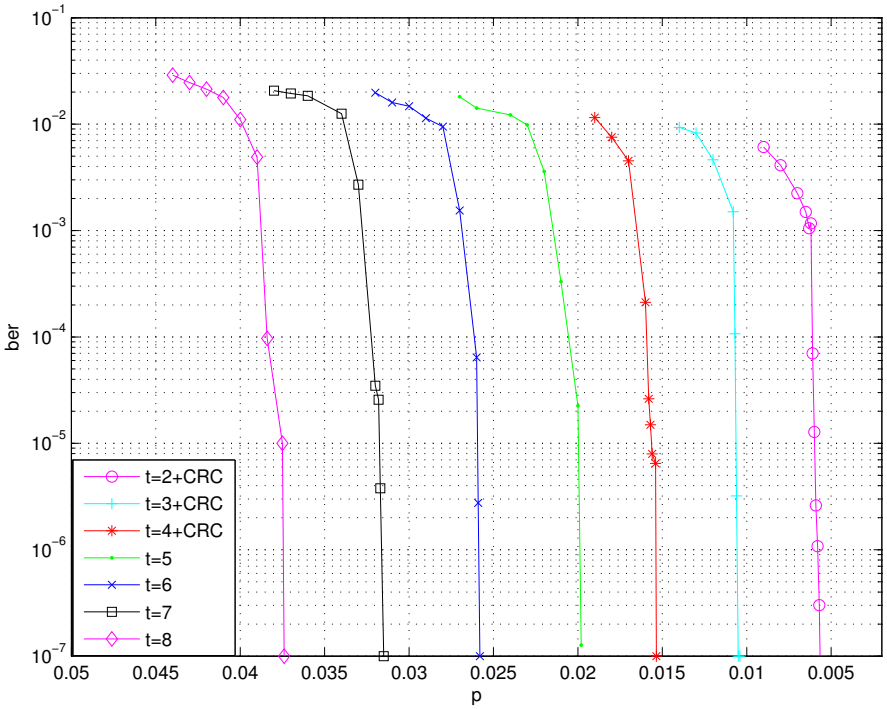



Figure 7.10: Bit error SW coding using a Staircase code with BCH component codes in GF($2^9$).

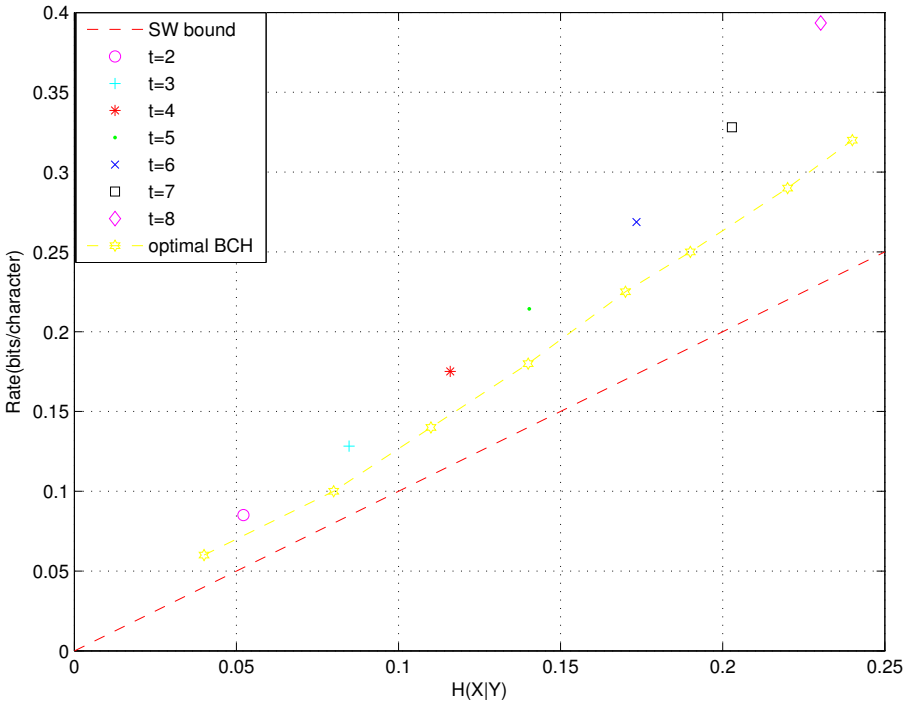



Figure 7.11: Rate curve of SW coding using a Staircase code with BCH component codes in GF($2^9$) compared to the optimal rate-adaptive BCH codes from [66].

## 7.5 Conclusion

We found that the Staircase codes can be implemented in the DSC scheme in a high correlation scenario, but it has inferior compression rate than using the syndrome approach with BCH codes, only the Staircase codes with BCH component codes from GF($2^{10}$) with $t = 2, 3$ have comparable source rates at very high correlation where $H(X|Y) < 0.05$.

The fixed array structure of the Staircase codes causes difficulties in adapting rates, which are required for the cases where correlations between sources are not constant. Using rate-adaptive RS codes as stated in Chapter 6 will definitely increase the compression rate, which is undesired.

Consequently, using BCH Staircase codes in a DSC scheme is proper only for specific component codes at very high correlation without rate-adaptivity.

# Chapter 8

# Conclusions

The study was set out to explore Staircase codes and their extensions. Staircase codes are concatenated product-like codes suitable for high-rate transmission with hard-decision decoding. They were originally designed for high-rate transmission on binary symmetric channels. The present study contributes to the application of these codes in some other areas; such as in burst-error channels, in time-varying channels, and in distributed source coding.

This section will summarize the empirical findings of the study's research on three aspects of Staircase codes.

**1. Staircase codes on burst-error channels**:

a: **Insertion of cyclic redundancy check codes for each component codes:** This is essential for component codes, which have small error correction capability, as it prevents erroneous decoding of the component codes and it lowers the error floor in the performance curve on both random-error channels and burst-error channels.

b: **Using RS codes as component codes:** We proposed using RS codes as component codes for high-rate Staircase codes transmitted on burst-error channels, on which the transmission of the baseline BCH Staircase code totally fails. This is due to the characteristic of the RS component codes, which are processed symbol-wise, and therefore are appropriate for burst-error channels. The decoding complexity per bit of the RS component code of length $n$ is proportional to the error correction capability $t$ and is inversely proportional to the symbol size $q$ for $n \gg t$. The complexity per bit and the decoding latency per bit of the

selected RS component codes are smaller than those of the baseline BCH codes for Staircase codes of the same rate. Thus RS Staircase codes are applicable for high-rate wireless transmission on the channel with burst errors without increasing decoding latency.

c: **Using diagonal interleaving:** We proposed using diagonal interleaving on Staircase codes transmitted on burst-error channels. This improves the performance of both BCH and RS Staircase codes on burst-error channels, in contrast to the well known row-column interleaving, which does not make any difference on burst-error channels, because the burst errors are distributed within rows or columns, which the iterative decoding of Staircase codes has achieved to correct. The diagonal interleaving of RS Staircase codes needs to be processed symbol-wise to maintain the burst-error correction capability of the RS component codes. The diagonal interleaving extends the burst-error correction capability of Staircase codes to a longer burst length with the trade off of more latency and memory usage. The symbol-wise diagonal de-interleaving of the RS Staircase codes processes with less time per bit than the bit-wise diagonal interleaving of the BCH Staircase codes. With the combination of the suggested methods, Staircase codes are able to correct burst errors with longer burst lengths.

**2. Staircase codes on a time-varying channel**:

We proposed a **rate-adaptive Staircase code**, which has RS codes as component codes to assist the BCH core codes. Through the puncturing of the RS codes the rate adaptivity is achieved. This rate-adaptive Staircase code is deployed in **Type-II hybrid ARQ** framework to improve the throughput performance. The performance and throughput simulation conformed to the theoretical performance and throughput analysis, which shows that with this rate-adaptation the Staircase code can correct errors in a wide range of input bit error probabilities. The comparison of this rate-adaptive Staircase code to a retransmission scheme shows a superior throughput performance of the proposed scheme in a wide range of input bit error probabilities. Hence, the proposed rate-adaptive Staircase code is suitable for high-rate wireless transmission on time-varying channels.

**3. Staircase codes in a distributed source coding (DSC) scheme**:

Even though the Staircase codes are possible to be implemented in DSC, the compression rate was shown to be inferior to the codes from literature. It is

only at very high source correlations $H(X|Y) < 0.05$ that the BCH Staircase code from GF($2^{10}$) has an equally good compression rate as the optimal codes from the literature. Consequently, only the BCH Staircase code from GF($2^{10}$) is applicable for usage in DSC schemes at restricted correlation values under the condition that a long block length is allowed, while the BCH Staircase codes from smaller field are not proper for DSC schemes.

Three major recommendations can be made for future work. The first concerns the error floor of the proposed method. To achieve very low error floor as stated in the Staircase codes analysis, a large number of bit realisations would have to be processed, which cannot be achieved with an implementation in software (e.g. MATLAB) due to time limitation. Hence parallel decoding through field-programmable gate arrays (FPGA) is an alternative to realize numerical investigation of such low error floor. The second recommendation concerns the analysis of the Staircase codes with the input bit error probability below the decoding threshold in burst-error channels. In random-error channels, the performance curve can be simply interpolated to reach the error floor; however the performance curves have different slopes for different codes and different burst-error characteristics in burst-error channels. This problem could possibly be solved with a mathematical derivation such that the performance of Staircase codes on burst-error channels can be predicted without extensive simulation. The third recommendation is the implementation of the proposed Staircase codes in a the real transmission system. The transmission could be optical or by radio frequency signals; however the burst or time-varying nature of such channels should first be evaluated, such that suitable parameters for rate-adaptive or burst-error correction Staircase codes can be selected.

Staircase codes are powerful channel codes for future wireless communication where the transmission relies on very high speed and thus high-rate, such that the decoding can be done in parallel without soft-decision requirements. The application of these codes on wireless channels requires adaptations, which were addressed in this work. This would significantly contribute to the improvement of wireless communication technology, which will then result in faster and more reliable data transmission.

# References


[1] A. Aaron and B. Girod. Compression with side information using turbo codes. In *Proceedings of the IEEE International Data Compression Conference (DCC)*, pages 252–261, April 2002. doi: 10.1109/DCC.2002.999963. 131

[2] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret. The DISCOVER codec: Architecture, techniques and evaluation. In *Proceedings of the Picture Coding Symposium (PCS'07)*, November 2007. 127

[3] T. Baicheva, S. Dodunekov, and P. Kazakov. Undetected error probability performance of cyclic redundancy-check codes of 16-bit redundancy. *IEE Proceedings–Communications*, 147(5):253–256, Oct 2000. ISSN 1350-2425. doi: 10.1049/ip-com:20000649. 61

[4] J. Bajcsy and P. Mitran. Coding for the Slepian-Wolf problem with turbo codes. In *Proceedings of the 2001 IEEE Global Telecommunications Conference (GLOBECOM '01)*, volume 2, pages 1400–1404, November 2001. doi: 10.1109/GLOCOM.2001.965721. 131

[5] E. A. Bender and S. G. Williamson. *Foundation of Combinatorics with Applications*. Dover, 2006. ISBN 0-486-44603-4. 104, 105

[6] S. Benjamin, F. Arash, H. Andrew, and F. Kschischang. Staircase codes: FEC for 100 Gb/s OTN. *Journal of Lightwave Technology*, 30(1):110–117, January 2012. ISSN 0733-8724. doi: 10.1109/JLT.2011.2175479. 2, 3, 29, 30, 36, 47, 52, 53, 73, 82, 89, 100, 107

[7] E. R. Berlekamp. On decoding binary Bose-Chaudhuri-Hocquenghem codes. *IEEE Transactions on Information Theory*, 11(4):577–80, October 1965. ISSN 0018-9448. doi: 10.1109/TIT.1965.1053810. 32, 37, 38, 41

[8] E. R. Berlekamp. *Algebraic Coding Theory*. New York: McGraw-Hill, 1968. 32, 37, 38, 41

[9] R. Blahut. *Algebraic Codes for Data transmission*. New York: Cambridge University Press, 2003. ix, 5, 15, 18, 23, 38

[10] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3:68–79, March 1959. doi: 10.1016/S0019-9958(60)90287-4. 32

[11] Martin Bossert. *Kanalcodierung*. Stuttgart, Germany: B.G. Teubner, 1991. 5

[12] J. B. Carruthers and S. M. Carroll. Statistical impulse response models for indoor optical wireless channels. *International Journal of Communication Systems*, 18(3):267–284, March 2005. doi: 10.1002/dac.703. 2

[13] F. Chang, K. Onohara, and T. Mizuochi. Forward error correction for 100 G transport networks. *IEEE Communications Magazine*, 48(3):S48–S55, March 2010. ISSN 0163-6804. doi: 10.1109/MCOM.2010.5434378. 2

[14] N. B. Chang. Rate adaptive non-binary LDPC codes with low encoding complexity. In *Proceedings of the Forty Fifth Asilomar Conference on Signals, Systems and Computers*, pages 664 – 668, November 2011. 88

[15] D. Chase. Code combining - a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets. *IEEE Transactions on Communications*, 33(5):385–393, May 1985. ISSN 0090-6778. doi: 10.1109/TCOM.1985.1096314. 89

[16] J. Chen, A. Khisti, D. M. Malioutov, and J. S. Yedidia. Distributed source coding using serially-concatenated-accumulate codes. In *Proceedings of the 2004 IEEE Information Theory Workshop*, pages 209–214, October 2004. doi: 10.1109/ITW.2004.1405301. 131

[17] N. Chen and Z. Yan. Complexity analysis of Reed-Solomon decoding over GF($2^m$) without using syndromes. *EURASIP Journal on Wireless Communication and Networking - Advances in Error Control Coding Techniques*, 2008(9):1–11, May 2008. doi: 10.1155/2008/843634. 82, 84

[18] R. T. Chien. Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes. *IEEE Transactions on Information Theory*, 10:357–363, October 1964. ISSN 0018-9448. doi: 10.1109/TIT.1964.1053699. 32, 37, 38, 41

[19] C. H. Cho, J. J. Won, and H. W. Lee. Performance of hybrid II ARQ schemes using punctured RS code for wireless ATM. *IEE Proceedings–Communications*, 148(4):229 – 233, August 2001. ISSN 1350-2425. doi: 10.1049/ip-com:20010399. 88

[20] T. Cover and J. Thomas. *Elements of Information Theory*. New York: John Wiley & Sons Inc, 1991. ISBN 0471200611. 5, 7, 8

[21] R. H. Deng and M. L. Lin. A type I hybrid ARQ system with adaptive code rates. *IEEE Transactions on Communications*, 43(2/3/4):733–737, February 1995. ISSN 0090-6778. doi: 10.1109/26.380101. 89

[22] D. Divsalar, S. Dolinar, and F. Pollara. Iterative turbo decoder analysis based on density evolution. *IEEE Journal on Selected Areas in Communications*, 19(5):891–907, May 2001. ISSN 0733-8716. doi: 10.1109/49.924873. 51

[23] P. L. Dragotti and M. Gastpar. *Distributed Source Coding: Theory, Algorithms and Applications*. Amsterdam: Academic Press/Elsevier, 2009. 131

[24] E. O. Elliott. Estimates of error rates for codes on burst-noise channels. *The Bell System Technical Journal*, 42:1977–1997, September 1963. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1963.tb00955.x. 3, 67, 101

[25] R.G. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962. ISSN 0096-1000. doi: 10.1109/TIT.1962.1057683. 43

[26] J. Garcia-Frias and Y. Zhao. Compression of correlated binary sources using turbo codes. *IEEE Communications Letters*, 5(10):417–419, October 2001. ISSN 1089-7798. doi: 10.1109/4234.957380. 131

[27] D. Garg and F. Adachi. Application of rates compatible punctured turbo coded hybrid ARQ to MC-CDMA mobile radio. *ETRI Journal*, 26(5):405–412, October 2004. ISSN 2233-7326. doi: 10.4218/etrij.04.0703.0003. 88

[28] G. H. Gho, L. Klak, and J. M. Kahn. Rate-adaptive coding for optical fiber transmission system. *Journal of Lightwave Technology*, 29(2):222–233, January 2011. ISSN 0733-8724. doi: 10.1109/JLT.2010.2099208. 88

[29] E. N. Gilbert. Capacity of a burst-noise channel. *The Bell System Technical Journal*, 39:1253–1265, September 1960. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1960.tb03959.x. 3, 67

[30] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero. Distributed video coding. *Proceedings of the IEEE*, 93(1):71–83, January 2005. ISSN 0018-9219. doi: 10.1109/JPROC.2004.839619. 127

[31] J. Hagenauer. Rate-compatible punctured convolutional codes (RCPC codes) and their application. *IEEE Transactions on Communications*, 36 (4):389 – 400, April 1988. ISSN 0090-6778. doi: 10.1109/26.2763. 88, 89, 90, 91, 100, 101

[32] C. Häger, A. G. I. Amat, H. D. Pfister, and et al.(2015). On parameter optimization for staircase codes. In *Proceedings of the Optical Fiber Communication Conference and Exposition (OFC)*, pages 1–3, March 2015. URL http://publications.lib.chalmers.se/publication/211702. 49

[33] Gerhard Hasslinger and Oliver Hohlfeld. The gilbert-elliott model for packet loss in real time services on the internet. In *In Proceedings of the 14th GI/ITG Conference on Measurement, Modelling and Evaluation of Computer and Communication Systems*, pages 269–283, 2008. 68

[34] Franz Hlawatsch. *Information Theory and Coding*. Wien, Austria: E389 Institute of Telecommunications, Technische University Wien, October 2011. 5, 10, 15, 16, 17, 23, 24, 27, 28, 58, 61

[35] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2:147–56, 1959. URL http://kom.aau.dk/~heb/kurser/NOTER/KOFA02.PDF. 32

[36] International Telecommunication Union (ITU). Interfaces for the optical transport network, 2015. URL http://www.itu.int/rec/T-REC-G.709/. 47

[37] O. Jetlund, G. E. Oien, K. J. Hole, and V. Markhus. Rate-adaptive coding and modulation with LDPC component codes. *European Cooperation in the Field of Scientific and Technical Research*, September 2002. URL http://www.fysel.ntnu.no/projects/beats/Documents/TD-02-108.pdf. 88

[38] Y. Jian, H. D. Pfister, and K. R. Narayanan. Approaching capacity at high rates with iterative hard-decision decoding. *in arXiv:1202.6095v3 [cs.IT] 21 May 2015*, May 2015. ix, 49, 50, 99

[39] J. S. Johnson. Introducing low-density parity-check codes. *Published Internal Technical Report, Department of Electrical and Computer Engineering, University of Newcastle, Australia*, 2000. 43, 64

[40] J. S. Johnson and Steven R. Weller. Resolvable 2-design for regular low-dentity parity-check codes. *IEEE Transactions on Communications*, 51(9): 1413–1419, September 2003. ISSN 0090-6778. doi: 10.1109/TCOMM.2003.816946. 44, 45, 46

[41] J. Justesen. Performance of product codes and related structures with iterated decoding. *IEEE Transactions on Communications*, 59(2):407–415, February 2011. ISSN 0090-6778. doi: 10.1109/TCOMM.2011.121410.090146. 19, 20, 21, 48, 58, 73, 99

[42] J. Kim, W. Hur, A. Ramamoorthy, and S. W. McLaughlin. Design of rate-compatible irregular LDPC codes for incremental redundancy hybrid ARQ

systems. In *Proceedings of the 2006 IEEE International Symposium on Information Theory*, pages 1139 – 1143, July 2006. doi: 10.1109/ISIT.2006.261962. 88

[43] P. Kukieattikool and N. Goertz. Staircase codes for high-rate wireless transmission on burst channel. *IEEE Wireless Communications Letters*, PP(99): 1–1, 2015. ISSN 2162-2337. doi: 10.1109/LWC.2015.2507573. 67

[44] J. Le Boudec. *Performance Evaluation of Computer and Communication Systems*. Lausanne, Switzerland: EPFL Press, 2010. URL `http://perfeval.epfl.ch/`. 54, 79, 80

[45] F. Lehmann and G. M. Maggio. Analysis of the iterative decoding of LDPC and product codes using the Gaussian approximation. *IEEE Transactions on Information Theory*, 49(11):2993–3000, November 2003. ISSN 0018-9448. doi: 10.1109/TIT.2003.819335. 51

[46] S. Lin and D. Costello. *Error Control Coding*. Upper Saddle River, N.J.: Prentice-Hall, 2004. ix, 2, 3, 5, 19, 25, 26, 27, 32, 37, 39, 41, 42, 43, 46, 67, 83, 84, 90, 93, 108

[47] A. D. Liveris, Z. Xiong, and C. N. Georghiades. Compression of binary sources with side information at the decoder using LDPC codes. *IEEE Communications Letters*, 6(10):440–442, October 2002. ISSN 1089-7798. doi: 10.1109/LCOMM.2002.804244. 131

[48] D. M. Mandelbaum. An adaptive-feedback coding scheme using incremental redundancy. *IEEE Transactions on Information Theory*, 20(3):388 – 389, May 1974. ISSN 0018-9448. doi: 10.1109/TIT.1974.1055215. 88

[49] Ivan Martino and Luca Martino. On the variety of linear recurrences and numerical semigroups. *Semigroup Forum*, 88(3):569–574, 2013. ISSN 1432-2137. doi: 10.1007/s00233-013-9551-2. 104

[50] F. Mattoussi. *Design and Optimization of AL-FEC codes: the GLDPC-Staircase Codes*. PhD thesis, Networking and Internet Architecture [cs.NI].,

Universite de Grenoble, 2014. URL https://tel.archives-ouvertes.fr/tel-00969573. 88

[51] R. J. McEliece and L. Swanson. On the decoder error probability for Reed - Solomon codes (corresp.). *IEEE Transactions on Information Theory*, 32(5): 701–703, January 1986. ISSN 0018-9448. doi: 10.1109/TIT.1986.1057212. 56, 73

[52] T. Moon. *Error Correction Coding Mathematical Methods and Algorithms*. Hoboken, N.J.: Willey-Interscience, 2005. 3, 5, 7, 8, 9, 35, 36, 37, 41, 67

[53] M. Mushkin and I. Bar-David. Capacity and coding for the Gilbert-Elliott channels. *IEEE Transactions on Information Theory*, 33(6):1277–1290, November 1989. ISSN 0018-9448. doi: 10.1109/18.45284. 70, 71

[54] Claudia Osmann. *Bewertung von Codierverfahren fuer einen Stoerungssicheren Datentransfer*. Dissertation, Mathematik der Gerhard-Mercator-Universitaet-Gesamthochschule-Duisburg, Germany, 1999. 69

[55] K. Oteng-Amoako and S. Nooshabadi. Asymmetric rate compatible turbo codes in hybrid automatic repeat request schemes. *IEE Proceedings–Communications*, 153:603 – 610, October 2006. ISSN 1350-2425. 88

[56] C. Pimentel and I. F. Blake. Enumeration of Markov chains and burst error statistics for finite state channel models. *IEEE Transactions on Vehicular Technology*, 48(2):415–428, March 1999. ISSN 0018-9545. doi: 10.1109/25.752565. 102, 103, 104, 106

[57] B. Pittel, J. Spencer, and N. Woemald. Sudden emergence of a giant k-core in a random graph. *Journal of Combinatorial Theory, Series B*, 67(1): 111–151, May 1996. doi: 10.1006/jctb.1996.0036. 20

[58] S. S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (DISCUS): design and construction. *IEEE Transactions on Information Theory*, 49(3):626–643, March 2003. ISSN 0018-9448. doi: 10.1109/TIT.2002.808103. 131

[59] R. Puri, A. Majumdar, and K. Ramchandran. PRISM: A video coding paradigm with motion estimation at the decoder. *IEEE Transactions on Image Processing*, 16(10):2436–2448, October 2007. ISSN 1057-7149. doi: 10.1109/TIP.2007.904949. 127

[60] H. A. Rajab and M. D. Yucel. The probability of error performance of Reed-Solomon codes over q-ary nonsymmetric channels. In *Proceedings of the 6th Mediterranean Electrotechnical Conference*, volume 1, pages 610–613, May 1991. doi: 10.1109/MELCON.1991.161912. 97, 98

[61] J. Ramsey. Realization of optimum interleavers. *IEEE Transactions on Information Theory*, IT-16(3):338–345, May 1970. ISSN 0018-9448. doi: TIT.1970.1054443. 76

[62] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, June 1960. doi: 10.1137/0108018. 3, 38, 67

[63] M. Rezaeian. Computation of capacity for Gilbert-Elliott channels, using a statistical method. In *Proceedings of the 6th Australian Workshop on Communication Theory*, pages 56–61, February 2005. doi: 10.1109/AUSCTW.2005.1624226. 72, 73

[64] A. Rosa. Combinatorial design theory. In *Proceedings of the Algebraic, Topological and Complexity Aspects of Graph Covers & Winter School in Harmonic Functions on Graphs and Combinatorial Designs*, Sepetn'a, Czech Republic, January 2014. URL `http://kam.mff.cuni.cz/~atcagc14/materialy/DesignTheoryNotes.pdf`. 45

[65] A. Roumy, K. Lajnef, and C. Guillemot. Rate-adaptive turbo-syndrome scheme for Slepian-Wolf coding. In *Proceedings of the Forty-First Asilomar Conference on Signals, Systems and Computers (ACSSC)*, pages 545–549, November 2007. doi: 10.1109/ACSSC.2007.4487272. 131

[66] M. Salmistraro, K. J. Larsen, and S. Forchhammer. Rate-adaptive BCH codes for distributed source coding. *EURASIP Journal on Advances in Sig-*

*nal Processing*, 2013(166), 2013. doi: 10.1186/1687-6180-2013-166. xiii, 132, 138, 140, 142

[67] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:623–656, October 1948. 5, 128

[68] A. Shiozaki. Adaptive type-II hybrid broadcast ARQ system. *IEEE Transactions on Communications*, 44(4):420–422, April 1996. ISSN 0090-6778. doi: 10.1109/26.489086. 89

[69] J. K. Skwirzynski. *Communication Systems and Random Process Theory*. Alphen aan den Rijn: Sijthoff & Noordhoff, 1978. 128

[70] David Slepian and Jack K. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, IT-19(4):471–480, July 1973. doi: 10.1109/TIT.1973.1055037. xii, 127, 128, 129

[71] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A method for solving key equation for decoding Goppa codes. *Information and Control*, 27:87–99, January 1975. doi: 10.1016/S0019-9958(75)90090-X. 38

[72] R.M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981. ISSN 0018-9448. doi: 10.1109/TIT.1981.1056404. 43

[73] D. Torrieri. Information-bit, information-symbol, and decoded-symbol error rates for linear block codes. *IEEE Transactions on Communications*, 36(5): 613–617, May 1988. ISSN 0090-6778. doi: 10.1109/26.1477. 48

[74] D.J. Torrieri. The information-bit error rate for block codes. *IEEE Transactions on Communications*, 32(4):474–476, Apr 1984. ISSN 0090-6778. doi: 10.1109/TCOM.1984.1096082. 97

[75] D. Truhhachev, L. Zhang, and F. R. Kschischang. Information transfer bounds on iterative thresholds of Staircase codes. Paper presented at the 2015 Information Theory and Applications workshop (ITA), Scripps Seaside Forum, La Jolla, USA, February 2015. URL `http://ita.ucsd.edu/workshop/15/files/paper/paper_1258.pdf`. 48, 99

[76] A. Tychopoulos, O. Koufopavlou, and I. Tomkos. FEC in optical communications. *IEEE Circuit and Devices Magazine*, pages 79–86, November 2006. 11

[77] International Telecommunication Union. Forward error correction for high bit-rate DWDM submarine systems. *Recommendation G 975.1: Series G Transmission Systems and Media, Digital Systems and Networks*, February 2004. URL https://www.itu.int/rec/T-REC-G.975.1-200402-I/en. 29

[78] D. Varodayan, A. Aaron, and B. Girod. Rate-adaptive distributed source coding using low-density parity-check codes. In *Proceedings of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*, pages 1203–1207, 2005. doi: 10.1109/ACSSC.2005.1599952. 132, 138

[79] C. X. Wang and M. Paetzold. A novel generative model for burst error characterization in Rayleigh fading channels. In *Proceedings of the 14th IEEE on Personal, Indoor and Mobile Radio Communications (PIMRC 2003)*, volume 1, pages 960–964, September 2003. doi: 10.1109/PIMRC.2003.1264416. 3

[80] J. K. Wolf and B. M. Kurkoski. Slepian-Wolf coding. *Scholarpedia*, 3(11): 6789, 2008. ISSN 1941-6016. doi: 10.4249/scholarpedia.6789. 130

[81] J.K. Wolf, A. Michelson, and A.H. Levesque. On the probability of undetected error for linear block codes. *IEEE Transactions on Communications*, 30(2):317–325, Feb 1982. ISSN 0090-6778. doi: 10.1109/TCOM.1982.1095473. 61

[82] A. D. Wyner. Recent results in the Shannon theory. *IEEE Transactions on Information Theory*, 20(1):2–10, January 1974. ISSN 0018-9448. doi: 10.1109/TIT.1974.1055171. 131

[83] A. D. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, January 1973. doi: 10.1109/TIT.1976.1055508. 127

[84] Z. Xiong, A. D. Liveris, and S. Cheng. Distributed source coding for sensor networks. *IEEE Signal Processing Magazine*, pages 80–94, September 2004. ISSN 1053-5888. doi: 10.1109/MSP.2004.1328091. xii, 127, 130

[85] R. D. Yates and D. J. Goodman. *Probability and Stochastic Processes*. John Wiley & Sons, Inc., 2005. 20, 69

[86] Chaehag. Yi and J. H. Lee. Interleaving and decoding scheme for a product code for a mobile data communication. *IEEE Transactions on Communications*, 45(2):144–147, February 1997. ISSN 0090-6778. doi: 10.1109/26.554359. 75, 76

[87] X. Yu, K. Sun, and D. Yuan. Comparative analysis on HARQ with turbo codes in Rician fading channel with low Rician factor. In *Proceedings of the 12th IEEE International Conference on Communication Technology (ICCT)*, pages 342 – 345, November 2010. doi: 10.1109/ICCT.2010.5689216. 88

[88] Yequn Zhang and I.B. Djordjevic. Staircase rate-adaptive ldpc-coded modulation for high-speed intelligent optical transmission. In *Proceedings of the Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, March 2014. doi: 10.1364/OFC.2014.M3A.6. 64, 88