Codes on Graphs and More

Hug, Florian

2012

Codes on Graphs and More

Florian Hug

LUND UNIVERSITY

Doctoral Dissertation
Information Theory
Lund, May 2012

Florian Hug
Department of Electrical and Information Technology
Information Theory
Lund University
P.O. Box 118, 221 00 Lund, Sweden

Series of licentiate and doctoral dissertations
ISSN 1654-790X; No. 38
ISBN 978-91-7473-284-9

Typeset in Palatino and Helvetica using LaTeX 2_ε.
Printed in Sweden by Tryckeriet i E-huset, Lund University, Lund.

Abstract

Modern communication systems strive to achieve reliable and efficient information transmission and storage with affordable complexity. Hence, efficient low-complexity channel codes providing low probabilities for erroneous receptions are needed. Interpreting codes as graphs and graphs as codes opens new perspectives for constructing such channel codes. Low-density parity-check (LDPC) codes are one of the most recent examples of codes defined on graphs, providing a better bit error probability than other block codes, given the same decoding complexity.

After an introduction to coding theory, different graphical representations for channel codes are reviewed. Based on ideas from graph theory, new algorithms are introduced to iteratively search for LDPC block codes with large girth and to determine their minimum distance. In particular, new LDPC block codes of different rates and with girth up to 24 are presented. Woven convolutional codes are introduced as a generalization of graph-based codes and an asymptotic bound on their free distance, namely, the Costello lower bound, is proven. Moreover, promising examples of woven convolutional codes are given, including a rate $5/20$ code with overall constraint length 67 and free distance 120.

The remaining part of this dissertation focuses on basic properties of convolutional codes. First, a recurrent equation to determine a closed form expression of the exact decoding bit error probability for convolutional codes is presented. The obtained closed form expression is evaluated for various realizations of encoders, including rate $1/2$ and $2/3$ encoders, of as many as 16 states. Moreover, MacWilliams-type identities are revisited and a recursion for sequences of spectra of truncated as well as tailbitten convolutional codes and their duals is derived. Finally, the dissertation is concluded with exhaustive searches for convolutional codes of various rates with either optimum free distance or optimum distance profile, extending previously published results.

Contents

Contents v

Preface ix

Acknowledgments xi

1 Introduction 1

1.1 A Basic Digital Communication Model 2

1.2 Some Channel Models 4

1.2.1 The Binary Symmetric Channel 4

1.2.2 The Additive White Gaussian Noise Channel 5

1.3 Channel Coding 8

1.3.1 Block Codes 8

1.3.2 Convolutional Codes 13

1.3.3 Optimal Decoding Principles 23

1.4 Dissertation Outline 28

2 Graphs, Codes, and Codes on Graphs 29

2.1 Trees and Trellises for Convolutional Codes 30

2.2 Trees and Trellises for Linear Block Codes 36

2.3 The Viterbi Algorithm 42

2.4 The BEAST 46

2.4.1 Finding the Weight Spectrum 46

2.4.2 Finding the Viterbi Spectrum 50

2.4.3 Maximum-Likelihood Decoding 51
2.4.4 Determine the Metric Thresholds 53
2.5 Low-Density Parity-Check Codes and Tanner Graphs 55
2.6 The Belief Propagation Algorithm 59

3 Voltage Graph-Based QC LDPC Block Codes with Large Girth 65

3.1 Quasi-Cyclic LDPC Block Codes 66
3.2 Base Matrices, Voltages, and their Graphs 69
3.3 Bounds on the Girth and the Minimum Distance 72
3.4 Searching for QC LDPC Block Codes with Large Girth 76
3.4.1 Step I: Creating a Tree Structure 77
3.4.2 Step II: Searching for a Suitable Voltage Assignment . . . 79
3.5 Minimum Distance of QC LDPC Block Codes 83
3.6 All-one Based QC LDPC Block Codes 84
3.7 Alternative Constructions 86
3.7.1 Steiner Triple Systems Based QC LDPC Block Codes . . . 91
3.7.2 Iterative QC LDPC Block Codes 94
3.7.3 Double-Hamming Based QC LDPC Block Codes 95
3.7.4 Binomial QC LDPC Block Codes 99
3.8 Case Study: IEEE 802.16 WiMAX 100

4 Woven Graph Codes 105

4.1 Graph-based Block Codes with Constituent Codes 106
4.2 Woven Graph Codes 109
4.3 Asymptotic Bound on the Free Distance of Woven Convolutional Graph Codes 111
4.4 Examples 117
4.5 Simulation Results 119

5 A Closed Form Expression for the Exact Bit Error Probability 121

5.1 Expressing the Bit Error Probability Using the Average Information Weights 122
5.2 Computing the Vector of Average Information Weights 128
5.3 Solving the Recurrent Equation 132

5.3.1 Determining the Limit of a Power Series 133
5.3.2 Deriving a Closed Form Expression 135
5.4 Additional Examples 136

6 MacWilliams-type Identities for Convolutional Codes 143
6.1 Weakly Equivalent Matrices 144
6.2 Block Spectra for Zero-Tail Terminated and Truncated Convolutional Codes 148
6.3 MacWilliams-Type Identities 148
6.4 Infinite Sequences of Spectra 155

7 Optimum and Near-Optimum Convolutional Codes 161
7.1 Distance Properties 162
7.1.1 Column Distance 162
7.1.2 Row Distance 164
7.1.3 Smith Form Decomposition 165
7.2 Optimum Free Distance Convolutional Codes 166
7.3 Optimum Distance Profile Convolutional Codes 171

Acronyms 175

References 177

Preface

This dissertation summarizes the results of my research work at the Department of Electrical and Information Technology at Lund University in Sweden. Most of the included material has appeared in the following journal papers or conference contributions:

- I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »A Closed Form Expression for the Exact Bit Error Probability for Viterbi Decoding of Convolutional Codes,« accepted for publication in *IEEE Transactions on Information Theory*, vol. 58, 2012.
- I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, »Searching for Voltage Graph-Based LDPC Tailbiting Codes with Large Girth,« *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2265 – 2279, April 2012.
- I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »A Note on Duality and MacWilliams-type Identities for Convolutional Codes,« *Problems of Information Transmission*, vol. 48, no. 1, April 2012.
- D. Johnsson, F. Bjärkeson, M. Hell and F. Hug, »Searching for New Convolutional Codes Using the Cell Broadband Engine Architecture,« *IEEE Communications Letters*, vol. 15, no. 5, pp. 560 – 562, May 2011.
- I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »Woven Convolutional Graph Codes with Large Free Distances,« *Problems of Information Transmission*, vol. 47, no. 1, pp. 3 – 18, January 2011.
- F. Hug, I. E. Bocharova, R. Johannesson, and B. D. Kudryashov, »A Rate $R = 5/20$ Hypergraph-Based Woven Convolutional Code with Free Distance 120,« *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1618–1623, April 2010.

- I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, »Some Voltage Graph-Based LDPC Tailbiting Codes with Large Girth,« in *Proc. IEEE International Symposium on Information Theory (ISIT'11)*, pp. 732–736, St. Petersburg, Russia, July 31 – August 8, 2011.
- I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »Double-Hamming Based QC LDPC Codes with Large Minimum Distance,« in *Proc. IEEE International Symposium on Information Theory (ISIT'11)*, pp. 923–927, St. Petersburg, Russia, July 31 – August 8, 2011.
- I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »Another Look at the Exact Bit Error Probability for Viterbi Decoding of Convolutional Codes,« in *Proc. International Mathematical Conference »50 Years of IPPI«*, Moscow, Russia, July 25 – 29, 2011.
- I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »On the Exact Bit Error Probability for Viterbi Decoding of Convolutional Codes,« in *Proc. Information Theory and Applications Workshop (ITA'11)*, San Diego, USA, February 2 – 6, 2011.
- I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, »New Low-Density Parity-Check Codes with Large Girth Based on Hypergraphs,« in *Proc. IEEE International Symposium on Information Theory (ISIT'10)*, pp. 819–823, Austin, USA, June 13 – 18, 2010.
- I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »On Weight Enumerators and MacWilliams Identity for Convolutional Codes,« in *Proc. Information Theory and Applications Workshop (ITA'10)*, San Diego, USA, January 31 – February 05, 2010.

I have also co-authored the following papers not included in this dissertation:

- F. Hug, and F. Rusek, »The BEAST for Maximum-Likelihood Detection in Non-Coherent MIMO Wireless Systems,« in *Proc. IEEE International Conference on Communications (ICC'10)*, Cape Town, South Africa, May 23 – 27, 2010.
- F. Hug, »Constructing Error-Correcting Codes with Huge Distances,« in *Partnership for Advanced Computing in Europe (PRACE) code porting workshop,* (invited talk) Linköping, Sweden, October 13 – 14, 2009.
- F. Hug, I. E. Bocharova, R. Johannesson, and B. D. Kudryashov, »Searching for High-Rate Convolutional Codes via Binary Syndrome Trellises,« in *Proc. IEEE International Symposium on Information Theory (ISIT'09)*, pp. 1358 – 1362, Seoul, South Korea, June 28 – July 3, 2009.
- V. V. Zyablov, F. Hug, and R. Johannesson, »Chained Gallager Codes,« in *Proc. International Symposium on Problems of Redundancy in Information and Control Systems*, St. Petersburg, Russia, May 26 – 30, 2009.

The research work included in this dissertation is supported in part by the Swedish Research Council through Grant 621-2007-6281 and 621-2010-5847.

Acknowledgments

A lot of people have contributed to this dissertation in various ways. I would like to take this opportunity to express my thankfulness to all of them. In particular, I would like to mention a few:

First of all, my deepest gratitude goes to my supervisor Rolf Johannesson for sharing all this knowledge and experience with me. Our long lasting discussions covering both scientific and personal matters have been invaluable to me. His encouragement and support as well as his optimistic attitude towards life in all aspects have influenced me greatly. In the end, Rolf is much more than »just« a supervisor, he's a really good friend and a person that I look up to and admire deeply.

During my time in Sweden, I had the honor of working together with Irina Bocharova and Boris Kudryashov. I highly appreciate their kindness and profound knowledge they shared with me and for always finding the time for discussions; personally, via mail, or using instant messaging. Moreover, thanks a lot for being excellent travel guides during our visit in St. Petersburg. I am grateful to Kamil Zigangirov for his support and the knowledge he shared with me during several interesting discussions.

My thanks go to all my colleagues and fellow PhD students at the Department of Electrical and Information Technology who created a friendly and inspiring work environment. Especially, I would like to express my gratitude to Martin Å. and Fredrik for proofreading some chapters of this dissertation. Also, and in no order of importance, I would like to thank my friends from the »extended« 3rd floor, Martin H., Paul, Dzevdan, Adnan, and Anders for introducing me to their Swedish social life and for our interesting lunch and coffee break discussions. Furthermore, I also would like to express my thanks and appreciation to the technical and administrative staff at the department.

Being part of the National Graduate School in Scientific Computing as well as having the possibilities to extensively use the resources at Lunarc, the Center for Scientific and Technical Computing at Lund University, thanks to generous grants from the Swedish National Infrastructure for Computing is highly acknowledged.

I am especially grateful to all my other friends in Sweden, who made sure that I did not spend all my time in front of the computer screen. In particular, I would like to mention my »lovely« neighbor Inga who always provided me with freshly baked buns & cookies, and Feffe for taking me out to learn proper Swedish. My thanks also go to Elinor for nightlong discussions and to all my other salsa and bachata friends, in particular, Very, Nata, Philip, and Kicki for joyful and memorable moments we shared on the dance floors.

Last, but not least, I am utmost grateful to my parents and my sister Catherina for their invaluable support, encouragement and unconditional love. But most importantly, my deepest thanks go to my love Daniela with whom I am looking forward to start a new chapter of our lives.

Florian Hug
Lund, May 2012

1 Introduction

Being able to communicate over long distances, wherever we are, and at any time, has become an essential part in our society. Only a few decades ago, fixed wired analog telephones were the only means of communication. With the invention of mobile phones, the situation changed completely and nowadays everybody is reachable worldwide at any given moment. At the same time, the importance of fixed wired telephony (even though digital) decreased, which is no longer solely sufficient for daily communications. Today, mobile phones are the most widespread communication technology in our society.

The ideas which, among others, made this development possible reach back to the year 1948 when Claude E. Shannon published his landmark paper »A Mathematical Theory of Communication« [Sha48]. Using probabilistic measures, Shannon described a new digital *communication model* and derived its limits for *reliable communication*, which laid the foundation for information and coding theory.

Based on Shannon's ideas, new concepts and techniques were developed, striving to practically achieve the previously outlined limits. Examples include digital cellular wireless networks, like the most popular second generation Global System for Mobile Communications (GSM) as well as its third- and fourth-generation successors, Universal Mobile Telecommunications System (UMTS) and Long Term Evolution (LTE), respectively, or Digital Subscriber Line (DSL) technology, which is still the most widespread Internet access technology. Besides traditional communication networks, new storage systems such as hard disks (HDDs) or solid state flash drives (SSDs) in computers, as well as compact disks (CDs), digital versatile disks (DVDs), or blue-ray disks (BDs) use coding for *reliable* and *efficient storage* of information.

In Shannon's communication model, all information, being either transmitted to its receiver or stored for later usage, propagates through some kind of *channel*. Such a channel depends largely on the underlying application and can represent, for example, a coaxial cable for a television signal, air in wireless communications, or electronic circuits and imperfect memories in case of HDDs or SSDs. If the receiver is not able to recover the original information sequence from its received distorted version, we say that an *error event* has occurred.

In order to provide reliable and efficient information transmission and storage, *channel coding*, also known as *error control coding*, is commonly used to protect the information sequences from error events during transmission over noisy channels or storage in imperfect memories. Channel coding adds additional redundant information to the sequences being transmitted or stored, which can be used to detect errors events and, more importantly, recover the original information sequences. In general, this can be achieved by two different kinds of channel codes: *block codes*, where a constant ratio of redundancy is added separately to each information sequence, and *convolutional codes*, where redundant information is added continuously using a convolution with consecutive information sequences.

In Section 1.1, Shannon's general model of a communication system is described in more detail. Different models for the transmission channel used within this dissertation are discussed in Section 1.2, while Section 1.3 focuses on the basic principles of block and convolutional codes, respectively. These sections are largely based on [Bos98], [JZ99], [LC04], and [PS08].

1.1 A BASIC DIGITAL COMMUNICATION MODEL

Figure 1.1 illustrates a block diagram of a basic *digital* communication system. Using sampling and quantizing, every analog signal can be approximated by a sequence of digital symbols with as high accuracy as required. Thus, the output of the *source*, even though representing any kind of analog or digital source signal like speech, audio, video or data, can be assumed to be an entirely digital sequence.

Using Shannon's *separation principle*, the processing of any (infinite) sequence prior to transmission can, without loss of optimality, be separated into two individual tasks: source encoding and channel encoding.

During *source encoding*, also known as *data compression*, a digital sequence is represented with as few binary digits (bits) as possible. Thereby, the natural redundancy of the original message is removed, and the bits at the output are ideally equiprobable, but independently distributed. As source encoding is beyond the scope of this dissertation, we combine the source with the

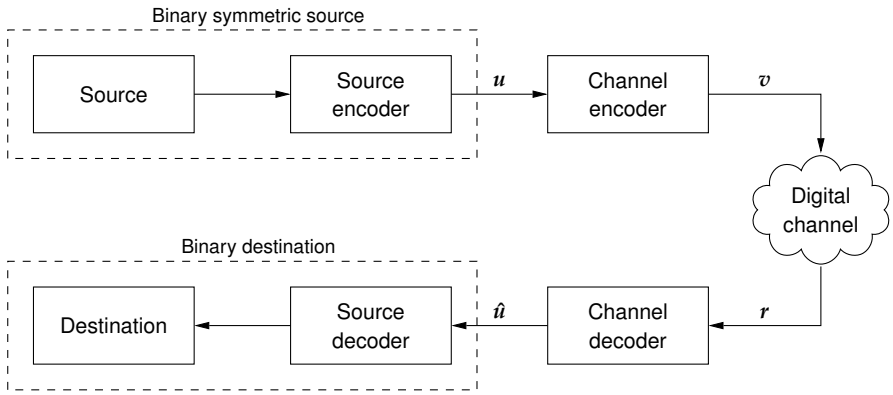


Figure 1.1: A model of a basic communication system.

source encoder, and refer to this block as a *binary symmetric source* (BSS). Such a BSS emits a (time-discrete) binary, random sequence $\boldsymbol{u}$, the so-called *information sequence*, consisting of equiprobable, but independently distributed binary symbols. Similarly, their counterparts at the receiver side, the *source decoder* as well as the *destination* can be combined to a single *binary destination*; see Figure 1.1.

Propagation conditions through the channel usually violate the requirements for reliable communication, as the probability for erroneous reception of the transmitted information sequence is unacceptably high. Using a *channel encoder*, artificial redundancy is added to the information sequence $\boldsymbol{u}$ in a controlled manner, forming the *code sequence* $\boldsymbol{v}$, also known as the *codeword*. The *channel decoder* at the receiver exploits this artificial redundancy to correct most of the introduced errors; reducing the overall error probability.

In many applications, the underlying fundamental channel is of analog nature. Thus, inside the *digital channel*, tuples of code symbols of the code sequence $\boldsymbol{v}$ are mapped to a modulation alphabet, converted to analog waveforms, modulated, and are finally transmitted over the analog (waveform) channel. The received analog waveforms, corrupted by noise (errors) during transmission, are demodulated and possibly quantized, before being emitted as a sequence of observed values, the so-called *received sequence* $\boldsymbol{r}$.

Based on the received sequence $\boldsymbol{r}$ and the structural knowledge of the artificial redundancy added by the channel encoder, the *channel decoder* outputs a decision on the information sequence $\hat{\boldsymbol{u}}$. If the decision $\hat{\boldsymbol{u}}$ coincides with the information sequence $\boldsymbol{u}$, the transmission has been successful; otherwise an uncorrectable error event has occurred. Finally, the sequence $\hat{\boldsymbol{u}}$ is forwarded to the *source decoder* which reconstructs the original information message and delivers it to its *destination*.

1.2 SOME CHANNEL MODELS

1.2.1 THE BINARY SYMMETRIC CHANNEL

One of the most fundamental and purely theoretical channels used within coding theory is the *binary symmetric channel* (BSC) as illustrated in Figure 1.2.

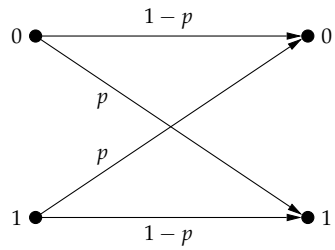


Figure 1.2: BSC with crossover probability p.

A binary input symbol passes the BSC unchanged with probability $1-p$, while the binary output symbol differs from its input with probability p. Thus, a BSC is completely characterized by the parameter p, its so-called *crossover probability*.

As both the input and output symbols of a BSC are binary, the received sequence $\boldsymbol{r}$ can be expressed as

$$\boldsymbol{r} = \boldsymbol{v} \oplus \boldsymbol{e} \tag{1.1}$$

where $\oplus$ denotes the modulo-2 addition, $\boldsymbol{v}$ is a code sequence and the *error pattern* $\boldsymbol{e}$ is a random sequence, whose elements are 0 with probability $1-p$, and 1 with probability p.

According to Shannon's *channel coding theorem* [Sha48], every discrete-input, memoryless channel (DMC) is characterized by a single parameter C, its *capacity*. Assume that the information sequence $\boldsymbol{u}$ is encoded into the code sequence $\boldsymbol{v}$ of length N, and transmitted over a channel at rate R *bits per channel use*. Then it is possible (for sufficiently large N) to achieve an *arbitrarily low* (block) error probability, as long as the transmission rate satisfies $R \leq C$. On the other hand, if $R > C$, we can not reach such an arbitrarily low error probability.

For example, in case of a BSC with crossover probability p, the channel capacity is

$$C = 1 - h(p) \tag{1.2}$$

where

$$h(x) = -x\log_2(x) - (1-x)\log_2(1-x) \tag{1.3}$$

is the binary entropy function.

1.2.2 THE ADDITIVE WHITE GAUSSIAN NOISE CHANNEL

Another more practical channel model for the transmission of analog waveforms is the (binary input) *additive white Gaussian noise* (AWGN) channel. Consider a binary code sequence $\boldsymbol{v}$, whose code symbols $v_i \in \{0,1\}$, $i = 0, 1, \ldots$, are mapped to symbols from a modulation alphabet $\mathcal{M}$ with average transmit energy E_s per modulation symbol. Then the corresponding modulation signal $s(t)$ can be expressed as

$$s(t) = \sum_k x_k p(t - kT) \qquad k = 0, 1, \ldots \tag{1.4}$$

where T denotes the symbol interval, $x_k \in \mathcal{M}$, and $p(t)$ is a real-valued transmit pulse, which is orthogonal to itself under T-shifts and has unit-energy.

When transmitting the modulation signal $s(t)$ over a (memoryless) analog waveform channel, a common assumption is disturbance by additive white Gaussian noise. Such noise can be modeled as a random noise process with a constant power spectral density (PSD) for all frequencies, where the two-sided PSD is denoted $N_0/2$. Using an optimum receiver, that is, a matched-filter for the transmit pulse $p(t)$, the received sequence $\boldsymbol{r}$ can be expressed by

$$\boldsymbol{r} = \boldsymbol{x} + \boldsymbol{n} \tag{1.5}$$

where $\boldsymbol{x}$ is the sequence of modulation symbols $x_k \in \mathcal{M}$ and $\boldsymbol{n}$ is a sequence of independently drawn Gaussian random variables with zero mean and variance $\sigma^2 = N_0/2$. Given $|\mathcal{M}|$ different modulation symbols x_k together with binary code symbols v_i, the modulation rate follows as $R_\mathrm{m} = \log_2 |\mathcal{M}|$ code symbols per channel use.

Throughout this dissertation, we will assume binary phase-shift keying (BPSK) with the modulation alphabet $\mathcal{M} = \left\{-\sqrt{E_\mathrm{s}}, \sqrt{E_\mathrm{s}}\right\}$. Then, every binary code symbol $v_t \in \{0,1\}$, at time instant t, is mapped to a modulation symbol $x_t \in \mathcal{M}$ according to

$$x_t = (1 - 2v_t)\sqrt{E_\mathrm{s}} \tag{1.6}$$

and thus the corresponding received symbol r_t can be expressed as

$$r_t = x_t + n_t = (1 - 2v_t)\sqrt{E_\mathrm{s}} + n_t \tag{1.7}$$

where n_t denotes an independently drawn Gaussian random variable with zero mean and variance $\sigma^2 = N_0/2$ at time instant t. Hence, the corresponding transition probability density function (PDF) for an AWGN channel is

$$p\left(r_t \mid x_t\right) = \frac{1}{\sqrt{\pi N_0}} e^{-(r_t - x_t)^2/N_0} \tag{1.8}$$

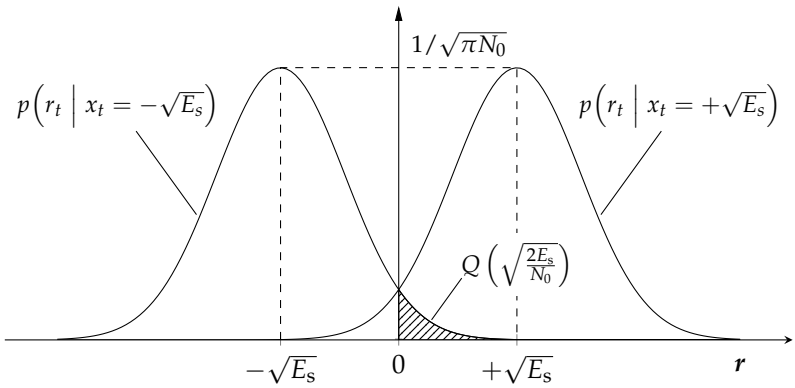


Figure 1.3: Conditional PDF for the AWGN channel using BPSK modulation.

Alternatively, using (1.6), we obtain

$$\begin{cases} p\left(r_t \mid v_t = 0\right) = \dfrac{1}{\sqrt{\pi N_0}} e^{-(r_t - \sqrt{E_s})^2 / N_0} \\ p\left(r_t \mid v_t = 1\right) = \dfrac{1}{\sqrt{\pi N_0}} e^{-(r_t + \sqrt{E_s})^2 / N_0} \end{cases} \tag{1.9}$$

which defines the likelihood of the received symbol r_t, given that the binary code symbol v_t was transmitted. In Figure 1.3 the conditional PDFs for an AWGN channel using BPSK modulation are illustrated.

By quantizing the received symbol r_t to be either positive or negative, an AWGN channel is degraded to a BSC (cf. Subsection 1.2.1). The corresponding crossover probability p for such a channel is indicated by the shaded area in Figure 1.3 and follows as

$$p = P\left(r_t > 0 \;\middle|\; x_t = -\sqrt{E_s} \right) = P\left(r_t > 0 \mid v_t = 1 \right) = Q\left(\sqrt{\frac{2E_s}{N_0}}\right) \tag{1.10}$$

where $Q(x)$ is the *complementary Gaussian distribution function*

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-y^2/2}\, \mathrm{d}y = \frac{1}{2} \operatorname{erfc}\left(\frac{x}{\sqrt{2}}\right) \tag{1.11}$$

Using the »mixed form« of Bayes' rule [WJ65], the *log-likelihood* of the binary code symbol v_t given the received symbol r_t can be expressed as

$$L\left(v_t \mid r_t\right) = \ln \frac{P\left(v_t = 0 \mid r_t\right)}{P\left(v_t = 1 \mid r_t\right)} = \ln \frac{p\left(r_t \mid v_t = 0\right) P\left(v_t = 0\right)}{p\left(r_t \mid v_t = 1\right) P\left(v_t = 1\right)} \tag{1.12}$$

If the code symbol v_t is *a priori* equiprobable distributed, then its log-likelihood value for an AWGN channel follows by combining (1.9) and (1.12) as

$$L\left(v_t \mid r_t\right) = \ln \frac{\frac{1}{\sqrt{\pi N_0}} e^{-(r_t - \sqrt{E_s})^2 / N_0}}{\frac{1}{\sqrt{\pi N_0}} e^{-(r_t + \sqrt{E_s})^2 / N_0}} = \frac{4 r_t \sqrt{E_s}}{N_0} \tag{1.13}$$

Applying typical sequences and random coding, the capacity of a real-valued AWGN channel with signaling energy E_s and two-sided PSD $N_0/2$ can be shown to be

$$C = \frac{1}{2} \log_2 \left(1 + \frac{2E_s}{N_0}\right) \tag{1.14}$$

In theory, the capacity C is achievable with a continuous Gaussian modulation alphabet $\mathcal{M}$. In practical applications, however, the modulation alphabet $\mathcal{M}$ is of finite size, and hence the capacity value given by (1.14) can not be reached.

In the area of channel coding it is often convenient to replace the signal-to-noise ratio per channel use E_s/N_0 by the signal-to-noise ratio (SNR) per information bit E_b/N_0, where $E_s = RE_b$ and the rate R denotes the number of bits transmitted per channel use. Note that within this dissertation, SNR always refers to the signal-to-noise ratio per information bit E_b/N_0.

As mentioned previously, the rate R for reliable communication must satisfy $R \leq C$, and thus

$$R \leq \frac{1}{2} \log_2 \left(1 + \frac{2RE_b}{N_0}\right) \tag{1.15}$$

Solving for E_b/N_0 yields

$$\frac{E_b}{N_0} \geq \frac{2^{2R} - 1}{2R} \tag{1.16}$$

where the right side is an increasing function of the rate R. By taking the limit $R \to 0$, we obtain the *Shannon limit*

$$\frac{E_b}{N_0} \geq \ln 2 \approx 0.693 \quad \left[-1.6\,\text{dB}\right] \tag{1.17}$$

which specifies the minimal required SNR for reliable communication over a real-valued AWGN channel at any rate $R > 0$ in presence of noise.

For a more practical communication channel like the rate $R = 1/2$, binary input AWGN channel, the capacity is given by

$$\begin{aligned} C = &\frac{1}{2} \int_{-\infty}^{\infty} p\left(r_t \mid v_t = 0\right) \log_2 \frac{p\left(r_t \mid v_t = 0\right)}{p\left(r\right)} \, \mathrm{d}r \\ &+ \frac{1}{2} \int_{-\infty}^{\infty} p\left(r_t \mid v_t = 1\right) \log_2 \frac{p\left(r_t \mid v_t = 1\right)}{p\left(r\right)} \, \mathrm{d}r = 0.188 \text{ dB} \end{aligned} \tag{1.18}$$

Even though Shannon's channel coding theorem specifies the limit for a reliable communication, it does not specify any practical coding scheme to achieve an arbitrarily low error probability at rates close to capacity. Instead, the design of practical coding schemes approaching the Shannon limit has been a topic of active research, leading to code designs such as turbo codes or low-density parity-check codes, which shall be discussed in Section 2.5.

1.3 CHANNEL CODING

1.3.1 BLOCK CODES

Consider the stream of binary symbols $\boldsymbol{u}$ at the output of a BSS, parsed into *blocks* of K information bits each. The binary K-tuple $\boldsymbol{u} = (u_0\, u_1 \ldots u_{K-1})$ with $u_i \in \{0, 1\}$, $i = 0, 1, \ldots, K-1$, is called an *information sequence*.

A *block encoder* maps every information sequence of length K to a distinct N-tuple $\boldsymbol{v} = (v_0\, v_1 \ldots v_{N-1})$, a so-called codeword of length N. Each of the $M = 2^K$ different codewords consists of N code symbols v_i, $i = 0, 1, \ldots, N-1$, in general taken from a q-ary alphabet $v_i \in \{0, 1, \ldots, q-1\}$. The set of all M codewords is denoted an (N, K) *block code* $\mathcal{B}$. In the following all block codes will be restricted to be *binary*, that is, $q = 2$ and thus $N \geq K$.

The difference $N - K \geq 0$ defines how much redundancy is added to protect the codewords during transmission. Hence, the *code rate* R_c is given as the ratio

$$R_\mathrm{c} = \frac{\log_2 M}{N} = \frac{K}{N} \tag{1.19}$$

and satisfies $0 < R_\mathrm{c} \leq 1$. Combined with the modulation rate R_m (cf. Subsection 1.2.2), the transmission rate R of the communication system follows as $R = R_\mathrm{c}\, R_\mathrm{m} = R_\mathrm{c} \log_2 |\mathcal{M}|$ bits per channel use. Since BPSK signaling is assumed within this dissertation, it follows that $R_\mathrm{m} = 1$, and thus R denotes both the transmission rate as well as the code rate.

Moreover, as a block code $\mathcal{B}$ is only defined via its set of M distinct codewords, there exist in total $M \cdot (M-1) \cdots 1 = M!$ different bijective mappings, that is, $M!$ different block encoders.

LINEAR BLOCK CODES

An (N, K) *linear* binary block code $\mathcal{B}$ is defined as a K-dimensional subspace of the N-dimensional vector space $\mathbb{F}_2^N$, where $\mathbb{F}_2$ denotes the binary (Galois) field. In other words, a linear binary block code $\mathcal{B}$ is a set of $M = 2^K$ codewords $\boldsymbol{v}_i$ such that any linear combination satisfies

$$\sum_i a_i \boldsymbol{v}_i \in \mathcal{B} \qquad i = 0, 1, \ldots, M-1 \quad a_i \in \mathbb{F}_2 \tag{1.20}$$

that is, the sum of any (two or more) codewords of a linear block code forms another codeword. In particular, every linear block code includes necessarily the all-zero codeword $\boldsymbol{v} = \mathbf{0}$.

Moreover, it follows from (1.20) that every (N, K) linear block code $\mathcal{B}$ contains exactly K linearly independent codewords $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{K-1} \in \mathcal{B}$ which form a *basis* for the K-dimensional subspace of the N-dimensional vector space $\mathbb{F}_2^N$.

In other words, every codeword $\boldsymbol{v} \in \mathcal{B}$ can be written as a linear combination of these K codewords

$$\boldsymbol{v} = u_0 \boldsymbol{g}_0 + u_1 \boldsymbol{g}_1 + \ldots + u_{K-1} \boldsymbol{g}_{K-1} \tag{1.21}$$

where $\boldsymbol{u} = (u_0\, u_1 \ldots u_{K-1})$ is the information sequence to be encoded. Using a matrix multiplication within the underlying binary field $\mathbb{F}_2$, this can be expressed as

$$\boldsymbol{v} = \boldsymbol{u} G \tag{1.22}$$

where the rows of the $K \times N$ binary *generator matrix* G are given by $\boldsymbol{g}_i$, $i = 0, 1, \ldots, K-1$. Equation (1.22) is a so-called *linear encoding rule*, which satisfies (1.20) and ensures that the all-zero information sequence $\boldsymbol{u} = \mathbf{0}$ is always mapped to the all-zero codeword $\boldsymbol{v} = \mathbf{0}$.

Performing elementary row operations on the generator matrix G, yields a different set of K linearly independent codewords which form another basis for the same K-dimensional subspace of the N-dimensional vector space $\mathbb{F}_2^N$. The corresponding generator matrix G' encodes the *same* linear block code $\mathcal{B}$, as only the mapping between $\boldsymbol{u}$ and $\boldsymbol{v}$ has been changed, while the set of codewords remains the same.

On the other hand, by permuting the columns of the generator matrix G, the positions of the code symbols v_i are exchanged, yielding a *different* set of codewords, a so-called *equivalent code*. More precisely, a linear block code $\mathcal{B}'$ is said to be equivalent to another linear block code $\mathcal{B}$ if there exists a permutation matrix Π such that every codeword $\boldsymbol{v}' \in \mathcal{B}'$ can be obtained by applying the permutation Π to a codeword $\boldsymbol{v} \in \mathcal{B}$, that is, $\boldsymbol{v}' = \boldsymbol{v}\,\Pi$.

Among the set of all (equivalent) generator matrices of an (N, K) linear block code, there are several generator matrices of particular interest. One such matrix is the *systematic* generator matrix G_{sys} which performs a mapping between the information sequence $\boldsymbol{u}$ and the codeword $\boldsymbol{v}$ such that the information symbols u_i, $i = 0, 1, \ldots, K-1$, appear unchanged among the code symbols v_i, $i = 0, 1, \ldots, N-1$. Performing elementary row operations, a generator matrix can always be transformed into its so-called *reduced row echelon form* (RREF). In the RREF, the *leading coefficient* of each row, also known as the *pivot* element, is always strictly to the right of the leading coefficient of the row above, and is the only nonzero entry in its column. The positions of these leading coefficients are commonly called the *systematic positions*. Using

the concept of equivalent codes, the columns can be permuted in such a way that all systematic positions are grouped together as the first K columns. Then the corresponding systematic generator matrix has the form

$$G_{\text{sys}} = (I_K \mid P) \tag{1.23}$$

where I_K denotes the $K \times K$ identity matrix and P is some $K \times (N-K)$ matrix, also known as the *parity block*. Using the linear encoding rule (1.22), a systematic codeword $\boldsymbol{v}_{\text{sys}}$ follows as

$$\boldsymbol{v}_{\text{sys}} = (u_0\, u_1 \ldots u_{K-1}\, v_K \ldots v_{N-1}) \tag{1.24}$$

whose first K code symbols coincide with the K information symbols.

Consider the set $\mathcal{B}$ of codewords $\boldsymbol{v}$, such that any codeword can be obtained from itself or from another codeword by a cyclic shift of t code symbol positions. Then the linear block code $\mathcal{B}$ is said to be *cyclic* or *quasi-cyclic* if $t = 1$ or $t > 1$, respectively. Among all equivalent generator matrices of a (quasi-) cyclic block code $\mathcal{B}$, there exists always a generator matrix whose rows are cyclically shifted t columns. However, a randomly chosen block code $\mathcal{B}$ may neither be cyclic nor quasi-cyclic.

Besides the generator matrix, any linear block code can be determined by an $(N-K) \times N$ *parity-check matrix* H, whose rows are linearly independent and orthogonal to those of the generator matrix G, such that

$$GH^{\mathrm{T}} = \mathbf{0} \tag{1.25}$$

where $\mathbf{0}$ denotes the $K \times (N-K)$ all-zero matrix. In particular, every codeword $\boldsymbol{v}$ has to satisfy all $N-K$ parity-checks specified by the rows of the parity-check matrix H, that is,

$$\boldsymbol{v}H^{\mathrm{T}} = \mathbf{0} \tag{1.26}$$

Similar to the generator matrix, the parity-check matrix is not unique, as its rows can consist of any $N-K$ linearly independent N-tuples which form an $(N-K)$-dimensional subspace of the N-dimensional vector space $\mathbb{F}_2^N$, orthogonal to the K-dimensional subspace formed by the rows of the generator matrix. In case of a systematic generator matrix (1.23), a corresponding systematic parity-check matrix H_{sys} is given by

$$H_{\text{sys}} = \left(P^{\mathrm{T}} \;\middle|\; I_{N-K}\right) \tag{1.27}$$

When receiving a binary sequence, for example the received sequence $\boldsymbol{r}$ given by (1.1) in case of a BSC, the parity-check matrix H can be used for error detection. Since the *syndrome*

$$\boldsymbol{s} = \boldsymbol{r}H^{\mathrm{T}} = (\boldsymbol{v} + \boldsymbol{e})\, H^{\mathrm{T}} = \underbrace{\boldsymbol{v}H^{\mathrm{T}}}_{=\mathbf{0}} + \boldsymbol{e}H^{\mathrm{T}} = \boldsymbol{e}H^{\mathrm{T}} \tag{1.28}$$

depends only on the binary error pattern $\boldsymbol{e}$, at least one error occurred during transmission if the syndrome $\boldsymbol{s} \neq \boldsymbol{0}$. However, having a zero syndrome $\boldsymbol{s} = \boldsymbol{0}$, is only a necessary but not a sufficient condition for an error free transmission, since any binary error pattern $\boldsymbol{e}$ identically to a codeword $\boldsymbol{v}$ also satisfies $\boldsymbol{e}H^{\mathrm{T}} = \boldsymbol{0}$. Hence, such a binary error pattern is said to be *nondetectable*. For every (N, K) linear block code, there exist $2^K - 1$ nondetectable error patterns.

BASIC DISTANCE PROPERTIES OF BLOCK CODES

The *Hamming weight* w_{H} of a binary vector $\boldsymbol{x}$ of length N is defined as the number of its nonzero elements

$$w_{\mathrm{H}}(\boldsymbol{x}) = \sum_{i=0}^{N-1} w_{\mathrm{H}}(x_i) \quad \text{with} \quad w_{\mathrm{H}}(x_i) = \begin{cases} 0, & x_i = 0 \\ 1, & x_i = 1 \end{cases} \tag{1.29}$$

Closely related, the *Hamming distance* d_{H} between any two binary vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, both of length N, is defined as their number of different elements

$$d_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) = w_{\mathrm{H}}(\boldsymbol{x} \oplus \boldsymbol{y}) = \sum_{i=0}^{N-1} w_{\mathrm{H}}(x_i \oplus y_i) \tag{1.30}$$

The Hamming distance, one of the most important concepts in coding theory, is a *metric*. Thus, for any two binary vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of same length, it satisfies

(i) $d_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ with equality if and only if $\boldsymbol{x} = \boldsymbol{y}$ *(positive definiteness)*

(ii) $d_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) = d_{\mathrm{H}}(\boldsymbol{y}, \boldsymbol{x})$ *(symmetry)*

(iii) $d_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) \leq d_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{z}) + d_{\mathrm{H}}(\boldsymbol{z}, \boldsymbol{y})$ for any binary $\boldsymbol{z}$ *(triangle inequality)*

The error-detecting and error-correcting capabilities of an (N, K) block code $\mathcal{B}$ are largely determined by its *minimum distance* $d_{\min}$ which is defined as

$$d_{\min} = \min_{\boldsymbol{v}, \boldsymbol{v}' \in \mathcal{B}, \boldsymbol{v} \neq \boldsymbol{v}'} \left\{ d_{\mathrm{H}}(\boldsymbol{v}, \boldsymbol{v}') \right\} \tag{1.31}$$

In case of a linear block code, the sum of any two codewords results in another codeword and (1.31) can be simplified to

$$d_{\min} = \min_{\boldsymbol{v} \in \mathcal{B}, \boldsymbol{v} \neq \boldsymbol{0}} \left\{ w_{\mathrm{H}}(\boldsymbol{v}) \right\} \tag{1.32}$$

Such an (N, K) block code with minimum distance $d_{\min}$ is often referred to as an $(N, K, d_{\min})$ block code. In particular, note that the minimum distance $d_{\min}$ is a code property; it is the same for all equivalent codes.

Next, let the set of all error patterns with at most t errors be denoted

$$\mathcal{E}_t = \{ \boldsymbol{e} \mid w_{\mathrm{H}}(\boldsymbol{e}) \leq t \} \tag{1.33}$$

Then, an $(N, K, d_{\min})$ block code is guaranteed to *detect* all error patterns in $\mathcal{E}_t$ if and only if $t < d_{\min}$ or to *correct* all error patterns if and only if

$$t \leq \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor \tag{1.34}$$

Finally, denote by A_w the number of codewords with Hamming weight w of an $(N, K, d_{\min})$ block code $\mathcal{B}$. Then the sequence

$$A_0,\ A_1, \ldots,\ A_{\mathrm{N}} \tag{1.35}$$

is called the *weight spectrum* of $\mathcal{B}$. As the sum of two (or more) codewords of a linear block code forms another codeword, the first *spectral component* of every linear block code is $A_0 = 1$ (since $\boldsymbol{v} + \boldsymbol{v} = \mathbf{0} \in \mathcal{B}$), while the next nonzero spectral component is $A_{d_{\min}}$.

■ **Example 1.1 (Single Parity-Check Code):**
A *single parity-check code* (SPC) of length N encodes a block of $N - 1$ information symbols $\boldsymbol{u}$ by appending a single parity-check bit, such that the resulting codeword $\boldsymbol{v}$ has an even Hamming weight. For $N = 3$ and $K = N - 1 = 2$, the codewords of the $(3, 2)$ SPC block code are

$\boldsymbol{u}$	$\boldsymbol{v}$
0 0	0 0 0
0 1	0 1 1
1 0	1 0 1
1 1	1 1 0

Clearly, an SPC block code is cyclic and systematic. The corresponding generator matrix G and parity-check matrix H for an (N, K) SPC block code are given by (1.23) and (1.27), respectively, with

$$P_{(K \times 1)} = (1\,1 \ldots 1)^{\mathrm{T}} \tag{1.36}$$

Since the minimum distance for all SPCs is $d_{\min} = 2$, any error pattern with odd weight can be detected, but no error pattern is guaranteed to be corrected.

■ **Example 1.2 (Repetition Code):**
An $(N, 1)$ *repetition code* (RC) of length N consists of the codewords $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$, obtained by repeating the information symbol $u \in \{0, 1\}$ N times, that is,

$$\boldsymbol{v}_0 = (0\,0 \ldots 0) \qquad \boldsymbol{v}_1 = (1\,1 \ldots 1) \tag{1.37}$$

with code rate $R = 1/N$. Moreover, the Hamming distance d_{H} between $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ is identical to the minimum distance $d_{\min}$, which is equal to the block length N. Thus, all error patterns in $\mathcal{E}_t$ with $t \leq \lfloor (N-1)/2 \rfloor$ can be corrected. Although increasing N improves the error-correcting capabilities of the code, it decreases the code rate R, which tends to zero for $N \to \infty$.

■ **Example 1.3 (Hamming Code):**
The parity-check matrix of a $(2^m - 1, 2^m - 1 - m)$, $m \geq 2$, *Hamming code* contains all nonzero m-tuples as its columns. For $m = 3$, we obtain the $(7,4)$ Hamming block code with parity-check matrix

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.38}$$

while for $m = 2$ the corresponding $(3,1)$ Hamming code is equal to the $(3,1)$ SPC code. Every Hamming code has minimum distance $d_{\min} = 3$ and is thereby capable of correcting all single-errors. Moreover, note that the parity-check matrix (1.38) contains all columns of the identity matrix, and hence is systematic.

1.3.2 CONVOLUTIONAL CODES

As an alternative to block codes, *convolutional codes* were introduced by Elias [Eli55] in 1955. While block codes are based on splitting the stream of information symbols into separate blocks of length K and encoding each block independently to a codeword $\boldsymbol{v}$ of length N, convolutional codes *continuously* encode a stream of information symbols of (theoretically) infinite length.

A *convolutional encoder* can be represented as a linear sequential circuit, consisting of only memory elements (shift registers) and modulo-2 adders. At each time instant, a tuple of b information symbols enters the encoder, while a tuple of $c \geq b$ code symbols appears at its output. Thus, the rate of a convolutional code $\mathcal{C}$ is

$$R_{\mathrm{c}} = \frac{b}{c} \tag{1.39}$$

and satisfies $0 < R_{\mathrm{c}} \leq 1$. As before, the rate R of the communication system is given by $R = R_{\mathrm{c}} R_{\mathrm{m}}$ (cf. Subsection 1.2.2), which for BPSK signaling becomes $R = R_{\mathrm{c}}$. For example, a convolutional encoder for a rate $R = 1/2$ convolutional code is illustrated in Figure 1.4.

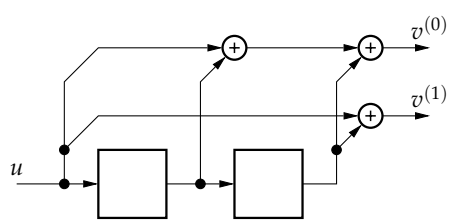


Figure 1.4: An encoder for a rate $R = 1/2$ convolutional code $\mathcal{C}$.

Due to the introduced memory elements in the linear sequential circuit, the c-tuple of code symbols $\boldsymbol{v}_t$ at time instant t depends not only on the information b-tuple $\boldsymbol{u}_t$ at the same time instant, but also on (some of) the information tuples at previous time instants, $\boldsymbol{u}_{t'}$, $t' < t$. To be more precise, a convolutional encoder is a *linear time invariant* (LTI) system whose code c-tuple at time instant t follows from

$$\boldsymbol{v}_t = f\left(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}, \ldots, \boldsymbol{u}_{t-m}\right) \tag{1.40}$$

where the parameter m is called the memory of the encoder, the function f is a linear function from $\mathbb{F}_2^{(m+1)b}$ to $\mathbb{F}_2^c$, and the information and code sequences are assumed to be causal, that is, zero for time instant $t < 0$.

Let the *semi-infinite* information and code sequences be denoted by $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively, such that,

$$\boldsymbol{u} = (\boldsymbol{u}_0\, \boldsymbol{u}_1 \ldots \boldsymbol{u}_t \ldots) = \left(u_0^{(0)}\, u_0^{(1)} \ldots u_0^{(b-1)} \;\; u_1^{(0)}\, u_1^{(1)} \ldots u_1^{(b-1)} \ldots\ldots\right) \tag{1.41}$$

$$\boldsymbol{v} = (\boldsymbol{v}_0\, \boldsymbol{v}_1 \ldots \boldsymbol{v}_t \ldots) = \left(v_0^{(0)}\, v_0^{(1)} \ldots v_0^{(c-1)} \;\; v_1^{(0)}\, v_1^{(1)} \ldots v_1^{(c-1)} \ldots\ldots\right) \tag{1.42}$$

where $u_t^{(i)}$ and $v_t^{(i)}$ refer to the ith component of the corresponding b-tuple $\boldsymbol{u}_t$ and c-tuple $\boldsymbol{v}_t$, respectively. Grouping those sequences according to their input and output, the ith information sequence $\boldsymbol{u}^{(i)}$ as well as the jth code sequences $\boldsymbol{v}^{(j)}$ follow as

$$\boldsymbol{u}^{(i)} = \left(u_0^{(i)}\, u_1^{(i)}\, u_2^{(i)} \ldots\ldots\right) \qquad i = 0, 1, \ldots, b-1 \tag{1.43}$$

$$\boldsymbol{v}^{(j)} = \left(v_0^{(j)}\, v_1^{(j)}\, v_2^{(j)} \ldots\ldots\right) \qquad j = 0, 1, \ldots, c-1 \tag{1.44}$$

Assuming that the memory elements are zero at time instant $t = 0$, the jth code sequence $\boldsymbol{v}^{(j)}$ can be written as the convolution of the information sequences $\boldsymbol{u}^{(i)}$, $i = 0, 1, \ldots, b-1$, with the corresponding impulse responses $\boldsymbol{g}_i^{(j)}$ between input i and output j, that is,

$$v_t^{(j)} = \sum_{i=0}^{b} \sum_{k=0}^{m} u_{t-k}^{(i)} g_{i,k}^{(j)} \tag{1.45}$$

where $g_{i,k}^{(j)}$ denotes the kth value of the impulse response $\boldsymbol{g}_i^{(j)}$, also known as the *generator sequence* $\boldsymbol{g}_i^{(j)}$. For any *finite impulse response* (FIR) LTI system, that is, for convolutional encoders *without feedback*, every generator sequence can be written as

$$\boldsymbol{g}_i^{(j)} = \left(g_{i,0}^{(j)}\, g_{i,1}^{(j)} \ldots g_{i,m}^{(j)}\right) \tag{1.46}$$

where the memory m denotes the maximum number of memory elements among all inputs of the corresponding linear sequential circuit.

Instead of calculating all c code symbols at time instant t separately according to (1.45), it is often more convenient to determine the corresponding code c-tuple $\boldsymbol{v}_t$ directly as

$$\boldsymbol{v}_t = \sum_{k=0}^{m} \boldsymbol{u}_{t-k} G_k = \boldsymbol{u}_t G_0 + \boldsymbol{u}_{t-1} G_1 + \cdots \boldsymbol{u}_{t-m} G_m \tag{1.47}$$

where G_k, $0 \leq k \leq m$, are binary $b \times c$ matrices with $g_{i,k}^{(j)}$ at row i and column j.

Using (1.41) and (1.42), the convolution (1.47) can be written more compactly as the matrix multiplication

$$\boldsymbol{v} = \boldsymbol{u}\boldsymbol{G} \tag{1.48}$$

where the semi-infinite generator matrix $\boldsymbol{G}$ is given by

$$\boldsymbol{G} = \begin{pmatrix} G_0 & G_1 & \dots & G_m & & & \\ & G_0 & G_1 & \dots & G_m & & \\ & & G_0 & G_1 & \dots & G_m & \\ & & & \ddots & \ddots & \dots & \ddots \end{pmatrix} \tag{1.49}$$

■ **Example 1.4:**
Consider the convolutional encoder given in Figure 1.4. Its generator sequences (impulse responses) are

$$\boldsymbol{g}_0^{(0)} = \begin{pmatrix} 1\,1\,1 \end{pmatrix} \qquad \boldsymbol{g}_0^{(1)} = \begin{pmatrix} 1\,0\,1 \end{pmatrix} \tag{1.50}$$

with memory $m = 2$. Thus, its $m+1$ binary 1×2 matrices G_0, G_1 and G_2 are

$$G_0 = \begin{pmatrix} 1 & 1 \end{pmatrix} \qquad G_1 = \begin{pmatrix} 1 & 0 \end{pmatrix} \qquad G_2 = \begin{pmatrix} 1 & 1 \end{pmatrix} \tag{1.51}$$

while its semi-infinite generator matrix $\boldsymbol{G}$ follows as

$$\boldsymbol{G} = \begin{pmatrix} 11 & 10 & 11 & & & \\ & 11 & 10 & 11 & & \\ & & 11 & 10 & 11 & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \tag{1.52}$$

Using (1.48), the code sequence $\boldsymbol{v}$ corresponding to the information sequence $\boldsymbol{u} = (1\,0\,1)$ is given by

$$\boldsymbol{v} = \boldsymbol{u}\boldsymbol{G} = (11\ 10\ 00\ 10\ 11) \tag{1.53}$$

While the previously introduced notations are suitable for FIR LTI systems, they have to be chosen more subtle when dealing with *infinite impulse response* (IIR) LTI systems, due to their generator sequences of infinite support. To accurately describe such an IIR LTI system, that is, a convolutional encoder *with feedback*, the information sequence $\boldsymbol{u}$ and code sequence $\boldsymbol{v}$ have to be represented using the D-transform in terms of the *delay operator* D (D-domain):

$$\boldsymbol{u}(D) = \sum_{t=0}^{\infty} \boldsymbol{u}_t D^t = \boldsymbol{u}_0 + \boldsymbol{u}_1 D + \boldsymbol{u}_2 D^2 + \cdots \tag{1.54}$$

$$\boldsymbol{v}(D) = \sum_{t=0}^{\infty} \boldsymbol{v}_t D^t = \boldsymbol{v}_0 + \boldsymbol{v}_1 D + \boldsymbol{v}_2 D^2 + \cdots \tag{1.55}$$

Similarly, every generator sequence $\boldsymbol{g}_i^{(j)}$ can be expressed as

$$\boldsymbol{g}_{ij}(D) = \sum_{t=0}^{m} g_{i,t}^{(j)} D^t = g_{i,0}^{(j)} + g_{i,1}^{(j)} D + g_{i,2}^{(j)} D^2 + \cdots \tag{1.56}$$

Note that the indices of the time domain representation $\boldsymbol{g}_i^{(j)}$ and the D-domain representation $\boldsymbol{g}_{ij}(D)$ differ. Using the latter representation, the linear encoding rules (1.45) and (1.48) can be rewritten as

$$\boldsymbol{v}(D) = \boldsymbol{u}(D) G(D) \tag{1.57}$$

where the $b \times c$ generator matrix $G(D)$ is given by

$$G(D) = \begin{pmatrix} \boldsymbol{g}_{00}(D) & \boldsymbol{g}_{01}(D) & \cdots & \boldsymbol{g}_{0(c-1)}(D) \\ \boldsymbol{g}_{10}(D) & \boldsymbol{g}_{11}(D) & \cdots & \boldsymbol{g}_{1(c-1)}(D) \\ \vdots & \vdots & \ddots & \cdots \\ \boldsymbol{g}_{(b-1)0}(D) & \boldsymbol{g}_{(b-1)1}(D) & \cdots & \boldsymbol{g}_{(b-1)(c-1)}(D) \end{pmatrix} \tag{1.58}$$

To guarantee a one-to-one mapping between the information sequence $\boldsymbol{u}$ and the code sequence $\boldsymbol{v}$, the generator matrix $G(D)$ has to have full rank. In case of an IIR LTI system, the generator sequences $\boldsymbol{g}_{ij}(D)$ are in general rational functions of the form $f(D)/q(D)$, where $f(D)$ and $q(D)$ are polynomials in D describing the *feedforward* and *feedback* part, respectively. Moreover, $q(D)$ is delayfree, that is, its first coefficient is nonzero. Finally, a generator matrix $G(D)$ is called *rational* if at least one of its generator sequences is rational, which holds for IIR LTI systems.

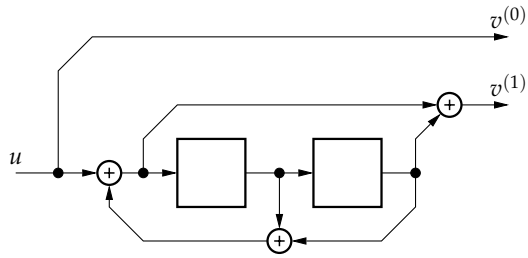


Figure 1.5: A rate $R = 1/2$ systematic encoder with feedback for a convolutional code $\mathcal{C}$.

■ **Example 1.4 (Cont'd):**
Applying the D-transform to the generator sequences in (1.50), we obtain

$$g_{00}(D) = 1 + D + D^2 \qquad g_{01}(D) = 1 + D^2 \tag{1.59}$$

and hence

$$G(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix} \tag{1.60}$$

■ **Example 1.5:**
The rate $R = 1/2$ convolutional systematic encoder with feedback, illustrated in Figure 1.5, represents an IIR LTI system and is determined by the rational generator matrix

$$G'(D) = \begin{pmatrix} 1 & \dfrac{1 + D^2}{1 + D + D^2} \end{pmatrix} \tag{1.61}$$

If all generator sequences $g_{ij}(D)$ are polynomials, the generator matrix $G(D)$ is said to be *polynomial* and its elements are referred to as *generator polynomials*. In particular, such a generator matrix can be written as

$$G(D) = G_0 + G_1 D + \cdots + G_m D^m \tag{1.62}$$

where G_t, $t = 0, 1, \ldots, m$, are the binary $b \times c$ matrices used in (1.47). If G_0 has full rank, then $G(D)$ is called an *encoding matrix*. Moreover, if a polynomial encoding matrix $G(D)$ has a polynomial right inverse $G^{-1}(D)$, such that

$$G(D)G^{-1}(D) = I \tag{1.63}$$

where I is the $b \times b$ identity matrix, then $G(D)$ is said to be *basic*. In particular, every basic encoding matrix is *noncatastrophic*, that is, every information sequence of infinite weight is mapped to a code sequence of infinite weight. On the other hand, if there exists an information sequence of infinite weight,

which is mapped to a code sequence of finite weight, the corresponding generator matrix is said to be *catastrophic*. In such cases, a finite number of transmission or decoding errors can yield infinitely many bit errors, and hence such catastrophic matrices should be avoided.

Let the ith *constraint length* ν_i of the generator matrix $G(D)$ be given by

$$\nu_i = \max_{j=0,1,\ldots,c-1} \Big\{ \deg \boldsymbol{g}_{ij}(D) \Big\} = \max \Big\{ \max_{j=0,1,\ldots,c-1} \big\{ \deg \boldsymbol{f}_{ij}(D) \big\}, \boldsymbol{q}_i(D) \Big\} \quad (1.64)$$

where the latter equality holds only for rational encoder matrices with $\boldsymbol{g}_{ij}(D) = \boldsymbol{f}_{ij}(D)/\boldsymbol{q}_i(D)$, assuming a common feedback polynomial $\boldsymbol{q}_i(D)$ within each row. Then the *overall constraint length* ν as well as the *memory* m of a convolutional encoder can be formally defined as

$$\nu = \sum_{i=0}^{b-1} \nu_i \qquad \text{and} \qquad m = \max_{i=0,1,\ldots,b-1} \{\nu_i\} \quad (1.65)$$

Apart from the generator matrix, any rate $R = b/c$ convolutional code $\mathcal{C}$ can be fully determined by a $(c-b) \times c$ parity-check matrix $H(D)$, whose rows specify independent parity-checks and are orthogonal to the generator matrix $G(D)$, such that

$$G(D)H(D)^{\mathrm{T}} = \mathbf{0} \quad (1.66)$$

where $\mathbf{0}$ is the $b \times (c-b)$ all-zero matrix. Analogous to (1.26) for block codes, it follows from (1.57), that every convolutional code sequence $\boldsymbol{v}(D)$ has to satisfy all $c-b$ parity-checks, and hence

$$\boldsymbol{v}(D)H(D)^{\mathrm{T}} = \mathbf{0} \quad (1.67)$$

Similar to (1.62), a parity-check matrix $H(D)$ with *syndrome memory* m_{s} can be expressed via its $m_s + 1$ binary submatrices H_i of size $(c-b) \times c$, that is,

$$H(D) = H_0 + H_1 D + \cdots + H_{m_{\mathrm{s}}} D^{m_{\mathrm{s}}} \quad (1.68)$$

Moreover, the corresponding semi-infinite parity-check matrix H in the time-domain follows analogously to (1.49) as

$$H = \begin{pmatrix} H_0 & & & \\ H_1 & H_0 & & \\ \vdots & H_1 & H_0 & \\ H_{m_{\mathrm{s}}} & \vdots & H_1 & \ddots \\ & H_{m_{\mathrm{s}}} & \vdots & \ddots \\ & & H_{m_{\mathrm{s}}} & \ddots \\ & & & \ddots \end{pmatrix} \quad (1.69)$$

and similar to (1.66)

$$GH^{\mathrm{T}} = \mathbf{0} \quad (1.70)$$

As before, the syndrome of a convolutional code follows as the product of the received sequence $\boldsymbol{r}$ and the transpose of its semi-infinite parity-check matrix H^{T}; commonly referred to as the *syndrome former*.

A convolutional code can be encoded by different generator matrices, which specify different mappings between the information sequences $\boldsymbol{u}$ and the code sequences $\boldsymbol{v}$. If two generator matrices $G(D)$ and $G'(D)$ generate the same convolutional code $\mathcal{C}$, that is, the same set of code sequences, they are called *equivalent*. In other words, two equivalent generator matrices $G(D)$ and $G'(D)$ satisfy

$$G(D) = T(D)G'(D) \tag{1.71}$$

where $T(D)$ is a $b \times b$ nonsingular matrix. In particular, every convolutional code can be described by an equivalent (rational) systematic generator matrix $G_{\mathrm{sys}}(D)$, such that, the information b-tuple at time instant t appears unchanged among the code c-tuple at the same time instant.

Finally note that every generator matrix can be realized by several different sequential circuits. Among those, two realizations are commonly used; namely, the *controller canonical form* (CCF) and the *observer canonical form* (OCF).

- A CCF encoder of a rate $R = b/c$ convolutional code consists of b shift-registers, one for each input. The length of the ith shift-register is equal to the ith constraint length ν_i, and hence a CCF encoder consists of in total ν memory elements. For example, the two CCF encoders for the generator matrices (1.60) and (1.61) are shown in Figure 1.4 and Figure 1.5, respectively.
- An OCF encoder of a rate $R = b/c$ convolutional code consists of c memory chains, one for each output. Figure 1.6 illustrates for example the OCF encoder for the systematic generator matrix (1.61).

For any encoder realization, we refer to the contents of its memory elements as its *encoder state*. The set of all possible states forms the *state space* of the encoder. Since the number of possible states is finite, an encoder for a convolutional code is an instance of a *finite-state machine*.

■ **Example 1.6:**
The encoder matrices $G(D)$ and $G'(D)$ used in Example 1.4 and 1.5 are equivalent since they satisfy (1.71)

$$\underbrace{\begin{pmatrix} 1+D+D^2 & 1+D^2 \end{pmatrix}}_{G(D)} = \underbrace{\begin{pmatrix} 1+D+D^2 \end{pmatrix}}_{T(D)} \underbrace{\begin{pmatrix} 1 & \dfrac{1+D^2}{1+D+D^2} \end{pmatrix}}_{G'(D)}$$

where $G'(D)$ is a systematic encoding matrix. In Figures 1.4 and 1.5 the CCF encoders for $G(D)$ and $G'(D)$ are illustrated, respectively, while the corresponding OCF encoder for $G'(D)$ is shown in Figure 1.6.

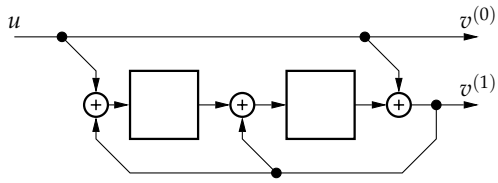


Figure 1.6: A systematic encoder for a rate $R = 1/2$ convolutional code $\mathcal{C}$, realized in OCF.

Finding an encoder realization for a given convolutional code with a minimal number of memory elements among all equivalent generator matrices is of particular interest. Such a generator matrix is referred to as a *minimal* generator matrix, while its corresponding realization is said to be a *minimal realization*. However, such a minimal realization might be neither in CCF nor in OCF.

Recall the definition of the generator matrix $G(D)$ in terms of its $m+1$ binary submatrices G_i, $i = 0, 1, \ldots, m$ in (1.62). It can be shown that for every minimal generator matrix $G(D)$, the binary submatrix G_0 has full rank, and hence $G(D)$ is a minimal encoding matrix.

Moreover, limiting ourselves to minimal encoding matrices whose minimal realizations are given in CCF, we obtain the set of, in general, rational *canonical* encoding matrices. Imposing the additional restriction to only polynomial generator sequences yields the subset of so-called *minimal-basic* encoding matrices, that are polynomial encoding matrices whose CCF realizations have the smallest number of memory elements ν among all equivalent, maybe rational, generator matrices.

In particular, it can be shown that for any rational generator matrix, there exists always an equivalent minimal-basic encoding matrix. Hence, when referring to the overall constraint length ν or the memory m of a convolutional code $\mathcal{C}$, those parameters correspond to a minimal realization in CCF of an equivalent minimal-basic polynomial encoding matrix for the convolutional code $\mathcal{C}$.

Consider the polynomial generator matrix of a convolutional code with memory m where each generator polynomial is represented as a binary generator sequence of length $m+1$, grouped into blocks of 3 bits. Converting each sequence of 3-bit blocks into octal digits, where zeros are being padded from the right if needed, the commonly used *octal notation* for a polynomial generator matrix is obtained. For example, the encoding matrix $G(D)$ in Example 1.6 is given by $(111\ 101)_2$ or $(7,5)_8$ while the generator polynomial $1 + D^2 + D^3 + D^4$ would be represented as $(101\ 110)_2 = (56)_8$.

BASIC DISTANCE PROPERTIES OF CONVOLUTIONAL CODES

The *Hamming weight* w_{H} as well as the *Hamming distance* d_{H}, previously defined for block codes in (1.30) and (1.29), can be similarly defined for convolutional codes, taking into account the binary sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ of (theoretically) infinite length but finite support:

$$d_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) = w_{\mathrm{H}}(\boldsymbol{x} \oplus \boldsymbol{y}) = \sum_{i=0}^{\infty} w_{\mathrm{H}}(x_i \oplus y_i) \tag{1.72}$$

with

$$w_{\mathrm{H}}(x_i) = \begin{cases} 0, & x_i = 0 \\ 1, & x_i = 1 \end{cases}$$

As a counterpart to the minimum distance for a block code, the *free distance* d_{free} determines the error-detecting and error-correcting capabilities for a convolutional code $\mathcal{C}$ and is defined as the minimum Hamming distance between any two code sequences $\boldsymbol{v}, \boldsymbol{v}' \in \mathcal{C}$. Due to the linearity of convolutional codes, this definition is equivalent to the minimum Hamming weight of any nonzero code sequence, that is,

$$d_{\mathrm{free}} = \min_{\boldsymbol{v} \in \mathcal{C}, \boldsymbol{v} \neq \boldsymbol{0}} \{w_{\mathrm{H}}(\boldsymbol{v})\} \tag{1.73}$$

Given the set of error patterns $\mathcal{E}_t$ according to (1.33), a convolutional code with free distance d_{free} is guaranteed to detect all error patterns in $\mathcal{E}_t$ as long as $t < d_{\mathrm{free}}$ or to correct all error patterns if and only if t satisfies (1.34), where the minimum distance d_{min} is replaced by the free distance d_{free}.

RELATIONS BETWEEN CONVOLUTIONAL AND LINEAR BLOCK CODES

Restricting the (theoretically) infinite length of a convolutional code $\mathcal{C}$ of memory m with minimal-basic encoding matrix $G(D)$ to a finite length, a corresponding block code $\mathcal{B}$ with similar properties is obtained. Such a block code $\mathcal{B}$ is said to originate from its *parent convolutional code* $\mathcal{C}$. Commonly used termination techniques are *truncation* (TR), *zero-tail* (ZT) termination, and *tail-biting* (TB), which shall be discussed in the following.

Truncation to length M starts the encoding process at the all-zero encoding state at time instant $t = 0$. After processing M information b-tuples, that is, at time instant $t = M$, the encoding process is terminated regardless of the current encoder state. This yields a codeword consisting of M c-tuples of an (Mc, Mb) linear block code. The generator matrix of the corresponding block

code is given by terminating the semi-infinite generator matrix (1.49) after M columns

$$\boldsymbol{G}^{(\mathrm{tr})} = \begin{pmatrix} G_0 & G_1 & G_2 & \dots & G_m & & & \\ & G_0 & G_1 & G_2 & \dots & G_m & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & G_0 & G_1 & G_2 & \dots & G_m \\ & & & & G_0 & G_1 & \dots & G_{m-1} \\ & & & & & \ddots & \ddots & \vdots \\ & & & & & & & G_0 \end{pmatrix} \tag{1.74}$$

As the encoding process is ended after processing M information b-tuples, the last m information b-tuples are not completely encoded, and hence less protected against channel distortion. Using zero-tail termination to length M, m additional »dummy« b-tuples are added to the encoding process, enforcing the all-zero ending state[1]. Hence, after having encoded M information b-tuples at time instant $t = M$, m additional »dummy« b-tuples are encoded which guarantee that at time instant $t = M + m$ the all-zero encoding state is reached. This yields an $\big((M+m)c, Mb\big)$ linear binary block code with a slightly decreased code rate

$$R^{(\mathrm{zt})} = \frac{Mb}{(M+m)c} < \frac{b}{c} = R \tag{1.75}$$

The generator matrix of the corresponding block code follows by terminating the semi-infinite generator matrix (1.49) after M rows, that is, after $M + m$ columns,

$$\boldsymbol{G}^{(\mathrm{zt})} = \begin{pmatrix} G_0 & G_1 & G_2 & \dots & G_m & & & \\ & G_0 & G_1 & G_2 & \dots & G_m & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & G_0 & G_1 & G_2 & \dots & G_m \end{pmatrix} \tag{1.76}$$

However, the rate loss introduced by zero-tail termination might not be acceptable in case of a small termination length M. Hence, a third alternative is tailbiting to length M, which fully encodes the last m information b-tuples without decreasing the code rate.

Consider the b-shift registers of a convolutional encoder with memory m and overall constraint length ν, realized in CCF. At time instant $t = M$, the contents of the ith shift register is equal to the previous ν_i information symbols, where ν_i denotes the ith constraint length, that is,

$$\left(u_{M-1}^{(i)}\, u_{M-2}^{(i)} \dots u_{M-\nu_i}^{(i)}\right) \tag{1.77}$$

[1]For encoders without feedback this is accomplished by m zero b-tuples, while for encoders with feedback these b-tuples have to be chosen more carefully and are in general nonzero.

Clearly, those information symbols have not yet passed the encoder, that is, are not yet fully encoded. However, suppose that we initialize the contents of the memory elements at time instant $t = 0$ with exactly those last information symbols given in (1.77). Then, those symbols are already partially encoded at time instants $t = 0, 1, \ldots, m-1$ and terminating the encoding process at time instant $t = M$ does not yield less protected information symbols. Such an (Mc, Mb) block code is determined by the generator matrix

$$\boldsymbol{G}^{(\mathrm{tb})} = \begin{pmatrix} G_0 & G_1 & G_2 & \ldots & G_m & & & \\ & G_0 & G_1 & G_2 & \ldots & G_m & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & G_0 & G_1 & G_2 & \ldots & G_m \\ G_m & & & & G_0 & G_1 & \ldots & G_{m-1} \\ \vdots & \ddots & & & & \ddots & \ddots & \vdots \\ G_1 & \ldots & G_m & & & & & G_0 \end{pmatrix} \tag{1.78}$$

Applying the same restriction to the semi-infinite parity-check matrix H specified in (1.69) with syndrome memory m_s yields the corresponding tail-biting parity-check matrix

$$H^{(\mathrm{tb})} = \begin{pmatrix} H_0 & & & & H_{m_\mathrm{s}} & H_{m_\mathrm{s}-1} & \ldots & H_1 \\ H_1 & H_0 & & & & H_{m_\mathrm{s}} & \ldots & H_2 \\ H_2 & H_1 & \ddots & & & & \ddots & \vdots \\ \vdots & H_2 & \ddots & H_0 & & & & H_{m_\mathrm{s}} \\ H_{m_\mathrm{s}} & \vdots & \ddots & H_1 & H_0 & & & \\ & H_{m_\mathrm{s}} & \ddots & H_2 & H_1 & \ddots & & \\ & & \ddots & \ldots & \ldots & \ddots & \ddots & \\ & & & H_{m_\mathrm{s}} & H_{m_\mathrm{s}-1} & \ldots & H_1 & H_0 \end{pmatrix} \tag{1.79}$$

where in general $m \neq m_\mathrm{s}$. However, the pair $G^{(\mathrm{tb})}$ and $H^{(\mathrm{tb})}$ satisfies

$$G^{(\mathrm{tb})} \left(H^{(\mathrm{tb})}\right)^{\mathrm{T}} = \mathbf{0} \tag{1.80}$$

given that (1.70) is fulfilled.

1.3.3 OPTIMAL DECODING PRINCIPLES

When discussing an *optimal decoder*, it is necessary to distinguish between an *optimal sequence (block) decoder* and an *optimal symbol decoder*. An optimal sequence decoder minimizes the *block error probability* P_B, that is, the probability that the decided information sequence $\hat{\boldsymbol{u}}$ differs from the actual information sequence $\boldsymbol{u}$, that is,

$$P_\mathrm{B} = P\left(\boldsymbol{u} \neq \hat{\boldsymbol{u}}\right) = P\left(\boldsymbol{v} \neq \hat{\boldsymbol{v}}\right) \tag{1.81}$$

For a information sequence $\boldsymbol{u}$ of length K with equiprobable information symbols, an optimal symbol decoder minimizes the (average) *bit error probability* P_b given by

$$P_\mathrm{b} = \frac{1}{K} \sum_{i=0}^{K-1} P\left(u_i \neq \hat{u}_i\right) \tag{1.82}$$

Since a block error occurs if there exists at least one and at most K bit errors, the block error probability P_B can be lower- and upper-bounded by the bit error probability P_b as

$$P_\mathrm{b} \leq P_\mathrm{B} \leq K P_\mathrm{b} \tag{1.83}$$

SEQUENCE DECODING

When communicating over a DMC like the BSC or the binary input, quantized AWGN channel, the block error probability P_B (1.81) can be expressed as

$$\begin{aligned} P_\mathrm{B} &= P\left(\boldsymbol{u} \neq \hat{\boldsymbol{u}}\right) = P\left(\boldsymbol{v} \neq \hat{\boldsymbol{v}}\right) \\ &= 1 - P\left(\boldsymbol{v} = \hat{\boldsymbol{v}}\right) \\ &= 1 - \sum_{\boldsymbol{r}} P\left(\boldsymbol{v} = \hat{\boldsymbol{v}} \mid \boldsymbol{r}\right) P\left(\boldsymbol{r}\right) \\ &\stackrel{\text{def}}{=} 1 - \sum_{\boldsymbol{r}} P\left(\hat{\boldsymbol{v}} \mid \boldsymbol{r}\right) P\left(\boldsymbol{r}\right) \\ &= 1 - \sum_{\boldsymbol{r}} P\left(\boldsymbol{r} \mid \hat{\boldsymbol{v}}\right) P\left(\hat{\boldsymbol{v}}\right) \end{aligned} \tag{1.84}$$

where the last identity follows from Bayes' rule. The conditional probability $P\left(\boldsymbol{r} \mid \boldsymbol{v}\right)$ describes the probability for observing the sequence $\boldsymbol{r}$ given that the code sequence $\boldsymbol{v}$ has been transmitted. This probability is also known as the *likelihood* of the code sequence $\boldsymbol{v}$ and is solely determined by the properties of the underlying channel. The probability $P\left(\boldsymbol{v} \mid \boldsymbol{r}\right)$ is denoted the *a posteriori* probability (APP) of the code sequence $\boldsymbol{v}$ and describes the probability for the transmission of the code sequence $\boldsymbol{v}$, given that the received sequence $\boldsymbol{r}$ is observed.

To minimize the block error probability P_B in (1.84) it is sufficient to maximize the APP $P\left(\boldsymbol{v} \mid \boldsymbol{r}\right)$ for every received sequence $\boldsymbol{r}$. Such a decoder is called a *maximum a posteriori* (MAP) sequence decoder and decides in favor of the code sequence $\hat{\boldsymbol{v}}$ with the largest probability, given the received sequence $\boldsymbol{r}$. Thus, the MAP decoding rule is

$$\hat{\boldsymbol{v}}_\mathrm{MAP} = \arg\max_{\boldsymbol{v}} \left\{ P\left(\boldsymbol{v} \mid \boldsymbol{r}\right) \right\} = \arg\max_{\boldsymbol{v}} \left\{ P\left(\boldsymbol{r} \mid \boldsymbol{v}\right) P\left(\boldsymbol{v}\right) \right\} \tag{1.85}$$

Such a MAP decoder requires the knowledge of the *a priori* probabilities $P(\boldsymbol{v}) = P(\boldsymbol{u})$ determined by the source, and hence has to be optimized individually for every communication system.

On the other hand, a BSS yields equal *a priori* probabilities for both the information and the code sequences. Hence, maximizing the APP is equivalent to maximizing the likelihoods $P(\boldsymbol{r} \mid \boldsymbol{v})$ and the *maximum-likelihood* (ML) decoding rule follows as

$$\hat{\boldsymbol{v}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{v}} \left\{ P(\boldsymbol{r} \mid \boldsymbol{v}) \right\} \tag{1.86}$$

Such an ML decoder can be applied to any communication system without demanding knowledge of the *a priori* probabilities of the underlying source. However, only in case of equal *a priori* probabilities, its decision is optimal.

Next, consider a BSC with crossover probability p, and let the likelihood of a codeword $\boldsymbol{v}$ be given by

$$P(\boldsymbol{r} \mid \boldsymbol{v}) = p^{d_{\mathrm{H}}(\boldsymbol{v},\boldsymbol{r})} (1-p)^{N-d_{\mathrm{H}}(\boldsymbol{v},\boldsymbol{r})} = (1-p)^N \left(\frac{p}{1-p} \right)^{d_{\mathrm{H}}(\boldsymbol{v},\boldsymbol{r})} \tag{1.87}$$

where N is the codeword length and $d_{\mathrm{H}}(\boldsymbol{v},\boldsymbol{r})$ denotes the Hamming distance between the received sequence $\boldsymbol{r}$ and the codeword $\boldsymbol{v}$. Since the crossover probability $p \leq 0.5$, we have

$$0 < \frac{p}{1-p} < 1 \tag{1.88}$$

and hence the likelihood of the codeword $\boldsymbol{v}$ in (1.87) is maximized by minimizing the Hamming distance $d_{\mathrm{H}}(\boldsymbol{v},\boldsymbol{r})$. In other words, an ML decoder for a BSC is equivalent to a *minimum (Hamming) distance* (MD) decoder. Such a decoder decides in favor of the information sequence $\hat{\boldsymbol{u}}$, whose codeword $\hat{\boldsymbol{v}}$ is closest to the received sequence $\boldsymbol{r}$ in terms of the Hamming distance

$$\hat{\boldsymbol{v}}_{\mathrm{ML}} = \hat{\boldsymbol{v}}_{\mathrm{MD}} = \arg\max_{\boldsymbol{v}} \left\{ \left(\frac{p}{1-p} \right)^{d_{\mathrm{H}}(\boldsymbol{v},\boldsymbol{r})} \right\} = \arg\min_{\boldsymbol{v}} \left\{ d_{\mathrm{H}}(\boldsymbol{v},\boldsymbol{r}) \right\} \tag{1.89}$$

Considering a unquantized AWGN channel with a continuous output alphabet, the sum over the conditional probabilities $P(\boldsymbol{r} \mid \boldsymbol{v})$ in (1.84) has to be replaced by the integral over the corresponding conditional PDFs $p(\boldsymbol{r} \mid \boldsymbol{v})$. From (1.8) it follows that

$$p(\boldsymbol{r} \mid \boldsymbol{v}) = p(\boldsymbol{r} \mid \boldsymbol{x}) = \left(\frac{1}{\sqrt{\pi N_0}} \right)^N e^{-\|\boldsymbol{r}-\boldsymbol{x}\|^2/N_0} \tag{1.90}$$

where $\|\boldsymbol{r}-\boldsymbol{x}\|^2$ denotes the squared Euclidean distance between the received sequence $\boldsymbol{r}$ and the modulated and transmitted sequence $\boldsymbol{x}$. In particular note that there exists a unique mapping between the code sequences $\boldsymbol{v}$ and modulated sequences $\boldsymbol{x}$, and hence their PDFs coincide.

Moreover, since the exponential function is monotonically increasing, the ML decoding rule (1.86) can be alternatively expressed as

$$\begin{aligned}\hat{\boldsymbol{x}}_{\mathrm{ML}} &= \arg\max_{\boldsymbol{x}} \left\{ p\left(\boldsymbol{r} \mid \boldsymbol{x}\right) \right\} = \arg\max_{\boldsymbol{x}} \left\{ \log p\left(\boldsymbol{r} \mid \boldsymbol{x}\right) \right\} \\ &= \arg\min_{\boldsymbol{x}} \left\{ \|\boldsymbol{r}-\boldsymbol{x}\|^2 \right\} = \arg\min_{\boldsymbol{x}} \left\{ \|\boldsymbol{r}-\boldsymbol{x}\| \right\} \end{aligned} \tag{1.91}$$

that is, the corresponding ML decoder is equivalent to a *minimum (squared) Euclidean distance decoder*, which decides in favor of the information sequence $\hat{\boldsymbol{u}}$, whose code sequence $\hat{\boldsymbol{v}}$, and hence modulated sequence $\hat{\boldsymbol{x}}$, is closest to the received sequence $\boldsymbol{r}$ in terms of the (squared) Euclidean distance.

Note that the squared Euclidean distance $\|\boldsymbol{r}-\boldsymbol{x}\|^2$ can be expressed as

$$\|\boldsymbol{r}-\boldsymbol{x}\|^2 = \|\boldsymbol{r}\|^2 + \|\boldsymbol{x}\|^2 - 2\,\mathrm{Re}\left(\langle \boldsymbol{r}, \boldsymbol{x}\rangle\right) \tag{1.92}$$

where

$$\langle \boldsymbol{r}, \boldsymbol{x}\rangle = \boldsymbol{r}\boldsymbol{x}^{\mathrm{H}} = \sum_i r_i\, x_i^* \tag{1.93}$$

denotes the correlation between two vectors and x_i^* is the complex conjugate of x_i. Under the assumption of equi-energy signaling, $\|\boldsymbol{x}\|^2$ is constant for all received sequences $\boldsymbol{r}$, and thus minimizing the squared Euclidean distance in (1.91) is equivalent to maximizing $\langle \boldsymbol{r}, \boldsymbol{x}\rangle$, yielding a *maximum correlation decoder*. Moreover, for a real-valued modulation alphabet like BPSK, the corresponding decoding rule can be further simplified to

$$\hat{\boldsymbol{x}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{x}} \left\{ \langle \boldsymbol{r}, \boldsymbol{x}\rangle \right\} = \arg\max_{\boldsymbol{x}} \left\{ \sum_i r_i x_i \right\} \tag{1.94}$$

Next, we consider the *weighted Hamming distance* between the received sequence $\boldsymbol{r}$ and the code sequence $\boldsymbol{v}$, defined as

$$d_{\mathrm{wH}}\left(\boldsymbol{r}, \boldsymbol{v}\right) = \sum_{i \in \mathcal{E}(\boldsymbol{r},\boldsymbol{x})} |r_i| \tag{1.95}$$

where $\mathcal{E}(\boldsymbol{r}, \boldsymbol{x})$ denotes the set of all error positions

$$\mathcal{E}(\boldsymbol{r}, \boldsymbol{x}) = \{i \mid \mathrm{sign}(r_i) \neq \mathrm{sign}(x_i)\} \tag{1.96}$$

For BPSK modulation, the sum in (1.94) can be expressed as

$$\sum_i r_i x_i = \sum_i |r_i| - 2\sum_{i \in \mathcal{E}(\boldsymbol{r},\boldsymbol{x})} |r_i| = \sum_i |r_i| - 2 d_{\mathrm{wH}}\left(\boldsymbol{r}, \boldsymbol{x}\right) \tag{1.97}$$

and hence minimizing the (squared) Euclidean distance in (1.91) is equivalent to minimizing the weighted Hamming distance. That is, the ML decoding rule for a *weighted Hamming distance decoder* is given by

$$\hat{\boldsymbol{x}}_{\mathrm{ML}} = \arg\min_{\boldsymbol{x}} \left\{ d_{\mathrm{wH}} \left(\boldsymbol{r}, \boldsymbol{x} \right) \right\} \tag{1.98}$$

Note that the ML decoding decision is not necessarily uniquely determined, as several code sequences might have the same probability. Commonly, such cases are resolved by coin-flipping, that is, by randomly choosing one of the code sequences with largest likelihood. While such events can occur when communicating over a BSC or a quantized AWGN channel, their probabilities are zero when communicating over an unquantized AWGN channel, and hence its ML decision is commonly assumed to be unique.

SYMBOL DECODING

Let $\{\boldsymbol{v}\}_{u_i=u}$ be the set of all code sequences $\boldsymbol{v}$ whose corresponding ith information symbol is equal to $u \in \{0,1\}$. For a given received sequence $\boldsymbol{r}$, denote by

$$P\left(\{\boldsymbol{x}\}_{u_i=u} \,\middle|\, \boldsymbol{r} \right) = P\left(u_i = u \mid \boldsymbol{r} \right) = \sum_{\boldsymbol{u}:u_i=u} P\left(\boldsymbol{u} \mid \boldsymbol{r} \right) \tag{1.99}$$

the probability of the set of transmitted sequences $\boldsymbol{x}$ whose ith information symbol is equal to $u \in \{0,1\}$, given the received sequence $\boldsymbol{r}$.

An optimal symbol decoder decides individually for each information symbol u_i, $i = 0, 1, \ldots, K-1$ in favor of the value $u \in \{0,1\}$, which maximizes (1.99). That is, the MAP symbol decoding rule is given by

$$\hat{u}_{i\,\mathrm{MAP}} = \arg\max_{u \in \{0,1\}} \left\{ P\left(u_i = u \mid \boldsymbol{r} \right) \right\} \qquad i = 0, 1, \ldots, K-1 \tag{1.100}$$

where the ith information symbol APP for a DMC is given by

$$P\left(u_i = u \mid \boldsymbol{r} \right) = \frac{1}{P(\boldsymbol{r})} \sum_{\boldsymbol{u}:u_i=u} P\left(\boldsymbol{r} \mid \boldsymbol{u} \right) P\left(\boldsymbol{u} \right) \tag{1.101}$$

While the performance of MAP/ML sequence decoding is equal among equivalent codes, MAP symbol decoding depends on the mapping between the information sequence $\boldsymbol{u}$ and the code sequence $\boldsymbol{v}$. In other words, the bit error probability P_{b} is an encoder property and depends on the encoder realization, while the corresponding block error probability P_{B} is invariant among equivalent encoders.

1.4 DISSERTATION OUTLINE

The material in this dissertation is organized as follows: In Chapter 2, prerequisites for the remaining chapters are reviewed. In particular, tree- and trellis-based representations for block and convolutional codes are discussed, as well as the graphical representation of low-density parity-check (LDPC) block codes in the form of Tanner graphs. Based on these representations different (suboptimal) decoding strategies are highlighted. The remaining material can be divided into two parts:

The first part (Chapter 3 and Chapter 4) is devoted to (generalized) LDPC block codes. In Chapter 3, algorithms for constructing quasi-cyclic LDPC block codes with large girth and for determining their minimum distance are presented. This chapter is largely based on [BHJ+10], [BHJ+11], [BHJK11c] and [BHJ+12]. Woven graph codes are introduced in Chapter 4 as a generalization of graph-based codes and the existence of such codes satisfying the Costello lower bound is proven. In particular, a rate $R = 5/20$ woven graph convolutional code with free distance as large as 120 is presented. These results appear in [HBJK10] and [BHJK11a].

The second part of this dissertation (Chapter 5 to Chapter 7) focuses on properties of convolutional codes. A recurrent equation to determine a closed form expression of the exact decoding bit error probability for convolutional codes is presented in Chapter 5. It is largely based on [BHJK11b], [BHJK11d], and [BHJK12b]. MacWilliams-type identities are revisited in Chapter 6 and a recursion for sequences of spectra of truncated as well as tailbitten convolutional codes and their duals is derived. Most of these results appear in [BHJK10] and [BHJK12a]. In Chapter 7, exhaustive searches are carried out for convolutional codes of various rates with either optimum free distance or optimum distance profile, extending previously published code tables. This chapter includes partly results from [JBHH11].

2

Graphs, Codes, and Codes on Graphs

Often it is useful for analyzing and understanding code properties and decoding methods to represent a code by its corresponding *graph*. Such a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is in general determined by a set of *vertices* or *nodes* $\mathcal{V} = \{\zeta_i\}$ and a set of *edges* or *branches* $\mathcal{E} = \{e_i\}$, where each edge connects exactly two vertices.

Forney [For67] introduced the trellis for convolutional codes to graphically describe a finite-state machine, which he used to prove the optimality of the *Viterbi algorithm* [Vit67] for ML sequence decoding of (terminated) convolutional codes. A similar trellis representation for linear block codes was later introduced by [BCJR74] and [Wol78].

The complexity of an exhaustive search for the ML code sequence is in general *exponential* in K, the length of its information sequence. On the other hand, using more advanced algorithms like the Viterbi algorithm, it is possible to find the same ML code sequence with a complexity which is only *linear* in K, but exponential in ν, the overall constraint length of the encoder, where $\nu \ll K$. Moreover, by studying graphical representations, the influence of various code properties can be analyzed more detailed, leading to improved decoding algorithms. An example of such an improved algorithm is the *Bidirectional Efficient Algorithm for Searching code Trees* (BEAST) [BHJK01] [BHJK04].

The concept of *bipartite graphs* was introduced by Tanner [Tan81] to describe the structure of linear block codes. Such graphs, also known as *Tanner graphs*, led to a new interpretation of LDPC codes, invented by Gallager [Gal62] [Gal63], from a graph theoretical point of view. Decoded iteratively, for example with the *belief propagation* (BP) decoding algorithm [Gal63] [Pea88], LDPC codes provide a better bit error rate (BER) performance than other block codes, given the same decoding complexity [RU08]. In particular, it

was shown that the BER of LDPC block codes with long block lengths N and being decoded with the BP algorithm can approach the theoretical Shannon limit closely [SFRU01].

In Section 2.1 and Section 2.2 we discuss the concepts of trees and trellises for convolutional codes and linear block codes. The Viterbi algorithm as an example of a trellis-based decoding algorithm is described in Section 2.3, while a more efficient tree-based decoding algorithm, the BEAST, is presented in Section 2.4. Section 2.5 focuses on LDPC block codes and their graphical representation as Tanner graphs. This chapter is concluded with Section 2.6, devoted to the BP algorithm, one of the most commonly used iterative decoding algorithms for LDPC block codes.

2.1 TREES AND TRELLISES FOR CONVOLUTIONAL CODES

The *state* of a system is defined as a description of its history, which, together with the present and future inputs, is sufficient to completely determine the present and future outputs of the system.

Consider a rate $R = b/c$ convolutional encoder realized in CCF with constraint lengths ν_i, $i = 0, 1, \ldots, b-1$, and overall constraint length ν. Such an encoder consists of ν memory elements, and hence its state space has size 2^ν. For example, its state at time instant t is represented by the ν-tuple $\boldsymbol{\sigma}_t$ of previous information symbols

$$\boldsymbol{\sigma}_t = \left(u_{t-1}^{(0)}\, u_{t-2}^{(0)} \ldots u_{t-\nu_0}^{(0)} \quad u_{t-1}^{(1)}\, u_{t-2}^{(1)} \ldots u_{t-\nu_1}^{(1)} \quad \ldots \quad u_{t-1}^{(b-1)}\, u_{t-2}^{(b-1)} \ldots u_{t-\nu_{(b-1)}}^{(b-1)} \right) \tag{2.1}$$

For a rate $R = 1/c$ convolutional encoder with memory m and realized in CCF, the state representation can be simplified to

$$\boldsymbol{\sigma}_t = (u_{t-1}\, u_{t-2} \ldots u_{t-m}) \tag{2.2}$$

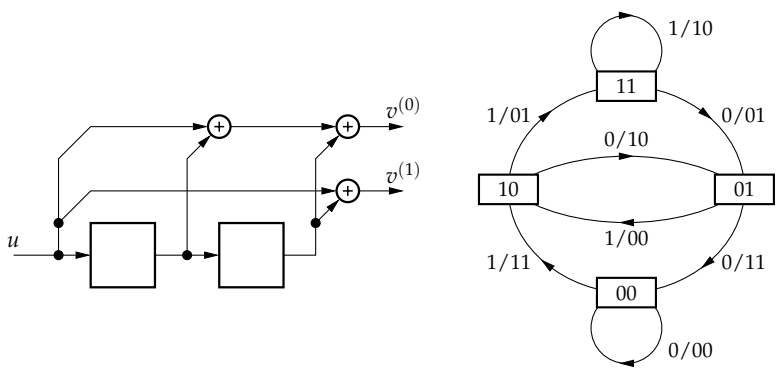


Figure 2.1: A convolutional encoder and its state transition diagram.

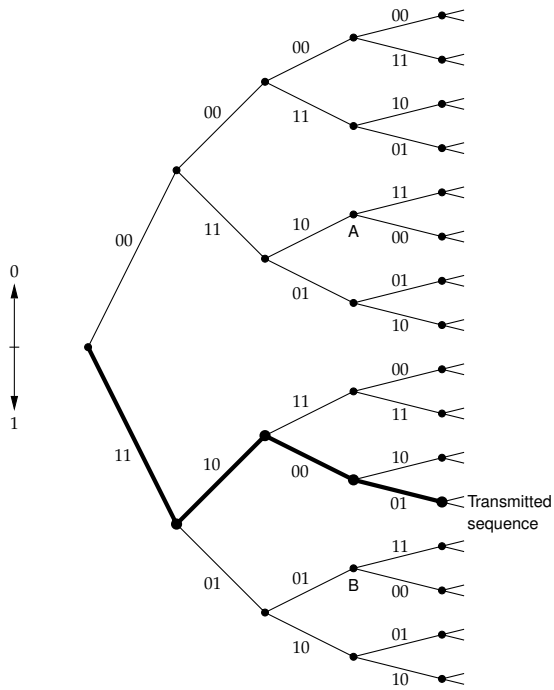


Figure 2.2: A rate $R = 1/2$ code tree.

The corresponding *state transition diagram* is a graphical representation of all possible encoder states $\boldsymbol{\sigma}$ together with their possible transitions from one state to another.

In general, the state transition diagram of a rate $R = b/c$ convolutional encoder with overall constraint length ν consists of 2^ν states, with 2^b branches leaving from and arriving at each node. A branch from state $\boldsymbol{\sigma}$ to state $\boldsymbol{\sigma}'$ is labeled by the information b-tuple $\boldsymbol{u}$ and code c-tuple $\boldsymbol{v}$ corresponding to the transition $\boldsymbol{\sigma} \to \boldsymbol{\sigma}'$; that is, each branch represents one of 2^b possible information b-tuples at every time instant.

■ **Example 2.1:**
Consider the rate $R = 1/2$ convolutional encoder determined by the encoding matrix $G(D) = (1 + D + D^2 \quad 1 + D^2)$. Its state transition diagram is illustrated together with its convolutional encoder in Figure 2.1.

An alternative representation is given by the *code tree*, which represents the set of code sequences by *paths* through the tree. The leftmost node of every such code tree is called the *root* and, starting from the root, there are 2^b branches stemming from each node. The branches leaving a node at *depth* (time instant) t are labeled with the code c-tuple $\boldsymbol{v}_t$.

■ **Example 2.1 (Cont'd):**
The corresponding code tree for the previously used convolutional encoder from Figure 2.1 is illustrated in Figure 2.2.

Since this encoder has only one binary input u, there are two branches leaving each state, where the upper and lower branch correspond to the information symbols 0 and 1, respectively. Following, for example, the path for the information sequence 1011 ... yields the corresponding code sequence 11 10 00 01 ....

Next, consider the two information sequences 010 and 110 which lead to the nodes A and B in Figure 2.2. While those sequences are different, both drive the encoder to the same encoder state $\sigma = 01$. Thus, the subtrees stemming from both nodes are identical and could, in principle, be merged together.

Replacing all equivalent nodes by a single node at each depth (time instant) of the code tree, we obtain a *trellis*-like structure. As before, the branches for the transitions $\sigma_t \to \sigma_{t+1}$ are labeled with the corresponding code c-tuples v_t. For a rate $R = b/c$ convolutional code with memory m and overall constraint length ν, there are 2^b branches leaving each node. Starting with the all-zero state at depth zero, the *root* of the trellis, it takes m time instants, the so-called *start-up phase*, until the trellis is fully developed and all 2^ν states have been reached. During the following *steady-state phase*, the trellis of a convolutional code has a regular structure with in total 2^ν states and 2^b branches leaving from and arriving at each state. As in the code tree, every path in the trellis corresponds to a possible code sequence.

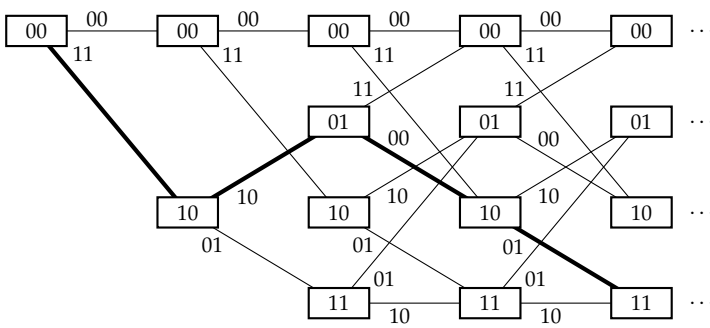


Figure 2.3: A binary trellis for a rate $R = 1/2$ convolutional code.

■ **Example 2.1 (Cont'd):**
Merging all equivalent nodes in the code tree in Figure 2.2 yields the corresponding trellis as illustrated in Figure 2.3, where in our case the upper and lower branches correspond to the information symbols 0 and 1, respectively. As before, following the path for the information sequence 1011 ... in the trellis, yields the corresponding code sequence 11 10 00 01 ....

Note that in case of a convolutional code with rate $R = b/c$ and $b > 1$, an information b-tuple enters the encoder at every time instant, which makes it necessary to further distinguish different branches. In such cases, the corresponding branches are additionally labeled with their information b-tuple $\boldsymbol{u}_t$.

Besides creating the previously described trellis for a convolutional code $\mathcal{C}$ based on its generator matrix $G(D)$ (cf. Figure 2.3), an equivalent trellis, the so-called *syndrome trellis* [SZ94], can be obtained from its corresponding polynomial parity-check matrix $H(D)$.

Consider a rate $R = b/c$ convolutional code $\mathcal{C}$ with overall constraint length ν and parity-check matrix $H(D)$ with syndrome memory m_s. Its semi-infinite parity-check matrix H in the time domain is given by (1.69) as

$$H = \begin{pmatrix} H_0 & & & \\ H_1 & H_0 & & \\ \vdots & H_1 & H_0 & \\ H_{m_\mathrm{s}} & \vdots & H_1 & \ddots \\ & H_{m_\mathrm{s}} & \vdots & \ddots \\ & & H_{m_\mathrm{s}} & \ddots \\ & & & \ddots \end{pmatrix} \tag{2.3}$$

where the $m_\mathrm{s} + 1$ binary submatrices H_i, $i = 0, 1, \ldots, m_\mathrm{s}$, are determined by (1.68), and H_0 is nonsingular with full rank $c - b$. Such a semi-infinite parity-check matrix is completely described by its *parity-check matrix module* $\widetilde{H}$ of size $(m_\mathrm{s} + 1)(c - b) \times c$ given by

$$\widetilde{H} = \begin{pmatrix} H_0 \\ H_1 \\ \vdots \\ H_{m_\mathrm{s}} \end{pmatrix} \tag{2.4}$$

whose binary columns are denoted by $\widetilde{\boldsymbol{h}}_i$, $i = 0, 1, \ldots, c - 1$.

For a rate $R = (c - 1)/c$ convolutional code, the parity-check module is of size $(m_\mathrm{s} + 1) \times c$ and each of its c columns $\widetilde{\boldsymbol{h}}_i$, $i = 0, 1, \ldots, c - 1$, is a binary vector of length $m_\mathrm{s} + 1$, corresponding to the parity-check polynomial $\boldsymbol{h}_i(D)$, written in binary notation starting from the top, that is,

$$\widetilde{\boldsymbol{h}}_i = (h_{i,0}\, h_{i,1} \ldots h_{i,m_\mathrm{s}})^\mathrm{T} \tag{2.5}$$

whose elements are determined by

$$\boldsymbol{h}_i(D) = \sum_{j=0}^{m_\mathrm{s}} h_{i,j} D^j \tag{2.6}$$

Following [Var98], the syndrome trellis for a rate $R = b/c$ convolutional code $\mathcal{C}$ can be constructed by connecting the identically syndrome subtrellises of the parity-check matrix module $\widetilde{H}$ and removing all paths that neither

start nor end in the all-zero encoder state. Such a syndrome subtrellis can be either *conventional* or *binary* (c-sectionalized), which shall both be described for rate $R = (c-1)/c$ convolutional codes in the following. Their generalization to rate $R = b/c$ convolutional codes is straightforward.

The conventional syndrome subtrellis for a rate $R = (c-1)/c$ convolutional code $\mathcal{C}$ with syndrome memory m_s consists of two trellis levels. Each trellis level contains 2^{m_s} different states $\boldsymbol{\sigma}$ with 2^{c-1} trellis branches leaving from and arriving at each state. Moreover, let a state $\boldsymbol{\sigma}$ be given as the binary m_s-tuple

$$\boldsymbol{\sigma} = (\sigma_0\, \sigma_1 \ldots \sigma_{m_\mathrm{s}-1}) \tag{2.7}$$

and let every branch be labeled by the binary c-tuple

$$\boldsymbol{v}_t = \left(v_t^{(0)}\, v_t^{(1)} \ldots v_t^{(c-1)}\right) \tag{2.8}$$

Then the state $\boldsymbol{\sigma}$ at the first trellis level is connected by a branch to the state $\boldsymbol{\sigma}'$ at the second trellis level and labeled by the c-tuple $\boldsymbol{v}_t$, such that

$$\begin{cases} \sigma_0 + \sum_{i=0}^{c-1} v_t^{(i)} h_{i,0} = 0 \\ \sigma_j + \sum_{i=0}^{c-1} v_t^{(i)} h_{i,j} = \sigma'_{j-1} \qquad j = 1, 2, \ldots, m_\mathrm{s} - 1 \\ \sum_{i=0}^{c-1} v_t^{(i)} h_{i,m_\mathrm{s}} = \sigma'_{m_\mathrm{s}-1} \end{cases} \tag{2.9}$$

The corresponding binary syndrome subtrellis for a rate $R = (c-1)/c$ convolutional code is specified by using only two branches arriving at and leaving from each state. This simplification, however, comes at the cost of $c-1$ additional intermediate layers in each subtrellis, where each intermediate layer may consists of as many as $2^{m_\mathrm{s}+1}$ different states.

■ **Example 2.2:**
Consider the rate $R = 2/3$ convolutional code with the polynomial parity-check matrix

$$H(D) = \begin{pmatrix} 1+D^2 & 1+D & 1+D+D^2 \end{pmatrix} \tag{2.10}$$

and syndrome memory $m_\mathrm{s} = 2$. Its $m_\mathrm{s} + 1 = 3$ binary submatrices are

$$H_0 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \qquad H_1 = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix} \qquad H_2 = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$$

and hence its parity-check matrix module follows as

$$\widetilde{H} = \begin{pmatrix} H_0 \\ H_1 \\ H_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \widetilde{h}_0 & \widetilde{h}_1 & \widetilde{h}_2 \end{pmatrix} \tag{2.11}$$

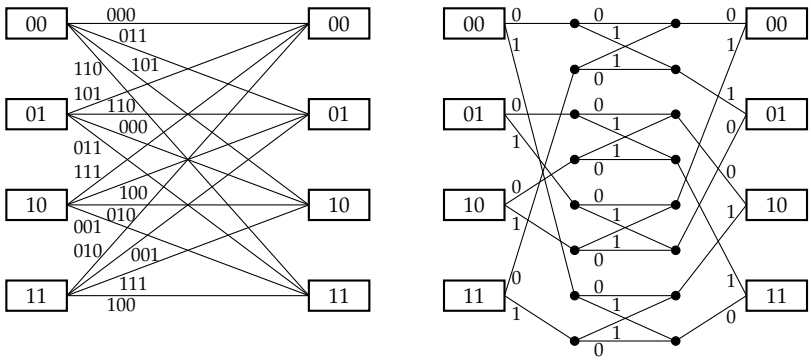


Figure 2.4: Conventional (left) and binary (right) syndrome trellis module for the rate $R = 2/3$ convolutional syndrome former used in Example 2.2.

Its conventional syndrome subtrellis is illustrated on the left side in Figure 2.4, where the state $\boldsymbol{\sigma} = 00$ at the first trellis level is connected by a branch with labeling $\boldsymbol{v}_t = 101$ to the state $\boldsymbol{\sigma}' = 10$ at the second trellis level, since it satisfies

$$\begin{cases} \sigma_0 + v_t^{(0)} h_{0,0} + v_t^{(1)} h_{0,1} + +v_t^{(2)} h_{0,2} = 0 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 = 0 \\ \sigma_1 + v_t^{(0)} h_{1,0} + v_t^{(1)} h_{1,1} + +v_t^{(2)} h_{1,2} = 0 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 = 1 = \sigma_0' \\ \qquad v_t^{(0)} h_{2,0} + v_t^{(1)} h_{2,1} + +v_t^{(2)} h_{2,2} = \qquad 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 = 0 = \sigma_1' \end{cases}$$

Introducing $c - 1 = 2$ additional intermediate layers yields the corresponding binary syndrome subtrellis as shown on the right side in Figure 2.4, where only two branches leave from and arrive at each state.

As mentioned before, a convolutional code $\mathcal{C}$ is defined as the set of all its code sequences $\boldsymbol{v}$; that is, the set of all possible paths within its trellis. The $(i+1)$th *Viterbi spectral component* of a convolutional encoder is denoted by $n_{d_{\text{free}}+i}$ and defines the number of paths with Hamming weight $d_{\text{free}} + i$ which depart from the all-zero path in the root of the code trellis and do not reach the all-zero encoder state until their termini. The infinite sequence of the number of code sequences with Hamming weights $d_{\text{free}} + i$, that is,

$$n_{d_{\text{free}}+i}, \quad i = 0, 1, \ldots \tag{2.12}$$

is called the *Viterbi spectrum* (or *free distance spectrum*) of a convolutional encoder in order to distinguish it from the weight spectrum of block codes. Note that the Viterbi spectrum is an encoder property (cf. Example 6.1). The corresponding generating function

$$T(W) = \sum_{i=0}^{\infty} n_{d_{\text{free}}+i} W^{d_{\text{free}}+i} \tag{2.13}$$

is called the *path weight enumerator*.

(a) Truncation **(b)** Zero-tail termination **(c)** Tailbiting

Figure 2.5: Schematic illustrations of different termination methods for trellises based on generator matrices.

Every convolutional encoder encodes an information sequence of (theoretically) infinite length, and hence the steady state phase of its trellis continues indefinitely. However, by restricting the length of a convolutional code, that is, by limiting its steady state phase, we obtain a corresponding block code with similar properties (cf. Subsection 1.3.2).

From the point of view of a trellis constructed from a convolutional generator matrix, truncation to length M corresponds to limiting the length of the steady state phase and ending the encoding process at any of the 2^ν possible states. However, the last m information b-tuples are not fully encoded, and hence less protected. Zero-tail termination adds m additional »dummy« b-tuples to fully encode the information sequence, which, however, leads to a slight rate loss. The last m trellis sections of such a zero-tail terminated code contain only transitions which lead towards the terminating all-zero encoder state, the *toor* (toor = root backwards) of the trellis [Mas78]. When using tailbiting, the starting and ending state of the trellis have to be identical, but not necessarily the all-zero state. Hence, a tailbiting trellis of length M contains all those paths of the code trellis, which start at any of the 2^ν states at time instant 0 and end after M branches in the same state as they started in.

The corresponding trellises for all three termination methods are schematically illustrated in Figure 2.5. Note that these illustrations are only valid for trellises constructed from convolutional generator matrices; not for the corresponding syndrome trellises.

2.2 TREES AND TRELLISES FOR LINEAR BLOCK CODES

Similar to convolutional codes, a rate $R = K/N$ linear block code can be represented by a trellis. Contrary to the trellis of a convolutional code, the trellis of a linear block code has a fixed length N and a *time-varying* trellis structure. A method for constructing such a code trellis based on its parity-check matrix H was introduced in [BCJR74] and further analyzed in [Wol78].

Consider for example the $(N-K) \times N$ parity-check matrix H of an (N, K) linear block code and denote its ith column by the size $N-K$ binary column vector $\boldsymbol{h}_i$, $i = 0, 1, \ldots, N-1$,. Moreover, let the set of all states $\boldsymbol{\sigma}$ at depth t, $t = 0, 1, \ldots, N$ be given by $\mathcal{I}_t$, where states at adjoined depths are connected by separate branches.

Then the trellis of a linear block code can be defined as

$$\mathcal{I}_0 = \{\boldsymbol{\sigma}_{\text{root}} = \mathbf{0}\} \tag{2.14}$$

$$\mathcal{I}_{t+1} = \left\{\boldsymbol{\sigma} \;\middle|\; \boldsymbol{\sigma} = \boldsymbol{\sigma}' + v_t\, \boldsymbol{h}_t^{\mathrm{T}},\;\; v_t \in \{0,1\},\;\; \boldsymbol{\sigma}' \in \mathcal{I}_t\right\} \quad t = 0, 1, \ldots, N-1 \tag{2.15}$$

where v_t denotes the tth binary code symbol of the codeword $\boldsymbol{v}$ of length N corresponding to a specific path, chosen such that it terminates at the all-zero state at trellis depth N, that is, $\mathcal{I}_N = \{\boldsymbol{\sigma} = \mathbf{0}\}$. In other words, every state $\boldsymbol{\sigma} \in \mathcal{I}_t$ at trellis depth t represents the *partial syndrome* for the path $\boldsymbol{\sigma}_{\text{root}} \to \boldsymbol{\sigma}$ with the corresponding partial codeword $(v_0\, v_1 \ldots v_{t-1})$.

The *state complexity* of the trellis at depth t follows as $\eta_t = \log_2 |\mathcal{I}_t|$, where the $(N+1)$-tuple

$$\boldsymbol{\eta} = (\eta_0\, \eta_1 \ldots \eta_N) \tag{2.16}$$

is called the *state complexity profile*, with *maximum state complexity*

$$\eta_{\max} = \max_t \{\eta_t\} \tag{2.17}$$

■ **Example 2.3:**
Consider the $(6,3)$ linear block code $\mathcal{B}$ with parity-check matrix

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.18}$$

obtained by removing the all-one column from the parity-check matrix of the $(7,4)$ Hamming code used in Example 1.3. The corresponding trellis diagram for the $(6,3)$ *shortened Hamming code* is illustrated in Figure 2.6 and has the state complexity profile $\boldsymbol{\eta} = (0\,1\,2\,2\,2\,1\,0)$.

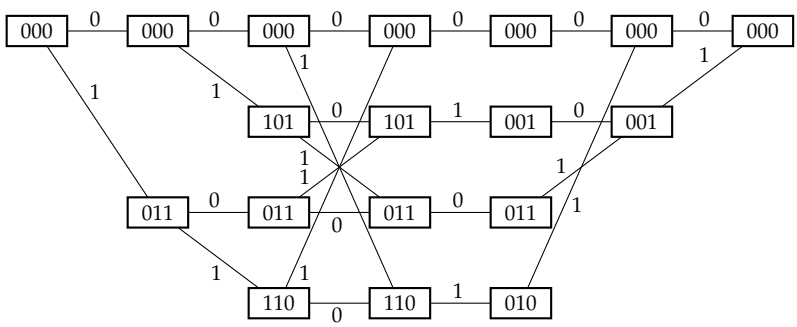


Figure 2.6: Trellis diagram of the $(6,3)$ shortened Hamming code.

The complexity of trellis-based algorithms like the Viterbi algorithm (cf. Section 2.3) is exponential to the maximum state complexity. Hence, finding the trellis with the smallest maximum state complexity is of particular interest. If the state complexity η_t at every depth of the bit-level[1] trellis $\mathcal{T}$ for a block code is smaller than or equal to the state complexity η'_t of any other bit-level trellis $\mathcal{T}'$ for the same block code, we refer to the trellis $\mathcal{T}$ as the *minimal trellis* for this block code [For88] [Mud88].

Such a minimal trellis can be constructed based on a generator matrix in its *trellis-oriented form* [For88] [KS95], also known as its *minimal-span form* [McE96]. In particular, using elementary row operations, every generator matrix can be reduced to its minimal-span form. Thus, every linear block code has a minimal trellis representation, which is moreover *unique* up to isomorphism.

Definition 2.1 (Minimal-Span Form [KS95] [McE96])
Let $\boldsymbol{g}_i$, $i = 0, 1, \ldots, K-1$ be the ith row of the generator matrix G of an (N, K) linear block code. Denote by $\texttt{start}(\boldsymbol{g}_i)$ and $\texttt{end}(\boldsymbol{g}_i)$ the first and the last nonzero position within the row $\boldsymbol{g}_i$, where $0 \leq \texttt{start}(\boldsymbol{g}_i) \leq \texttt{end}(\boldsymbol{g}_i) < N$, respectively. Then a generator matrix is said to be in its minimal-span form if

$$\texttt{start}(\boldsymbol{g}_i) \neq \texttt{start}(\boldsymbol{g}_j) \qquad \texttt{end}(\boldsymbol{g}_i) \neq \texttt{end}(\boldsymbol{g}_j) \tag{2.19}$$

where $i \neq j$, $i, j = 0, 1, \ldots, K-1$. The *span* of the row $\boldsymbol{g}_i$ refers to the interval $[\texttt{start}(\boldsymbol{g}_i), \texttt{end}(\boldsymbol{g}_i)]$, while the interval $[\texttt{start}(\boldsymbol{g}_i), \texttt{end}(\boldsymbol{g}_i) - 1]$ is denoted its *active interval*. Moreover, the row $\boldsymbol{g}_i$ is said to be *active* at position j if $j \in [\texttt{start}(\boldsymbol{g}_i), \texttt{end}(\boldsymbol{g}_i) - 1]$.

Based on Gaussian elimination, the minimal-span form of a generator matrix can be obtained as follows:

Algorithm MS (Minimal-span form [KS95])

1. Reduce the generator matrix G by Gauss elimination to its *row echelon form* (REF), where the leading coefficient of every row is always strictly to the right of the leading coefficient of the row above. Hence, every row has a unique starting position.
2. Perform a second Gaussian elimination »bottom up« to obtain unique ending positions, that is, start from the last element in the last row and iterate upwards, such that the unique starting positions remain unchanged.

[1] A bit-level trellis describes a trellis, where every branch between two states is labeled with a single output bit. Using *sectionalizing*, trellises with several output bits on every branch can be obtained, reducing in general the corresponding state complexity. In the following we will, however, focus only on bit-level trellises.

The active interval of row g_i determines the time instants at which the corresponding information symbol u_i influences the encoder state. Hence, the state complexity η_{t+1} at depth $t+1$, $t = 0, 1 \ldots, N-1$, corresponds directly to the number of active rows in column t of the minimal-span generator matrix.

Denote by $\mathcal{A}_t$ the set of rows of a minimal-span generator matrix, which are active in column t. Then the set of all states $\mathcal{S}_{t+1}$ at depth $t+1$ of the bit-level trellis is defined as

$$\mathcal{S}_0 = \{\sigma_{\text{root}} = \mathbf{0}\} \tag{2.20}$$

$$\mathcal{S}_{t+1} = \{\sigma \mid \sigma = (\sigma_0\, \sigma_1 \ldots \sigma_{K-1})\} \quad t = 0, 1, \ldots, N-1 \tag{2.21}$$

where the state σ is a K-tuple with

$$\sigma_i = \begin{cases} u_i & \text{if } g_i \in \mathcal{A}_t \\ 0 & \text{otherwise} \end{cases} \quad i = 0, 1, \ldots, K-1 \tag{2.22}$$

According to Definition 2.1 at most a single row becomes active in column t, $t = 0, 1, \ldots, N-1$. Denote such a row (if it exists) by $g_{\texttt{start}^{-1}(t)}$ and the corresponding column t of the minimal-span generator matrix G by γ_t. Then the branch label v_t for the transition between the two states $\sigma \in \mathcal{S}_t$ and $\sigma' \in \mathcal{S}_{t+1}$ in the bit-level trellis is determined by

$$v_t = \begin{cases} \sigma\gamma_t + u_{\texttt{start}^{-1}(t)} & \text{if } g_{\texttt{start}^{-1}(t)} \text{ exists} \\ \sigma\gamma_t & \text{otherwise} \end{cases} \tag{2.23}$$

We note that the start and the end of an active row correspond directly to either a branch expansion or a branch merge in the bit-level trellis. To be more precise, a minimal bit-level trellis satisfies the following properties:

(i) if there exists a row g with $\texttt{start}(g) = t$, then two branches stem from each state at depth t and are labeled by opposite code symbols

(ii) if there exists a row g with $\texttt{end}(g) = t$, then two branches merge into a single state at depth $t+1$

(iii) if there exists no row g with either $\texttt{start}(g) = t$ or $\texttt{end}(g) = t$, then every state at depth t is connected to exactly one state at depth $t+1$

(iv) if there exists a row g with $\texttt{start}(g) = \texttt{end}(g) = t$ (a *trivial* span), then every state at depth t is connected by two parallel branches, labeled with opposite code symbols, to the same state at depth $t+1$

Note that the state complexity profile varies among equivalent codes. In particular, reordering the columns of the generator matrix G might yield an equivalent block code with an improved state complexity profile [Mas78] [For88] [Mud88].

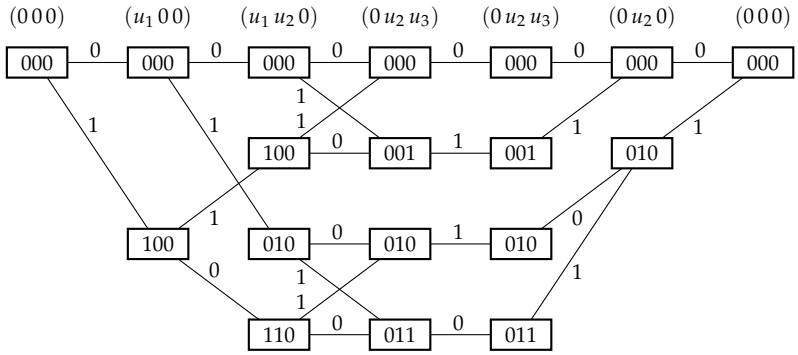


Figure 2.7: Minimal trellis for the $(6, 3, 3)$ shortened Hamming code.

The trellis with the best state complexity profile among all equivalent block codes is called the *absolute minimal trellis* [BB93]. Finding such an absolute minimal trellis is however an NP-hard problem.

Finally, by »unfolding« the bit-level trellis, that is, by ignoring the mergers that occur in the code trellis, we obtain the corresponding *code tree*. Similarly to a code trellis, there are two branches stemming from each state at depth t if there exists a row $\boldsymbol{g}$ with $\texttt{start}(\boldsymbol{g}) = t$; otherwise only one. However, unlike the previously discussed code trees for convolutional codes (cf. Section 2.1), a code tree for a linear block code has a time-varying structure.

■ **Example 2.3 (Cont'd):**
Consider the systematic generator matrix G for the $(6, 3)$ linear block code

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{2.24}$$

with minimum distance $d_{\min} = 3$. Since this generator matrix is already given in RREF, only the second step of the minimal-span form algorithm has to be applied to ensure unique endings in each row. Adding the second and third row to the first row, yields the generator matrix in its minimal-span form

$$G_{\mathrm{MS}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{2.25}$$

where the active interval for each row is shaded in gray. The corresponding minimal trellis is illustrated in Figure 2.7 and has the state complexity profile $\boldsymbol{\eta} = (0\,1\,2\,2\,2\,1\,0)$, where the number of active rows in column t of its minimal-span generator matrix corresponds to the state complexity η_{t+1}, $t = 0, 1, \ldots, 5$.

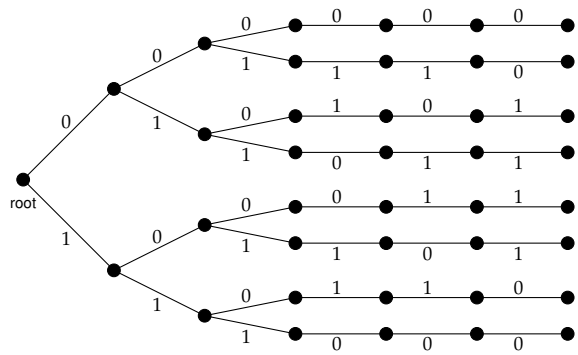


Figure 2.8: Forward code tree for the $(6,3,3)$ shortened Hamming code starting with the root node at depth 0.

An active interval starts in each of the columns $i = 0$, 1, and 2, corresponding to a branch expansion at depth i. Similar, these active intervals end in either column $j = 2$, 4, or 5, and hence at each depth $j + 1$ two branches are merged together. In particular, this trellis is identical up to a permutation to the one given in Figure 2.6, which is based on the corresponding parity-check matrix H.

By »unfolding« the minimal trellis, that is, by ignoring all state mergers, the corresponding code tree for the $(6,3,3)$ shortened Hamming code is obtained and illustrated in Figure 2.8. Such a code tree is also known as a *forward code tree*, where all paths stem from the root node at depth 0 and progress forward. On the other hand, if we start from the end of the minimal trellis, the toor node at depth N, and »unfold« the trellis backwards, we obtain a so-called *backward code tree* as shown in Figure 2.9. Note that a state expansion in a forward code tree corresponds to a state merger in the corresponding backward code tree and vice versa.

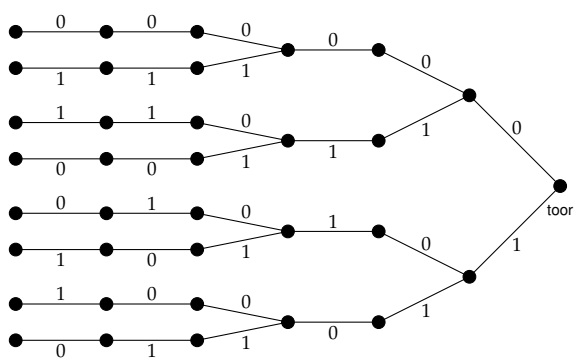


Figure 2.9: Backward code tree for the $(6,3,3)$ shortened Hamming code starting with the toor node at depth 6.

2.3 THE VITERBI ALGORITHM

The Viterbi algorithm [Vit67] is an ML sequence decoding algorithm for block and convolutional codes based on their trellis representation. Given a received sequence $\boldsymbol{r}$, an ML sequence decoding algorithm decides in favor of a certain information sequence $\boldsymbol{u}$ whose code sequence $\boldsymbol{v}$ is most probable, assuming equiprobable information and code sequences. That is, depending on the output-alphabet of the underlying transmission channel, an ML decoding algorithm maximizes the conditional probability $P(\boldsymbol{r} \mid \boldsymbol{v})$ in case of a discrete output-alphabet (like the BSC or the quantized AWGN channel) or the corresponding conditional PDF $p(\boldsymbol{r} \mid \boldsymbol{v})$ in case of a continuous output-alphabet (like the unquantized AWGN channel).

Considering only memoryless transmission channels with a discrete output-alphabet, the conditional probability can be expressed as

$$P(\boldsymbol{r} \mid \boldsymbol{v}) = \prod_t P(\boldsymbol{r}_t \mid \boldsymbol{v}_t) = \prod_t \prod_k P\left(r_t^{(k)} \,\middle|\, v_t^{(k)}\right) \tag{2.26}$$

Taking the logarithm, the ML decoding rule can be formulated as the decision in favor of a certain information sequence $\boldsymbol{u}$, whose trellis path maximizes the *cumulative Viterbi branch metric*[2]

$$\mu_{\mathrm{V}}(\boldsymbol{r}, \boldsymbol{v}) = \log P(\boldsymbol{r} \mid \boldsymbol{v}) = \sum_t \mu_{\mathrm{V}}(\boldsymbol{r}_t, \boldsymbol{v}_t) \tag{2.27}$$

where the *branch metric* for time instant t is given by

$$\mu_{\mathrm{V}}(\boldsymbol{r}_t, \boldsymbol{v}_t) = \sum_k \mu_{\mathrm{V}}\left(r_t^{(k)}, v_t^{(k)}\right) \tag{2.28}$$

Following [Mas84], the individual metric increments $\mu_{\mathrm{V}}(r_t^{(k)}, v_t^{(k)})$ can be expressed as

$$\mu_{\mathrm{V}}\left(r_t^{(k)}, v_t^{(k)}\right) = A\left(\log P\left(r_t^{(k)} \,\middle|\, v_t^{(k)}\right) - f_t^{(k)}\left(r_t^{(k)}\right)\right) \tag{2.29}$$

where A is a positive scaling factor and f denotes an arbitrarily chosen function. For convenience, the function f is often chosen as

$$f_t^{(k)}\left(r_t^{(k)}\right) = \min_{v_t^{(k)}} \left\{\log P\left(r_t^{(k)} \,\middle|\, v_t^{(k)}\right)\right\} \tag{2.30}$$

[2]Here the term »metric« does not necessarily satisfy the metric conditions in a strict mathematical sense. Within coding theory, a metric is commonly used somewhat loosely to describe a distance measure.

assuming that the minimum exists. For example, in case of a BSC with crossover probability p it is convenient to choose

$$f_t^{(k)}\left(r_t^{(k)}\right) = \log p \qquad \text{and} \qquad A = -\left(\log \frac{p}{1-p}\right)^{-1} \tag{2.31}$$

such that the calculation of the individual metric increments is simplified to

$$\mu_{\mathrm{V}}\left(r_t^{(k)}, v_t^{(k)}\right) = 1 - d_{\mathrm{H}}\left(r_t^{(k)}, v_t^{(k)}\right) = \begin{cases} 1, & \text{if } r_t^{(k)} = v_t^{(k)} \\ 0, & \text{otherwise} \end{cases} \tag{2.32}$$

For a memoryless unquantized AWGN channel with BPSK modulation, the derivations of the cumulative Viterbi branch metric (2.27) and the individual metric increments (2.28) are still valid, if the probability $P(\boldsymbol{r} \mid \boldsymbol{v})$ is replaced by the corresponding PDF $p(\boldsymbol{r} \mid \boldsymbol{v})$.

Moreover, since there exists a unique mapping between every code sequence $\boldsymbol{v}$ and its modulated and transmitted sequence $\boldsymbol{x}$, it holds that

$$p(\boldsymbol{r} \mid \boldsymbol{v}) = p(\boldsymbol{r} \mid \boldsymbol{x}) = \prod_t \prod_k p\left(r_t^{(k)} \;\middle|\; x_t^{(k)}\right)$$

where the likelihood for the kth received symbol $r_t^{(k)}$ at time instant t, conditioned on the corresponding transmitted symbol $x_t^{(k)}$ follows from (1.8) as

$$\log p\left(r_t^{(k)} \;\middle|\; x_t^{(k)}\right) = \log\left(\frac{1}{\sqrt{\pi N_0}} e^{-\frac{1}{N_0}\left(r_t^{(k)} - x_t^{(k)}\right)^2}\right) \tag{2.33}$$

Using the squared Euclidean distance $\|\boldsymbol{r} - \boldsymbol{x}\|^2$, that is,

$$\|\boldsymbol{r} - \boldsymbol{x}\|^2 = \sum_t \sum_k \left(r_t^{(k)} - x_t^{(k)}\right)^2 \tag{2.34}$$

as a distance measure, the individual Viterbi metric increments $\mu_{\mathrm{V}}\left(r_t^{(k)}, v_t^{(k)}\right)$ in (2.28) are determined by

$$\mu_{\mathrm{V}}\left(r_t^{(k)}, v_t^{(k)}\right) = -\left(r_t^{(k)} - x_t^{(k)}\right)^2 \tag{2.35}$$

However, note that the (nonsquared) Euclidean distance does neither satisfy (2.27) nor (2.28), and hence can not be used as a distance measure for the Viterbi ML sequence decoding algorithm.

On the other hand, in case of a correlation decoder (1.94) or weighted Hamming distance decoder (1.98), the corresponding Viterbi metric increments are given by

$$\mu_{\mathrm{V}}\left(r_t^{(k)}, v_t^{(k)}\right) = r_t^{(k)} x_t^{(k)} \tag{2.36}$$

which, for BPSK modulation, is equivalent to

$$\mu_{\mathrm{V}}\left(r_t^{(k)}, v_t^{(k)}\right) = \begin{cases} -\left|r_t^{(k)}\right| & \text{if } \operatorname{sign}\left(r_t^{(k)}\right) = \operatorname{sign}\left(x_t^{(k)}\right) \\ 0 & \text{otherwise} \end{cases} \tag{2.37}$$

Finally, for any given sequence $\boldsymbol{y}$, denote by $\boldsymbol{y}_{[i,j-1]} = \boldsymbol{y}_{[i,j)}$, $0 \leq i < j$, its segment $(\boldsymbol{y}_i\ \boldsymbol{y}_{i+1} \ldots \boldsymbol{y}_{j-1})$. Then, the path metric for time instant t, $t = 0, 1, \ldots$, can be recursively computed according to (2.27) using the branch metrics (2.28), that is,

$$\mu_{\mathrm{V}}\left(\boldsymbol{r}_{[0,t+1)}, \boldsymbol{v}_{[0,t+1)}\right) = \mu_{\mathrm{V}}\left(\boldsymbol{r}_{[0,t)}, \boldsymbol{v}_{[0,t)}\right) + \mu_{\mathrm{V}}\left(\boldsymbol{r}_t, \boldsymbol{v}_t\right) \tag{2.38}$$

where the initial value is formally defined as

$$\mu_{\mathrm{V}}\left(\boldsymbol{r}_{[0,0)}, \boldsymbol{v}_{[0,0)}\right) = 0 \tag{2.39}$$

Algorithm V (Viterbi)

1. For a given code, construct its trellis and assign the cumulative Viterbi branch metric $\mu_{\mathrm{V}} = 0$ to its stating state at trellis depth $t = 0$.

2. For each state σ_{t+1} at trellis depth $t+1$, determine the cumulative Viterbi branch metric $\mu_{\mathrm{V}}(\boldsymbol{r}_{[0,t+1)}, \boldsymbol{v}_{[0,t+1)})$ for all paths arriving at this state σ_{t+1} as the sum of the cumulative Viterbi branch metric $\mu_{\mathrm{V}}(\boldsymbol{r}_{[0,t)}, \boldsymbol{v}_{[0,t)})$ stored in the state of each of its predecessors σ_t and the branch metric increment $\mu_{\mathrm{V}}(\boldsymbol{r}_t, \boldsymbol{v}_t)$ associated with the connecting transition $\sigma_t \to \sigma_{t+1}$ (ADD). Find the *maximum* among these metrics (COMPARE), assign the resulting value to the state σ_{t+1}, and label it with the shortest path to it (SELECT).

3. If the end of the trellis (the toor) is reached, output as the decided codeword $\hat{\boldsymbol{v}}$ (one of) the path(s) with largest Viterbi metric which terminates at the toor node. This is (one of) the ML path(s). Otherwise, increment t by 1 and continue with step **V-2**.

Note that the starting state of a tailbiting block code is not known beforehand, and thus the Viterbi decoding algorithm has to be applied for each possible pair of starting and ending states, $\sigma_{\mathrm{root}} = \sigma_{\mathrm{toor}}$, that is, 2^{ν} times. Then, the overall ML decision is determined by the Viterbi decoding algorithm as the trellis path with the largest cumulative Viterbi metric.

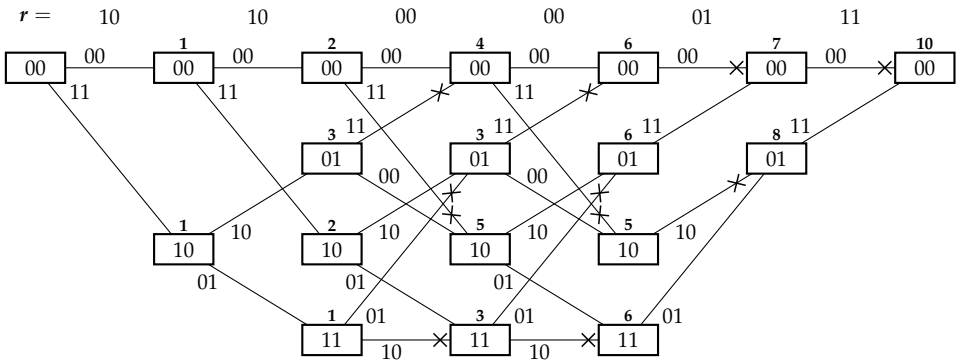


Figure 2.10: A binary trellis for a rate $R = 1/2$ convolutional code.

■ **Example 2.4:**
Consider the rate $R = 1/2$, memory $m = 2$ convolutional code with generator matrix $G(D) = (1 + D + D^2 \quad 1 + D^2)$, used to communicate over a BSC with crossover probability p. Using zero-tail termination, the four information symbols $\boldsymbol{u} = 1011$, followed by $m = 2$ zero »dummy« symbols are encoded to form the codeword $\boldsymbol{v} = (11\ 10\ 00\ 01\ 01\ 11)$ which is transferred over the BSC. The corresponding trellis for this zero-tail terminated convolutional code is illustrated in Figure 2.10.

Assuming two transmission errors, one at the second and one at the eighth code symbol, yields the received sequence

$$\boldsymbol{r} = (1\underline{0}\ 10\ 00\ 0\underline{0}\ 01\ 11)$$

where the erroneous positions are underlined. At every trellis state, we determine the cumulative Viterbi branch metrics for all subpaths leading to this state, but discard all except the one with the largest cumulative Viterbi branch metric. In case of a tie, we commonly use *coin-flipping* as a tie-breaker, that is, one of the subpaths with largest cumulative Viterbi branch metric is randomly chosen. The Viterbi metrics and the discarded subpaths are illustrated in Figure 2.10. The decided ML codeword after six trellis sections is $\hat{\boldsymbol{v}} = (11\ 10\ 00\ 01\ 01\ 11)$ whose information sequence is $\hat{\boldsymbol{u}} = (1\,0\,1\,1)$, that is, both transmission errors have been corrected.

The individual steps of the Viterbi algorithm are illustrated in Figure 2.11, where at each state, the surviving subpath is marked in bold, while the discarded subpath is dashed. Next to each state, the cumulative Viterbi branch metric of the surviving subpath is given, together with the metric of the discarded subpath in parenthesis (if applicable). After processing all six trellis sections, the Viterbi algorithm decides in favor of the information sequence $\hat{\boldsymbol{u}}$, whose corresponding path results in the largest cumulative Viterbi branch metric at the end of the trellis.

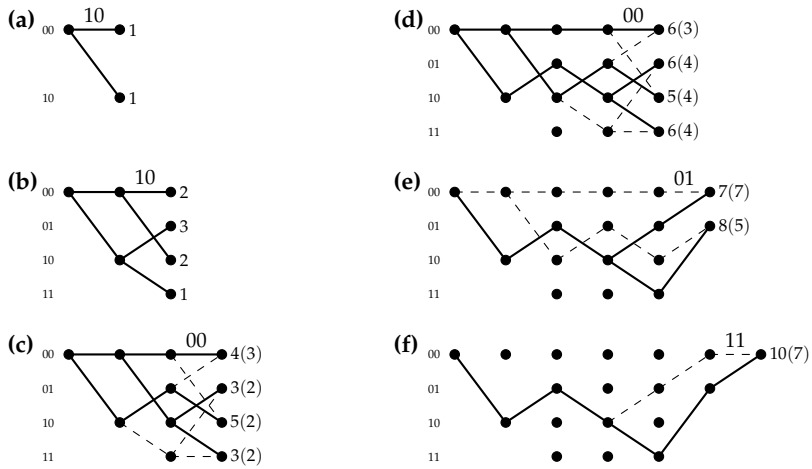


Figure 2.11: Development of subpaths through the trellis.

Consider for example Figure 2.11(c). After the startup phase, two branches arrive at each state σ. For example, at encoder state $\sigma = 00$, the two arriving subpaths $00 \to 10 \to 01 \to 00$ and $00 \to 00 \to 00 \to 00$ have the cumulative Viterbi branch metrics $\mu_{\mathrm{V}} = 3$ and $\mu_{\mathrm{V}} = 4$, respectively, and hence the first subpath is discarded in favor of the second one.

2.4 THE BEAST

The BEAST, a Bidirectional Efficient Algorithm for Searching code Trees, was introduced in [BHJK01] and [BHJK04] as an efficient algorithm for finding the weight and Viterbi spectrum of block and convolutional codes, respectively. Since searching for the number of code sequences with a certain Hamming weight resembles finding the closest code sequence to a given received sequence, the BEAST was extended in [BJKL04] and [BHJK05] to decoding of block codes.

2.4.1 FINDING THE WEIGHT SPECTRUM

Consider a rate $R = K/N$ binary block code, whose codewords are binary N-tuples $\boldsymbol{v} = (v_0\, v_1 \ldots v_{N-1})$. Such a code can be described either by a forward code tree of length N starting from the root node (at depth 0) and going forward, or by a backward code tree starting from the toor node (at depth N) and going backward (cf. Figures 2.8 and 2.9). In the following, we shall distinguish between the forward and the backward code tree by the subscripts F and B, respectively.

Every node ζ in such a code tree has a unique parent node ζ^{P}, at most two children nodes ζ^{C}, and is characterized by three parameters: its state $\boldsymbol{\sigma}(\zeta)$, weight $\omega(\zeta)$, and depth $\ell(\zeta)$. Its depth is equal to the length (in branches) of the path arriving at node ζ and starting from either the root node ζ_{root} or toor node ζ_{toor}, while its weight is determined by the accumulated Hamming weight of the corresponding path.

Consider for example the path $\zeta_{\mathrm{root}} \rightarrow \zeta$ in the forward code tree corresponding to the codeword segment $\boldsymbol{v}_{[0,\ell_{\mathsf{F}}(\zeta)-1]}$. Its accumulated Hamming weight is given by

$$\omega_{\mathsf{F}}(\zeta) = w_{\mathrm{H}}\left(\boldsymbol{v}_{[0,\ell_{\mathsf{F}}(\zeta))}\right) = \sum_{i=0}^{\ell_{\mathsf{F}}(\zeta)-1} w_{\mathrm{H}}\left(v_i\right) \tag{2.40}$$

where $\ell_{\mathsf{F}}(\zeta_{\mathrm{root}}) = 0$, $\omega_{\mathsf{F}}(\zeta_{\mathrm{root}}) = 0$, and $\boldsymbol{\sigma}(\zeta_{\mathrm{root}}) = \mathbf{0}$.

Similarly, the path $\zeta_{\mathrm{toor}} \rightarrow \zeta$ in the backward code tree corresponds to the codeword segment $\boldsymbol{v}_{[N-\ell_{\mathsf{B}}(\zeta),N-1]}$ and yields the accumulated Hamming weight

$$\omega_{\mathsf{B}}(\zeta) = w_{\mathrm{H}}\left(\boldsymbol{v}_{[N-\ell_{\mathsf{B}}(\zeta),N)}\right) = \sum_{i=N-\ell_{\mathsf{B}}(\zeta)}^{N-1} w_{\mathrm{H}}\left(v_i\right) \tag{2.41}$$

where $\ell_{\mathsf{B}}(\zeta_{\mathrm{toor}}) = 0$, $\omega_{\mathsf{B}}(\zeta_{\mathrm{toor}}) = 0$, and $\boldsymbol{\sigma}(\zeta_{\mathrm{toor}}) = \mathbf{0}$.

Clearly, for every codeword $\boldsymbol{v}$ with Hamming weight w, there exists a path $\zeta_{\mathrm{root}} \rightarrow \zeta_{\mathrm{toor}}$ in the code tree with an intermediate node ζ such that

$$\omega_{\mathsf{F}}(\zeta) = \left\lfloor \frac{w}{2} \right\rfloor \qquad \omega_{\mathsf{B}}(\zeta) = \left\lceil \frac{w}{2} \right\rceil \qquad \ell_{\mathsf{F}}(\zeta) + \ell_{\mathsf{B}}(\zeta) = N \tag{2.42}$$

Hence, searching for all such paths (codewords) can be split up into two separate and independent steps; a *forward search* for all path segments $\zeta_{\mathrm{root}} \rightarrow \zeta$ with Hamming weight w_{F} as well as a *backward search* for all path segments $\zeta_{\mathrm{toor}} \rightarrow \zeta$ with Hamming weight w_{B}, where the *forward* and *backward weights* w_{F} and w_{B} can be chosen freely[3] as long as

$$w_{\mathsf{F}} + w_{\mathsf{B}} = w \tag{2.43}$$

Since every branch in a bit-level trellis is labeled by exactly one code symbol, the length of any such path segment has to satisfy

$$\omega_{\mathsf{F}}(\zeta) \leq \ell_{\mathsf{F}}(\zeta) \leq N - \omega_{\mathsf{B}}(\zeta) \tag{2.44}$$

$$\omega_{\mathsf{B}}(\zeta) \leq \ell_{\mathsf{B}}(\zeta) \leq N - \omega_{\mathsf{F}}(\zeta) \tag{2.45}$$

In other words, the maximum depth of the forward and backward code tree is limited by $N - w_{\mathsf{B}}$ and $N - w_{\mathsf{F}}$, respectively.

[3]In order to efficiently exploit the bidirectional idea behind the BEAST, the size of the forward and backward code trees should be balanced, that is, the forward and backward weights w_{F} and w_{B} should be approximately equal.

Algorithm BS (BEAST for finding the spectral component A_w)

1. *Forward search:* Starting at the root node ζ_{root}, grow a forward code tree to obtain the set of nodes[4]

$$\mathcal{F} = \left\{\zeta \mid \omega_{\mathsf{F}}(\zeta) = w_{\mathsf{F}},\, \omega_{\mathsf{F}}(\zeta^{\mathrm{P}}) < w_{\mathsf{F}},\, \ell_{\mathsf{F}}(\zeta) \leq N - w_{\mathsf{B}}\right\}$$

   where w_{F} and w_{B} are chosen according to (2.43). The set $\mathcal{F}$ contains the leaves of the partially explored forward code tree, whose accumulated Hamming weights are equal to the forward weight w_{F}.

2. *Backward search:* Starting at the toor node ζ_{toor}, grow a backward code tree to obtain the set of nodes[4]

$$\mathcal{B} = \left\{\zeta \mid \omega_{\mathsf{B}}(\zeta) = w_{\mathsf{B}},\, \omega_{\mathsf{B}}(\zeta^{\mathrm{C}}) > w_{\mathsf{B}},\, \ell_{\mathsf{B}}(\zeta) \leq N - w_{\mathsf{F}}\right\}$$

   Similar to step **1**, the set $\mathcal{B}$ contains the last interior nodes of the partially explored backward code tree, before their accumulated Hamming weights exceed the backward weight w_{B}.

3. *Matching:* Find all pairs of nodes $(\zeta, \zeta') \in \mathcal{F} \times \mathcal{B}$ such that

$$\boldsymbol{\sigma}(\zeta) = \boldsymbol{\sigma}(\zeta') \qquad \ell_{\mathsf{F}}(\zeta) + \ell_{\mathsf{B}}(\zeta') = N \tag{2.46}$$

   Each such match describes a unique codeword with Hamming weight $w = \omega_{\mathsf{F}}(\zeta) + \omega_{\mathsf{B}}(\zeta') = w_{\mathsf{F}} + w_{\mathsf{B}}$. Thus, the number of codewords with Hamming weight w, that is, the spectral component A_w, follows as

$$A_w = \sum_{(\zeta,\zeta') \in \mathcal{F} \times \mathcal{B}} \chi(\zeta, \zeta')$$

   where χ is the match-indicator function, defined as

$$\chi(\zeta, \zeta') = \begin{cases} 1 & \text{if (2.46) holds} \\ 0 & \text{otherwise} \end{cases}$$

[4]The conditions $\omega_{\mathsf{F}}(\zeta^{\mathrm{P}}) < w_{\mathsf{F}}$ and $\omega_{\mathsf{B}}(\zeta^{\mathrm{C}}) > w_{\mathsf{B}}$ in the forward and backward set $\mathcal{F}$ and $\mathcal{B}$, respectively, are necessary to avoid multiple matches corresponding to the same codeword. While one of these two conditions would be sufficient to avoid such multiple matches, the second condition helps to additional decrease the number of stored nodes.

■ **Example 2.5:**
Consider the $(6,3,3)$ shortened Hamming block code from Example 2.3 whose forward and backward code trees are given in Figures 2.8 and 2.9, respectively.

Assume that we want to find the number of codewords with Hamming weight $w = d_{\min} = 3$, that is, A_3. According to (2.43), a possible choice for the forward and backward weights is given by $w_{\mathsf{F}} = 2$ and $w_{\mathsf{B}} = 1$.

The corresponding, partially explored, forward and backward code trees are illustrated in Figure 2.12. The nodes stored in the forward set $\mathcal{F}$ and in the backward set $\mathcal{B}$ are marked by squares, while dashed lines in the backward code tree indicate branches to children nodes exceeding the backward weight w_{B}. In total, the forward and backward sets contain $|\mathcal{F}| = 6$ and $|\mathcal{B}| = 4$ nodes, respectively, which are given below:

$$\mathcal{F} = \Big\{ \Big(\sigma = (1\,0\,0), \ell = 2\Big), \Big(\sigma = (0\,1\,0), \ell = 3\Big), \Big(\sigma = (0\,1\,1), \ell = 3\Big), \\ \Big(\sigma = (0\,1\,0), \ell = 4\Big), \Big(\sigma = (0\,0\,1), \ell = 4\Big), \Big(\sigma = (0\,1\,0), \ell = 5\Big) \Big\}$$

$$\mathcal{B} = \Big\{ \Big(\sigma = (0\,1\,0), \ell = 1\Big), \Big(\sigma = (0\,1\,0), \ell = 2\Big), \\ \Big(\sigma = (0\,0\,1), \ell = 2\Big), \Big(\sigma = (1\,0\,0), \ell = 4\Big) \Big\}$$

Comparing these nodes, we find 3 node pairs satisfying the condition (2.46), which are highlighted in Figure 2.12: state $\sigma = (1\,0\,0)$ with $\ell_{\mathsf{F}} = 2$ and $\ell_{\mathsf{B}} = 4$ as well as states $\sigma = (0\,1\,0)$ and $\sigma = (0\,0\,1)$, both with $\ell_{\mathsf{F}} = 4$ and $\ell_{\mathsf{B}} = 2$. Hence, the $(6,3,3)$ shortened Hamming code contains $A_3 = 3$ codewords of Hamming weight $w = 3$.

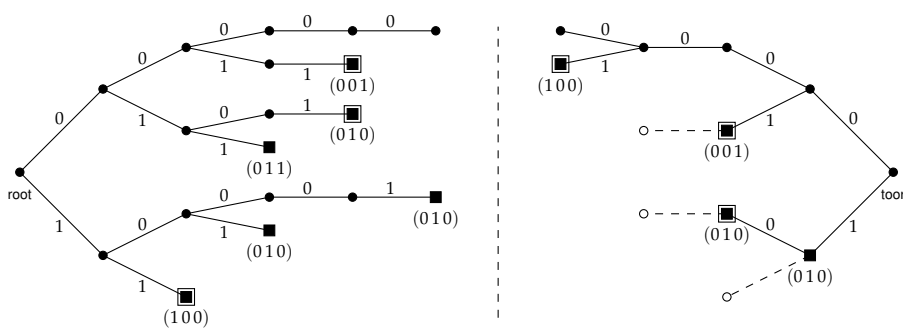


Figure 2.12: Partially explored forward and backward code trees used by the BEAST to determine the number of codewords with Hamming weight $w = 3$ for the $(6,3,3)$ shortened Hamming code.

2.4.2 FINDING THE VITERBI SPECTRUM

Consider a rate $R = b/c$ convolutional code with free distance d_{free}. Recall, that its $(i+1)$th Viterbi spectral component is denoted by $n_{d_{\text{free}}+i}$ and is defined as the number of paths with Hamming weight $d_{\text{free}} + i$, which diverge from the all-zero path at the root of the code trellis and terminate in the all-zero encoder state, but do not merge with the all-zero path until their termini.

Similar to the case when finding the spectral component for a given block code, there exists an intermediate node $\boldsymbol{\sigma}(\zeta) \neq \mathbf{0}$ for every path with Hamming weight w in the code tree which satisfies exactly one of the following c conditions

$$\omega_{\mathsf{F}}(\zeta) = \left\lfloor \frac{w}{2} \right\rfloor + j \qquad \omega_{\mathsf{B}}(\zeta) = \left\lceil \frac{w}{2} \right\rceil - j \qquad j = 0, 1, \ldots, c-1 \tag{2.47}$$

where the additional term $j = 0, 1, \ldots, c-1$ originates from the fact that every branch in the corresponding trellis or code tree is labeled by a code c-tuple.

Since the length of the detour from the all-zero path varies among different code sequences, several toor nodes have to be taken into account; one for each possible length of the detour of its code sequences. However, due to the regular and time-invariant structure of the trellis for convolutional codes, it is sufficient to only consider a single toor node and instead omit the restriction to a specific depth (length) in (2.42).

Algorithm BVS (BEAST for finding the Viterbi spectral component n_w)

1. *Forward search:* Starting at the root node ζ_{root}, extend the forward code tree to obtain c sets indexed by $j = 0, 1, \ldots, c-1$ containing only the states $\boldsymbol{\sigma}(\zeta)$ of all nodes ζ satisfying

$$\mathcal{F}_{+j} = \left\{ \zeta \mid \omega_{\mathsf{F}}(\zeta) = w_{\mathsf{F}} + j,\ \omega_{\mathsf{F}}(\zeta^{\mathrm{P}}) < w_{\mathsf{F}},\ \boldsymbol{\sigma}(\zeta) \neq \mathbf{0} \right\}$$

   where w_{F}, and hence w_{B}, are chosen according to (2.43).

2. *Backward search:* Starting at the toor node ζ_{toor}, extend the backward code tree to obtain c sets indexed by $j = 0, 1, \ldots, c-1$ containing only the states $\boldsymbol{\sigma}(\zeta)$ of all nodes ζ satisfying

$$\mathcal{B}_{-j} = \left\{ \zeta \mid \omega_{\mathsf{B}}(\zeta) = w_{\mathsf{B}} - j,\ \omega_{\mathsf{B}}(\zeta^{\mathrm{C}}) > w_{\mathsf{B}},\ \boldsymbol{\sigma}(\zeta) \neq \mathbf{0} \right\}$$

3. *Matching:* For every pair $\{\mathcal{F}_{+j}, \mathcal{B}_{-j}\}$, $j = 0, 1, \ldots, c-1$, find all pairs of nodes $(\zeta, \zeta') \in \mathcal{F}_{+j} \times \mathcal{B}_{-j}$ with equal states $\boldsymbol{\sigma}(\zeta) = \boldsymbol{\sigma}(\zeta')$. Then the number of convolutional code sequences with Hamming weight w follows as

$$n_w = \sum_{j=0}^{c-1} \sum_{(\zeta,\zeta') \in \mathcal{F}_{+j} \times \mathcal{B}_{-j}} \chi(\zeta, \zeta')$$

2.4.3 MAXIMUM-LIKELIHOOD DECODING

Besides finding the weight and Viterbi spectrum for block and convolutional codes, the BEAST can be used for ML sequence decoding of (N, K) block codes. Assuming that the squared Euclidean distance or the (weighted) Hamming distance is used as a distance measure, ML sequence decoding corresponds to finding the codeword $\boldsymbol{v}$, whose transmitted sequence $\boldsymbol{x}$ is closest to the received sequence $\boldsymbol{r}$ in terms of the chosen metric.

Denote by $d_\mu(\boldsymbol{r}, \boldsymbol{v})$ the chosen metric between the received sequence $\boldsymbol{r}$ and the codeword $\boldsymbol{v}$, where

$$d_\mu(\boldsymbol{r}, \boldsymbol{v}) = \begin{cases} \|\boldsymbol{r} - \boldsymbol{x}\|^2 & \text{squared Euclidean distance, cf. (1.91)} \\ d_{\mathrm{wH}}(\boldsymbol{r}, \boldsymbol{x}) & \text{weighted Hamming distance, cf. (1.98)} \\ d_{\mathrm{H}}(\boldsymbol{r}, \boldsymbol{v}) & \text{Hamming distance, cf. (1.89)} \end{cases} \tag{2.48}$$

If we replace the accumulated Hamming weights (2.40) and (2.41) by the accumulated decoding metrics, we obtain

$$\omega_{\mathsf{F}}(\zeta) = d_\mu\left(\boldsymbol{r}_{[0,\ell_{\mathsf{F}}(\zeta))}, \boldsymbol{v}_{[0,\ell_{\mathsf{F}}(\zeta))}\right) = \sum_{i=0}^{\ell_{\mathsf{F}}(\zeta)-1} d_\mu(r_i, v_i) \tag{2.49}$$

$$\omega_{\mathsf{B}}(\zeta) = d_\mu\left(\boldsymbol{r}_{[N-\ell_{\mathsf{B}}(\zeta),N)}, \boldsymbol{v}_{[N-\ell_{\mathsf{B}}(\zeta),N)}\right) = \sum_{i=N-\ell_{\mathsf{B}}(\zeta)}^{N-1} d_\mu(r_i, v_i) \tag{2.50}$$

Then the BEAST can be used to find a path in the code trellis with smallest metric-sum $w = w_{\mathsf{F}}(\zeta) + w_{\mathsf{B}}(\zeta) = d_\mu(\boldsymbol{r}, \boldsymbol{v})$, that is, the codeword $\boldsymbol{v}$ which minimizes the metric with respect to the received sequence $\boldsymbol{r}$, and hence corresponds to an ML decoding decision.

However, as the metric of such a path is preliminary unknown, it is necessary to introduce a threshold T (and threshold increments δ_i, cf. Subsection 2.4.4), such that the BEAST can be used to determine *all* codewords whose metric is smaller than or equal to this threshold T. Similarly to (2.43), let such a threshold T be split up into a forward and a backward threshold T_{F} and T_{B}, respectively, satisfying

$$T_{\mathsf{F}} + T_{\mathsf{B}} = T \tag{2.51}$$

Additionally replacing the weight equalities in the previous definitions of the forward and backward sets by metric inequalities, the BEAST is guaranteed to find all paths in the code tree whose metrics is smaller than or equal to the current threshold T. However, since the forward and backward code trees are extended separately, it is possible that the sum of the two partial metrics $w_{\mathsf{F}}(\zeta) + w_{\mathsf{B}}(\zeta)$ exceeds the threshold T. Such a path does not necessarily correspond to an ML codeword decision and therefore has to be disregarded.

In particular, only if there exists a matching node pair whose metric-sum is smaller than or equal to the threshold T, the algorithm is terminated; otherwise the threshold T is increased by the next threshold increment δ_i and the corresponding code trees have to be further extended.

Note, unlike for Viterbi sequence decoding, the metric used by the BEAST has to be a *nondecreasing* function of the path lengths. This condition is satisfied by the metrics given in (2.48), but is for example violated by the correlation metric (1.94).

Algorithm BML (BEAST for ML decoding of block codes)

1. *Initialization:* Determine suitable threshold increments δ_i, $i = 1, 2, \ldots, N$ and set the threshold T to its starting value according to Subsection 2.4.4. Moreover initialize the forward and backward code trees with the root and the toor node ζ_{root} and ζ_{toor}, respectively, both with depth $\ell_{\mathsf{F}} = \ell_{\mathsf{B}} = 0$ and metric $\omega_{\mathsf{F}} = \omega_{\mathsf{B}} = 0$.

2. *Forward search:* Extend the forward code tree to find the set of nodes

$$\mathcal{F} = \left\{ \zeta \;\middle|\; \omega_{\mathsf{F}}(\zeta) \geq T_{\mathsf{F}},\, \omega_{\mathsf{F}}(\zeta^{\mathrm{P}}) < T_{\mathsf{F}},\, \ell_{\mathsf{F}}(\zeta) \leq N - \min_{\zeta' \in \mathcal{B}} \{\ell_{\mathsf{B}}(\zeta')\} \right\}$$

   In other words, every node whose depth and metric are smaller than $N - \min_{\zeta' \in \mathcal{B}} \{\ell_{\mathsf{B}}(\zeta')\}$ and T_{F}, respectively, shall be extended, where T_{F}, and hence T_{B} are chosen according to (2.51). Once the metric of a child node reaches or exceeds T_{F}, the corresponding state is stored in the forward set $\mathcal{F}$ together with its metric, depth, and parent node. The knowledge of the parent node is necessary in order to be able to perform »backtracking«, that is, to determine the path segment leading to this node. However, due to this requirement, it is necessary to additionally store *all* intermediate nodes.

3. *Backward search:* Extend the backward code tree to find the set of nodes

$$\mathcal{B} = \left\{ \zeta \;\middle|\; \omega_{\mathsf{B}}(\zeta) \leq T_{\mathsf{B}},\, \ell_{\mathsf{B}}(\zeta) \leq N - \min_{\zeta' \in \mathcal{B}} \{\ell_{\mathsf{F}}(\zeta')\} \right\}$$

   For ML decoding we need to store all interior nodes with metric smaller than or equal to T_{B} and depth below $N - \min_{\zeta' \in \mathcal{B}} \{\ell_{\mathsf{F}}(\zeta')\}$. In particular, for every node, its metric, depth, and parent node have to be stored. Note that the previously used condition on child nodes exceeding T_{B} has to be dropped since the smallest metric for a complete path is not known beforehand, and thus all possible path segments smaller than or equal to the threshold T have to be taken into account.

4. *Matching:* Find all pairs of nodes $(\zeta, \zeta') \in \mathcal{F} \times \mathcal{B}$ such that

$$\boldsymbol{\sigma}(\zeta) = \boldsymbol{\sigma}(\zeta') \qquad \ell_{\mathsf{F}}(\zeta) + \ell_{\mathsf{B}}(\zeta') = N \tag{2.52}$$

Each such match describes a unique codeword with metric $w = \omega_{\mathsf{F}}(\zeta) + \omega_{\mathsf{B}}(\zeta')$ with respect to the received sequence $\boldsymbol{r}$. If the smallest metric-sum among all such codewords satisfies $w \leq T$, then the ML decoding decision is found. To determine (one of) the corresponding ML codeword(s) $\hat{\boldsymbol{v}}$, we perform backtracking from (one of) its matching node(s) ζ to both the root and the toor nodes. Otherwise, if the metric-sum of all matching nodes exceeds the current threshold T, the next threshold increment δ_i is added to T, before the algorithm continues with step **BML-2**.

Note that the calculation of the forward and backward sets in step **BML-2** and **BML-3** are practically independent and can be carried out in parallel by neglecting the condition on the maximum search depth, which yields only a negligible increase in the size of the corresponding forward and backward sets. Alternatively, the efficiency of the BEAST can be enhanced by only increasing the smaller one of the two code trees. That is, instead of increasing the threshold T by the next increment δ in step **BML-4**, it is sufficient to only increase either T_{F} or T_{B} by δ_i. Hence, only one code tree has to be recalculated (extended) while the other one can be reused.

2.4.4 DETERMINE THE METRIC THRESHOLDS

The choice of the threshold increments δ_i, $i = 0, 1, \ldots, N-1$, depends largely on the chosen metric, but is critical to the efficiency of the BEAST. If the threshold increments δ_i are chosen too large, the forward and backward code trees will be extended unnecessarily, while choosing those values too small results in several (unnecessary) iterations before finding the ML decoding decision.

When using the Hamming distance measure, the path metric can only grow in unit steps, and hence $\delta_i = 1$, $i = 0, 1, \ldots, N-1$. Moreover, good block codes can correct error patterns up to roughly half the minimum distance, and thus a good choice for the initial threshold is

$$T = \sum_{i=0}^{t-1} \delta_i, \quad \text{with } t = \lceil d_{\min}/2 \rceil \tag{2.53}$$

For the weighted Hamming distance measure (1.95), the path metric can increase in steps of $|r_i|$; hence, the threshold increments follow as the sorted absolute values $|r_i|$ in ascending order, while the initial threshold is similarly determined by (2.53).

Finally, consider the squared Euclidean distance measure as given by (1.92), where for BPSK signaling $\|\boldsymbol{x}\|^2 = NE_s$. A suitable initial threshold is given by the minimal distance between the received sequence $\boldsymbol{r}$ and any possibly transmitted sequence $\boldsymbol{x}$, that is,

$$T = \min_{\boldsymbol{x}} \left\{ \|\boldsymbol{r} - \boldsymbol{x}\|^2 \right\} = \|\boldsymbol{r}\|^2 + NE_s - 2\sqrt{E_s} \sum_{i=0}^{N-1} |r_i| \tag{2.54}$$

The corresponding threshold increments δ_i are determined as the sorted absolute values $4\sqrt{E_s}\,|r_i|$ in ascending order, such that the threshold

$$T + \sum_{j=0}^{i} \delta_j \tag{2.55}$$

corresponds to the minimal squared Euclidean distance between the received sequence $\boldsymbol{r}$ and any possibly transmitted sequence $\boldsymbol{x}$, where the i most unreliable received symbols have been flipped.

■ **Example 2.6:**
Consider the $(6, 3, 3)$ shortened Hamming code with generator matrix (2.25), used to communicate over an AWGN channel. Suppose we obtain the received sequence

$$\boldsymbol{r} = \begin{pmatrix} 0.82 & -0.42 & 0.17 & 1.25 & 0.83 & 0.37 \end{pmatrix}$$

Using the weighted Hamming distance measure, the threshold increments δ_i are given by the sorted absolute received symbols $|r_i|$

$$\{\delta_i\}_{i=1}^{6} = \{0.17, 0.37, 0.42, 0.82, 0.83, 1.25\}$$

and hence the initial threshold follows according to (2.53) as

$$T = \sum_{i=1}^{\lceil 3/2 \rceil} \delta_i = \delta_1 + \delta_2 = 0.17 + 0.37 = 0.54$$

A suitable choice for the forward and backward thresholds T_{F} and T_{B}, such that (2.51) is satisfied, is for example given by

$$T_{\mathsf{F}} = T_{\mathsf{B}} = T/2 = 0.27 \tag{2.56}$$

Growing the corresponding forward and backward code trees according to **BML-2** and **BML-3**, yields the partially explored code trees as illustrated in Figure 2.13. The corresponding forward set $\mathcal{F}$ and backward set $\mathcal{B}$ contain 4 and 7 nodes, respectively, which are given below:

$$\begin{aligned}\mathcal{F} = \Big\{ &\Big(\boldsymbol{\sigma} = (1\,0\,0), \ell_{\mathsf{F}} = 1, \omega_{\mathsf{F}} = 0.82\Big), \Big(\boldsymbol{\sigma} = (0\,0\,0), \ell_{\mathsf{F}} = 2, \omega_{\mathsf{F}} = 0.42\Big), \\ &\Big(\boldsymbol{\sigma} = (0\,1\,0), \ell_{\mathsf{F}} = 4, \omega_{\mathsf{F}} = 1.25\Big), \Big(\boldsymbol{\sigma} = (0\,1\,0), \ell_{\mathsf{F}} = 5, \omega_{\mathsf{F}} = 1.0\Big) \Big\}\end{aligned}$$

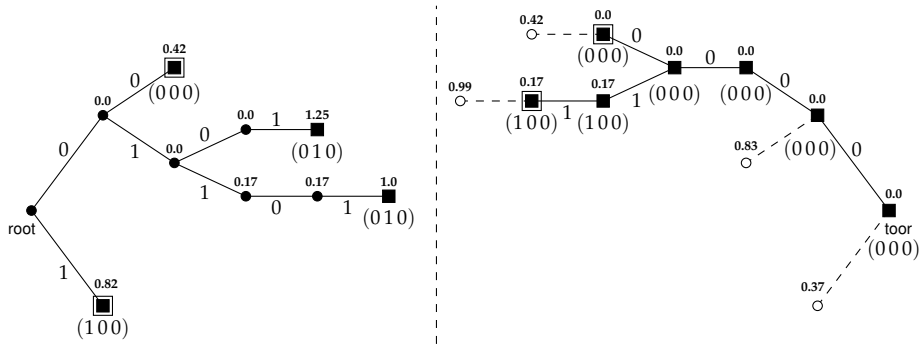


Figure 2.13: Partially explored forward and backward code trees used by the BEAST when decoding the received sequence $\boldsymbol{r}$ from Example 2.6.

$$\begin{aligned}\mathcal{B} = \Big\{ &\Big(\boldsymbol{\sigma} = (000), \ell_{\mathsf{B}} = 0, \omega_{\mathsf{B}} = 0.0\Big), \Big(\boldsymbol{\sigma} = (000), \ell_{\mathsf{B}} = 1, \omega_{\mathsf{B}} = 0.0\Big), \\ &\Big(\boldsymbol{\sigma} = (000), \ell_{\mathsf{B}} = 2, \omega_{\mathsf{B}} = 0.0\Big), \Big(\boldsymbol{\sigma} = (000), \ell_{\mathsf{B}} = 3, \omega_{\mathsf{B}} = 0.0\Big), \\ &\Big(\boldsymbol{\sigma} = (100), \ell_{\mathsf{B}} = 4, \omega_{\mathsf{B}} = 0.17\Big), \Big(\boldsymbol{\sigma} = (000), \ell_{\mathsf{B}} = 4, \omega_{\mathsf{B}} = 0.0\Big), \\ &\Big(\boldsymbol{\sigma} = (100), \ell_{\mathsf{B}} = 5, \omega_{\mathsf{B}} = 0.17\Big)\Big\}\end{aligned}$$

Matching those two sets yields two pairs of nodes satisfying (2.52):

(i) Node $\boldsymbol{\sigma} = (000)$ with $\ell_{\mathsf{F}} = 2$, $\ell_{\mathsf{B}} = 4$, and metric-sum $\omega = \omega_{\mathsf{F}} + \omega_{\mathsf{B}} = 0.42 + 0.0 = 0.42$, corresponding to the all-zero codeword 000000.

(ii) Node $\boldsymbol{\sigma} = (100)$ with $\ell_{\mathsf{F}} = 1$, $\ell_{\mathsf{B}} = 5$, and metric-sum $\omega = \omega_{\mathsf{F}} + \omega_{\mathsf{B}} = 0.82 + 0.17 = 0.99$, corresponding to the codeword 111000.

Clearly, the first pair of nodes has the smallest metric-sum $\omega = 0.42$ and, moreover, satisfies $\omega \leq T = 0.54$. Hence, this path determines the ML decoding decision and we obtain the ML codeword $\hat{\boldsymbol{v}}_{\mathrm{ML}} = 000000$.

2.5 LOW-DENSITY PARITY-CHECK CODES AND TANNER GRAPHS

Low-density parity-check (LDPC) codes were initially invented by Gallager in the early 1960s [Gal62] [Gal63] and constitute an important class of block codes defined on bipartite graphs. We shall start by introducing some basic properties of bipartite graphs, before discussing LDPC block codes in more detail towards the end of this section.

BASIC GRAPH PROPERTIES

An undirected graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$ is defined by a set of vertices $\mathcal{V} = \{\zeta_i\}$ and a set of edges $\mathcal{E} = \{e_i\}$, where each edge connects exactly two vertices. The *degree of a vertex* denotes the number of edges that are connected to it.

Consider the set of vertices $\mathcal{V}$ of a graph partitioned into t disjoint subsets $\mathcal{V}_k$, $k = 0, 1, \ldots, t-1$. Such a graph is said to be *t-partite* if no edge connects two vertices from the same set $\mathcal{V}_k$, $k = 0, 1, \ldots, t-1$.

Similarly, consider a set of vertices $\mathcal{V}_k$ of such a t-partite graph, which is itself partitioned into s disjoint subsets $\mathcal{V}_k^{(\ell)}$, $\ell = 0, 1, \ldots, s-1$. Then, the corresponding set of vertices $\mathcal{V}_k$ is said to be *s-partite*, if and only if there is no possibility to connect two vertices from the same subset $\mathcal{V}_k^{(\ell)}$, $\ell = 0, 1, \ldots, s-1$, by using (not more than) two distinct edges.

Suppose that $\mathcal{G}$ is a bipartite graph with two disjoint subsets of vertices, $\mathcal{V}_0$ and $\mathcal{V}_1$, with $|\mathcal{V}_0| > |\mathcal{V}_1|$. Commonly, the vertices in $\mathcal{V}_0$ are called *symbol nodes* while the vertices in $\mathcal{V}_1$ are referred to as *constraint* or *parity-check nodes*. Such a bipartite graph is also known as a *Tanner graph* [Tan81] and determines a linear block code $\mathcal{B}$ with block length $|\mathcal{V}_0|$ and dimension at least $|\mathcal{V}_0| - |\mathcal{V}_1|$.

The parity-check matrix H of the corresponding block code $\mathcal{B}$ consists of $|\mathcal{V}_1|$ rows and $|\mathcal{V}_0|$ columns, where every row and column corresponds to a constraint node in $\mathcal{V}_1$ and a symbol node in $\mathcal{V}_0$, respectively. If an edge directly connects the symbol node $v_i \in \mathcal{V}_0$ with the constraint node $c_j \in \mathcal{V}_1$, then the entry in row j and column i of the parity-check matrix H is set to one; and to zero otherwise.

■ **Example 2.7:**
Consider the $(6,3,3)$ shortened Hamming code with parity-check matrix

$$H = \begin{array}{c} \\ c_0 \\ c_1 \\ c_2 \end{array} \begin{array}{c} \begin{array}{cccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{2.57}$$

Its corresponding Tanner graph contains 6 symbol nodes v_j, $j = 0, 1, \ldots, 5$, as well as 3 constraint nodes c_i, $i = 0, 1, 2$, and is given in Figure 2.14. The symbol nodes in $\mathcal{V}_0$ are illustrated by black circles having either degree 1 or 2, while the constraint nodes in $\mathcal{V}_1$ are given by white circles having degree 3.

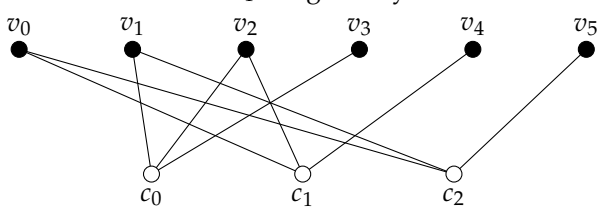


Figure 2.14: Tanner graph of the shortened Hamming $(6,3)$ code defined by the parity-check matrix (2.57).

From a graph theoretical point of view, every undirected graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$ can be represented either by its $|\mathcal{V}| \times |\mathcal{E}|$ *incidence matrix* $T = (t_{ij})$ or by its $|\mathcal{V}| \times |\mathcal{V}|$ *adjacency matrix* $A = (a_{ij})$.

If the edge e_j, $j = 0, 1, \ldots, |\mathcal{E}| - 1$, is incident (connected) to the vertex v_i, $i = 0, 1, \ldots, |\mathcal{V}| - 1$, then the corresponding entry t_{ij} of the incidence matrix T is set to one; and to zero otherwise. Similarly, if the vertex v_i is directly connected to vertex v_j, $i, j = 0, 1, \ldots, |\mathcal{V}| - 1$, then the entry a_{ij} of the adjacency matrix A is equal to one; and equal to zero otherwise.

For a bipartite graph $\mathcal{G}$ with two distinct sets of vertices $\mathcal{V}_0$ and $\mathcal{V}_1$, the adjacency matrix A has the following block structure

$$A = \begin{pmatrix} 0 & B \\ B^{\mathrm{T}} & 0 \end{pmatrix} \tag{2.58}$$

where B denotes the $|\mathcal{V}_1| \times |\mathcal{V}_0|$ *biadjacency matrix* of the graph $\mathcal{G}$. In particular note that the biadjacency matrix B of a graph $\mathcal{G}$ is equal to the parity-check matrix H of the corresponding linear block code $\mathcal{B}$.

■ **Example 2.7 (Cont'd):**
Consider the Tanner graph determined by the parity-check matrix H in (2.57) with $|\mathcal{V}| = 9$ vertices and $|\mathcal{E}| = 9$ edges. Its 9×9 incidence matrix is given by

$$T = \begin{array}{c} c_0 \\ c_1 \\ c_2 \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.59}$$

while its 9×9 adjacency matrix follows as

$$A = \begin{array}{c} \\ c_0 \\ c_1 \\ c_2 \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccccccc} c_0 & c_1 & c_2 & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \left(\begin{array}{ccc|cccccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array} \tag{2.60}$$

In particular note that its biadjacency matrix B coincides with the parity-check matrix H given in (2.57).

Finally, denote by a *path* of length ℓ in a graph $\mathcal{G}$ an alternating sequence of $\ell+1$ vertices ζ_i, $i = 0, 1, \ldots, \ell$, and ℓ edges e_i, $i = 0, 1, \ldots, \ell-1$, with $e_i \neq e_{i+1}$, such that the vertices ζ_i and ζ_{i+1} are connected by the edge e_i. If the first and the final vertex of a path coincide, that is, if $\zeta_0 = \zeta_\ell$, we refer to such a path as a *cycle*. It is called a *simple cycle* if all its vertices and edges are distinct, except for the first and final vertex, which coincide. The length of the shortest simple cycle in a graph is denoted its *girth* g. Note that the girth of a bipartite graph is by definition always even.

LOW-DENSITY PARITY-CHECK BLOCK CODES

An LDPC block code can be either defined by a Tanner graph $\mathcal{G}$ or a *sparse* parity-check matrix H, that is, a parity-check matrix »containing mostly 0s and relatively few 1s« [Gal62] [Gal63][5].

Commonly, LDPC block codes can be characterized by having either *random/pseudo-random* or *nonrandom* structures, where LDPC block codes with nonrandom structures are subdivided into *regular* or *irregular*, while random / pseudo-random LDPC codes are in general irregular.

If the parity-check matrix H contains exactly J ones in each column and exactly L ones in each row, the corresponding (nonrandom) LDPC block code is referred to as (J, L)-*regular*; and *irregular* otherwise. In particular, we note that in the Tanner graph of a (J, L)-regular LDPC block code all symbol and constraint nodes have degree J and L, respectively.

When constructing a (J, L)-regular LDPC block with block length N and *design rate* $R_\mathrm{d} = 1 - J/L$, we obtain a parity-check matrix of size $(N-K) \times N$. Since such a matrix can in general have linearly dependent rows, it does not necessarily have full rank. Removing those rows yields the final parity-check matrix of an LDPC block code with slightly larger rate $R \geq R_\mathrm{d}$.

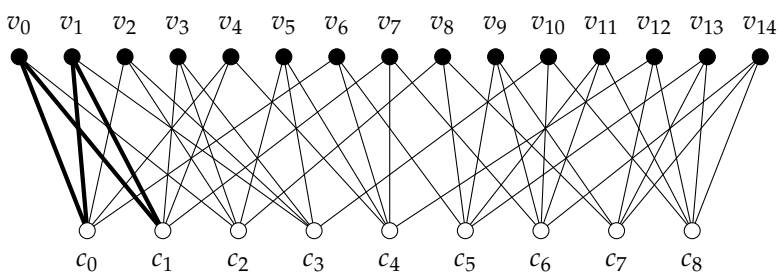


Figure 2.15: Tanner graph of the $(3, 5)$-regular LDPC block code defined by parity-check matrix (2.61).

[5]Hereinafter, short block codes will often be used within examples. For simplicity, we will refer to these codes as LDPC block codes, even though the requirement »containing mostly 0s and relatively few 1s« is not satisfied.

■ **Example 2.8:**
Consider the 9×15 parity-check matrix H of the $(J = 3, L = 5)$-regular LDPC block code with (design) rate $R_{\mathrm{d}} = R = 6/15 = 1 - 3/5$.

$$H = \begin{array}{c} \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & c_7 & c_8 & c_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array} \tag{2.61}$$

The corresponding Tanner graph with 15 symbol nodes and 9 constraint nodes is illustrated in Figure 2.15. Every constraint node has degree $L = 5$ while every symbol node has degree $J = 3$. Consider the simple cycle of length 4 given by $v_0 \to c_0 \to v_1 \to c_1 \to v_0$ which is marked in Figure 2.15. This simple cycle has the shortest possible length within this graph, and hence the girth of this $(3, 5)$-regular LDPC block code follows as $g = 4$.

Finally, note that in case of a (J, L)-regular LDPC block code with design rate $R_{\mathrm{d}} = K/N$, the corresponding Tanner graph consists of N symbol nodes, $N - K$ constraint nodes, and JN edges. Hence, its incidence and adjacency matrices are of size $(2N - K) \times (JN)$ and $(2N - K) \times (2N - K)$, respectively. Moreover, interpreting the incidence matrix of such a Tanner graph as another parity-check matrix, we obtain a $(JN,\ (J-2)N + K)$ linear block code with larger block length JN.

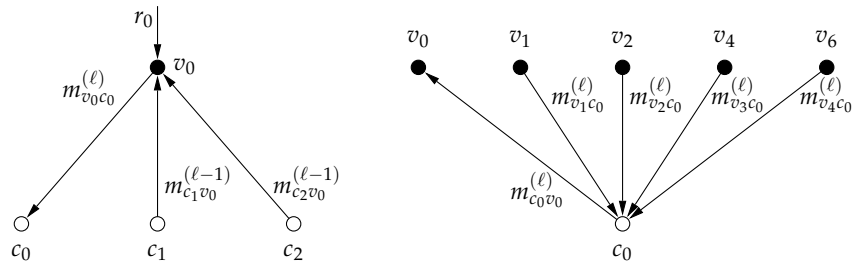
2.6 THE BELIEF PROPAGATION ALGORITHM

The belief propagation (BP) algorithm [Gal63] [Pea88] is an efficient iterative decoding algorithm for LDPC block codes and belongs to the more general class of iterative *message passing algorithms*. It is based on computing the marginal APPs for the code symbols v_i, that is,

$$P(v_i = 0 \mid \boldsymbol{r}) \quad i = 0, 1, \ldots, N-1 \tag{2.62}$$

where N denotes the block length of the LDPC block code and $\boldsymbol{r}$ is a sequence of N observed (received) values.

Consider a bipartite graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$ with two disjoint sets of vertices: the set of symbol nodes $\mathcal{V}_0$ and the set of constraint nodes $\mathcal{V}_1$. During each



(a) Symbol node. **(b)** Constraint node.

Figure 2.16: Computational functions f_v and f_c evaluated at a symbol node v_0 and constraint node c_0, respectively.

iteration ℓ, messages are passed along the edges of the underlying graph from the symbol nodes in $\mathcal{V}_0$ to the constraint nodes in $\mathcal{V}_1$ and vice versa.

Denote the set of vertices being adjacent to a vertex x by $\mathcal{A}(x)$. Then the message from the symbol node $v_i \in \mathcal{V}_0$ to the constraint node $c_j \in \mathcal{V}_1$ and vice versa during iteration ℓ can be expressed as

$$\begin{array}{lll} m^{(\ell)}_{v_ic_j} & \text{with } c_j \in \mathcal{A}(v_i) & (v_i \to c_j) \\ m^{(\ell)}_{c_jv_i} & \text{with } v_i \in \mathcal{A}(c_j) & (c_j \to v_i) \end{array}$$

To be more precise, the message $m^{(\ell)}_{v_ic_j}$ from symbol node v_i to constraint node c_j during iteration ℓ is based on the observed value r_i of the symbol node v_i as well as on the messages received from neighboring constraint nodes during the previous iteration $\ell - 1$; except the message received from constraint node c_j. Hence, we obtain,

$$m^{(\ell)}_{v_ic_j} = f_v\left(\overline{m}^{(\ell-1)}_{c_jv_i}, r_i\right) \quad \text{with } \overline{m}^{(\ell-1)}_{c_jv_i} = \left\{ m^{(\ell-1)}_{cv_i} \;\middle|\; c \in \mathcal{A}(v_i)\backslash\{c_j\} \right\} \tag{2.63}$$

where $f_v(\ )$ is an arbitrary function evaluated at every symbol node. In Figure 2.16(a) this function is illustrated for the symbol node v_0 as specified by the $(3,5)$-regular LDPC block code in Example 2.8.

Similarly, the message $m^{(\ell)}_{c_jv_i}$ from constraint node c_j to symbol node v_i during iteration ℓ is based on the messages received from neighboring symbol nodes during the same iteration; excluding the message received from symbol node v_i, and hence

$$m^{(\ell)}_{c_jv_i} = f_c\left(\overline{m}^{(\ell)}_{v_ic_j}\right) \quad \text{with } \overline{m}^{(\ell)}_{v_ic_j} = \left\{ m^{(\ell)}_{vc_j} \;\middle|\; v \in \mathcal{A}(c_j)\backslash\{v_i\} \right\} \tag{2.64}$$

where $f_c(\ )$ denotes an arbitrary function evaluated at every constraint node. The corresponding function for the constraint node c_0, as specified by the $(3,5)$-regular LDPC block code in Example 2.8, is illustrated in Figure 2.16(b).

In a BP algorithm, these messages correspond to *probabilities* or *beliefs*. Under the assumption that the received messages are mutually independent, the functions $f_v(\ )$ and $f_c(\ )$ can be derived directly. However, since the length of the shortest simple cycle in a graph is equal to its girth g, this independence assumption holds only during the first $g/2$ iterations. Messages which are passed along the edges during later iterations violate this assumption, and yield, in general, a suboptimal decoding decision.

MESSAGES FROM SYMBOL NODES

First, consider the function $f_v(\ )$ evaluated at every symbol node. Using log-likelihoods, the message $m_{v_i c_j}^{(\ell)}$ from symbol node v_i to constraint node c_j during iteration ℓ can be written as

$$m_{v_i c_j}^{(\ell)} = f_v\left(\overline{m}_{c_j v_i}^{(\ell-1)}, r_i\right) = L\left(v_i \;\middle|\; \overline{m}_{c_j v_i}^{(\ell-1)}, r_i\right) \tag{2.65}$$

In other words, the message $m_{v_i c_j}^{(\ell)}$ corresponds to the likelihood that the code symbol (symbol node) v_i has a certain binary value, given the received symbol r_i as well as the messages received from all constraint nodes adjacent to the symbol node v_i during the previous iteration $\ell - 1$, excluding the message received from constraint node c_j. Note that the log-likelihood ratios as defined in (1.12) satisfy

$$L\left(v_i \;\middle|\; \overline{m}_{c_j v_i}^{(\ell-1)}, r_i\right) = L\left(\overline{m}_{c_j v_i}^{(\ell-1)}, r_i \;\middle|\; v_i\right) \tag{2.66}$$

since the code symbols v_i are equiprobable. Hence, assuming that the messages received from different constraint nodes as well as the observed symbol r_i are mutually independent, (2.66) can be re-applied, and

$$\begin{aligned} m_{v_i c_j}^{(\ell)} = L\left(\overline{m}_{c_j v_i}^{(\ell-1)}, r_i \;\middle|\; v_i\right) &= \sum_{c \in \mathcal{A}(v_i)\setminus\{c_j\}} L\left(m_{c v_i}^{(\ell-1)} \;\middle|\; v_i\right) + L\left(r_i \mid v_i\right) \\ &= \sum_{c \in \mathcal{A}(v_i)\setminus\{c_j\}} L\left(v_i \;\middle|\; m_{c v_i}^{(\ell-1)}\right) + L\left(v_i \mid r_i\right) \end{aligned} \tag{2.67}$$

where $L(v_i \mid r_i)$ is commonly called the *intrinsic information*.

MESSAGES FROM CONSTRAINT NODES

Similarly, the message $m_{c_j v_i}^{(\ell)}$ from constraint node c_j to symbol node v_i corresponds to the likelihood that the binary code symbol v_i has a certain value given the messages received from all symbol nodes adjacent to the constraint

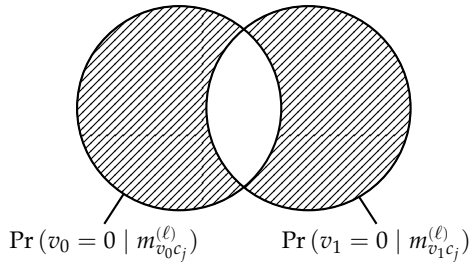


Figure 2.17: Venn diagram of probabilities used by the BP algorithm.

node c_j during the same iteration ℓ except the message received from the symbol node v_i. Using log-likelihoods, the corresponding function $f_c(\ )$ can be expressed as

$$m^{(\ell)}_{c_j v_i} = f_v\left(\overline{m}^{(\ell)}_{c_j v_i}\right) = L\left(v_i \;\middle|\; \overline{m}^{(\ell)}_{v_i c_j}\right) \tag{2.68}$$

Since the modulo-2 sum of all adjacent symbol nodes has to be equal to zero at every constraint node c_j, it follows that

$$\bigoplus_{v \in \mathcal{A}(c_j)} v = 0 \tag{2.69}$$

or, equivalently,

$$v_i = \bigoplus_{v \in \mathcal{A}(c_j) \setminus \{v_i\}} v \qquad \text{with } v_i \in \mathcal{A}(c_j) \tag{2.70}$$

Moreover, let

$$1 - P\left(v_0 \oplus v_1 = 0 \;\middle|\; m^{(\ell)}_{v_0 c_j}, m^{(\ell)}_{v_1 c_j}\right)$$

denote the probability of the two equiprobable events $v_0 = 1$, $v_1 = 0$ and $v_0 = 0$, $v_1 = 1$, which are illustrated by the shaded area in Figure 2.17. This probability can be alternatively expressed as

$$P\left(v_0 = 0 \middle| m^{(\ell)}_{v_0 c_j}\right) + P\left(v_1 = 0 \middle| m^{(\ell)}_{v_1 c_j}\right) - 2\,P\left(v_0 = 0 \middle| m^{(\ell)}_{v_0 c_j}\right) P\left(v_1 = 0 \middle| m^{(\ell)}_{v_1 c_j}\right)$$

Hence it follows that

$$\begin{aligned} &2\,P\left(v_0 \oplus v_1 = 0 \;\middle|\; m^{(\ell)}_{v_0 c_j}, m^{(\ell)}_{v_1 c_j}\right) - 1 \\ &\quad = \left(2\,P\left(v_0 = 0 \;\middle|\; m^{(\ell)}_{v_0 c_j}\right) - 1\right)\left(2\,P\left(v_1 = 0 \;\middle|\; m^{(\ell)}_{v_1 c_j}\right) - 1\right) \end{aligned} \tag{2.71}$$

and, by induction, that

$$2\,P\Big(\bigoplus_{v \in \mathcal{A}(c_j) \setminus \{v_i\}} v = 0 \;\Big|\; \overline{m}^{(\ell)}_{v c_j}\Big) - 1 = \prod_{v \in \mathcal{A}(c_j) \setminus \{v_i\}} \left(2\,P\left(v = 0 \;\middle|\; m^{(\ell)}_{v c_j}\right) - 1\right) \tag{2.72}$$

Next, we recall the definition of the log-likelihood function in (1.12) where

$$P\left(v=0 \;\middle|\; m_{vc_j}^{(\ell)}\right) = \frac{\exp\left(L\left(v \;\middle|\; m_{vc_j}^{(\ell)}\right)\right)}{\exp\left(L\left(v \;\middle|\; m_{vc_j}^{(\ell)}\right)\right)+1} \tag{2.73}$$

and thereby conclude that

$$2\,P\left(v=0\,\middle|\, m_{vc_j}^{(\ell)}\right)-1 = \frac{\exp\left(L\left(v\,\middle|\, m_{vc_j}^{(\ell)}\right)\right)-1}{\exp\left(L\left(v\,\middle|\, m_{vc_j}^{(\ell)}\right)\right)+1} = \tanh\left(\frac{1}{2}\,L\left(v\,\middle|\, m_{vc_j}^{(\ell)}\right)\right) \tag{2.74}$$

Hence the log-likelihood ratio in (2.68) can be written as

$$m_{c_jv_i}^{(\ell)} = 2\tanh^{-1}\left(2\,P\left(v_i=0 \;\middle|\; \overline{m}_{v_ic_j}^{(\ell)}\right)-1\right) \tag{2.75}$$

and, combining (2.72), (2.74), and (2.75), it follows that

$$m_{c_jv_i}^{(\ell)} = 2\,\tanh^{-1}\Big(\prod_{v\in\mathcal{A}(c_j)\setminus\{v_i\}}\tanh\left(\frac{1}{2}\,L\left(v \;\middle|\; m_{vc_j}^{(\ell)}\right)\right)\Big) \tag{2.76}$$

where the symbol node v_i has been replaced by its complementary modulo-2 sum according to (2.70). Moreover, combining (2.67) and (2.76) yields the final updating rules for the BP algorithm

$$m_{v_ic_j}^{(\ell)} = L\left(v_i \mid r_i\right) + \sum_{c\in\mathcal{A}(v_i)\setminus\{c_j\}} m_{cv_i}^{(\ell-1)} \tag{2.77}$$

$$m_{c_jv_i}^{(\ell)} = 2\,\tanh^{-1}\left(\prod_{v\in\mathcal{A}(c_j)\setminus\{v_i\}}\tanh\left(\frac{1}{2}\,m_{v_ic_j}^{(\ell)}\right)\right) \tag{2.78}$$

FINAL DECISIONS AND STOPPING CRITERIA

The messages are passed along the edges of the underlying bipartite graph until a maximum number of iterations $\ell_{\max}$ is reached. Then, the binary decision for the ith symbol node $\hat{v}_i$ after $\ell = \ell_{\max}$ iterations is determined analogously to (2.77) by taking into account *all* received messages, that is,

$$\hat{v}_i^{(\ell)} = \operatorname{sign}\left(L\left(v_i \;\middle|\; \mathbf{BP}^{(\ell)}\right)\right) = \operatorname{sign}\left(L\left(v_i \mid r_i\right) + \sum_{c\in\mathcal{A}(v_i)} m_{cv_i}^{(\ell)}\right) \tag{2.79}$$

where the condition on $\mathbf{BP}^{(\ell)}$ denotes the information obtained by performing the BP algorithm with ℓ iterations. When transmitting over a discrete channel like the BSC or the quantized AWGN channel, the final log-likelihood ratio can be zero. In such a case, the binary decision for the symbol node $\hat{v}_i$ is determined by flipping a fair coin.

As a variant, (2.79) can be evaluated during each iteration and ending the decoding algorithm in advance, once the current tentative symbol node decisions form a valid codeword.

Based on the previous discussions, the BP decoding algorithm can be formulated as follows:

Algorithm BP (Belief Propagation)

1. At iteration $\ell = 0$ initialize the messages from symbol node v_i to constraint node c_j by

$$m_{v_i c_j}^{(0)} = L\left(v_i \mid r_i\right) \tag{2.80}$$

where r_i denotes the ith received symbol, observed at symbol node v_i.

2. Update all messages from constraint node c_j to symbol node v_i according to (2.78).

3. (i) Stop if the maximum number of iteration $\ell_{\max}$ is reached and determine the binary symbol node decisions $\hat{v}_i^{(\ell_{\max})}$ according to (2.79).

(ii) Optionally, determine the current tentative binary symbol node decisions $\hat{v}_i^{(\ell)}$ at iteration ℓ as given by (2.79) and stop if those values form a valid codeword.

4. Increase ℓ by one, update all messages from symbol node v_i to constraint node c_j according to (2.77), and go to step **BP-2**.

Further generalizing the BP algorithm yields the sum-product algorithm for *factor graphs*, used among others to determine the marginals of multivariable PDFs [Wym07].

3

Voltage Graph-Based QC LDPC Block Codes with Large Girth

Due to their low decoding complexity and good BER performance close to the theoretical limit, LDPC block codes are a suitable choice for modern communication standards [IEE05] [Eur08] [Eur09].

In general, LDPC block codes have a minimum distance which is less than that of the best known linear codes, but due to their structure they are suitable for low-complexity iterative decoding algorithm, like for example the BP algorithm (cf. Section 2.6). While the girth of the underlying graph determines the number of independent decoding iterations, the minimum distance of such a code seems to play a minor role for iterative decoding algorithms, since the error-correcting capabilities of such suboptimal procedures are often less than those guaranteed by the minimum distance. More precisely it was shown in [DLZ+09] that their BER performance for high SNRs is predominantly dictated by the structure of the smallest so-called *absorbing sets*. However, since the size of these absorbing sets is upper-bounded by the minimum distance, LDPC block codes with large minimum distance are of particular interest.

In this chapter, we shall focus on the class of quasi-cyclic (QC) (J,L)-regular LDPC block codes, which is a subclass of regular (nonrandom) LDPC codes (cf. Section 2.5) whose codewords can be obtained from another codeword by a cyclic shift of $t \geq 1$ code symbol positions. Such codes are most most suitable for algebraic design and can be efficiently represented in the form of tail-bitten convolutional codes. Commonly, QC LDPC block codes are constructed based on combinatorial approaches using either finite geometries [KLF01] or Steiner Triple Systems (STSs) [JW01a] [JW01b] and have girth $g \geq 6$. Among others, Tanner et al. [TSF01] constructed QC LDPC block codes with rate $R = 2/5$ and girth up to 12 based on subgroups of the multiplicative group of the binary (Galois) field $\mathbb{F}_2$. This method was further generalized in [TSS+04].

Although QC LDPC block codes are not asymptotically optimal, they can outperform nonrandom or pseudorandom LDPC block codes (from asymptotically optimal ensembles) for short or moderate block lengths [Fos04]. This motivates searching for good short QC LDPC block codes.

Section 3.1 focuses on QC LDPC block codes constructed from tailbitten convolutional codes. The corresponding base and voltage graphs, used when searching for QC LDPC block codes with large girth, are introduced in Section 3.2. Bounds on the girth and the minimum distance for QC (J, L)-regular LDPC block codes are discussed in Section 3.3. In Section 3.4, new search algorithms for QC LDPC block codes are presented. Moreover, a new algorithm for computing the minimum distance of QC (J, L)-regular LDPC block codes is described in Section 3.5, while new examples of QC (J, L)-regular LDPC block codes with girth between 10 and 24 are given in Section 3.6.

Finally, Section 3.7 focuses on constructing QC LDPC block codes from different base matrices, including binomials and STSs, presenting new examples of QC (J, L)-regular LDPC block codes constructed in such a way. Section 3.8 concludes this chapter by comparing the BER performance of newly found QC (J, L)-regular LDPC block codes with $J = 3$, rates $R = 1/2$, $R = 2/3$, $R = 3/4$, as well as $R = 5/6$, and block lengths around 576 with the corresponding irregular LDPC block codes defined in the IEEE 802.16 WiMAX standard of same rate and block length.

3.1 QUASI-CYCLIC LDPC BLOCK CODES

Consider a rate $R = b/c$ convolutional code $\mathcal{C}$ determined by the parity-check matrix

$$H(D) = \begin{pmatrix} h_{00}(D) & h_{01}(D) & \dots & h_{0(c-1)}(D) \\ h_{10}(D) & h_{11}(D) & \dots & h_{1(c-1)}(D) \\ \vdots & \vdots & \ddots & \\ h_{(c-b-1)0}(D) & h_{(c-b-1)1}(D) & \dots & h_{(c-b-1)(c-1)}(D) \end{pmatrix} \tag{3.1}$$

with syndrome memory m_s whose parity-check polynomials are monomial entries $h_{ij}(D) = D^{w_{ij}}$ of degree $w_{ij} \leq m_s$, where w_{ij} are nonnegative integers. Clearly, such a parity-check matrix $H(D)$ can be represented by its *degree matrix* $W = (w_{ij})$, $i = 0, 1, \dots, c-b-1$ and $j = 0, 1, \dots, c-1$. Starting with Section 3.7, the restriction to parity-check matrices with only monomial entries will be relaxed to additionally allow zero and binomial entries.

The semi-infinite parity-check matrix for such a convolutional code is given in (1.69). Following Subsection 1.3.2, the $M(c-b) \times Mc$ tailbiting parity-check matrix $H^{(\text{tb})}$ of a corresponding linear QC block code $\mathcal{B}$ is obtained by

tailbiting the semi-infinite parity-check matrix to length M c-tuples, that is,

$$H^{(\mathrm{tb})} = \begin{pmatrix} H_0 & & & & H_{m_\mathrm{s}} & H_{m_\mathrm{s}-1} & \dots & H_1 \\ H_1 & H_0 & & & & H_{m_\mathrm{s}} & \dots & H_2 \\ H_2 & H_1 & \ddots & & & & \ddots & \vdots \\ \vdots & H_2 & \ddots & H_0 & & & & H_{m_\mathrm{s}} \\ H_{m_\mathrm{s}} & \vdots & \ddots & H_1 & H_0 & & & \\ & H_{m_\mathrm{s}} & \ddots & H_2 & H_1 & \ddots & & \\ & & \ddots & \dots & \dots & \ddots & \ddots & \\ & & & H_{m_\mathrm{s}} & H_{m_\mathrm{s}-1} & \dots & H_1 & H_0 \end{pmatrix} \tag{3.2}$$

In particular note that every cyclic shift of a codeword $\boldsymbol{v}$ of $\mathcal{B}$ by c positions modulo Mc yields another codeword.

Due to the restriction to only monomial entries in $H(D)$, $H^{(\mathrm{tb})}$ is (J, L)-*regular*, that is, it contains exactly J and L ones in each column and row, respectively. In particular, using a rate $R = b/c$ parent convolutional code, it follows that $J = c - b$ and $L = c$. Such a parity-check matrix $H^{(\mathrm{tb})}$ determines a (J, L)-regular LDPC block code, or, using the concept of biadjacency matrices, a Tanner graph with vertex and constraint node degree J and L, respectively. Moreover, to satisfy the sparsity of the parity-check matrix $H^{(\mathrm{tb})}$, we assume the tailbiting length M to be much larger than L (and hence J), that is, $M \gg L$.

Consider the Tanner graph with the biadjacency matrix $H^{(\mathrm{tb})}$, corresponding to a QC (J, L)-regular LDPC code, obtained from the parity-check matrix of a tailbiting LDPC block code. Clearly, by letting the tailbiting length M tend to infinity, we return to the convolutional parity-check matrix $H(D)$ (3.1) of the parent convolutional code $\mathcal{C}$. In terms of Tanner graph representations, this procedure corresponds to unwrapping the underlying graph and extending it in the time domain towards infinity. Hereinafter, we will call the girth of such an infinite Tanner graph the *free girth* and denote it by g_{free}.

Finally, note that the first c columns of $H^{(\mathrm{tb})}$ are repeated throughout the whole matrix in a cyclicly shifted manner. Hence, by reordering the columns as $0, c, 2c, \dots, (M-1)c, 1, c+1, 2c+1, \dots, (M-1)c+1$, *etc.* and the rows as $0, (c-b), 2(c-b), \dots, (M-1)(c-b), 1, (c-b)+1, 2(c-b)+1, \dots, (M-1)(c-b)+1$, *etc.* we obtain a parity-check matrix of an equivalent (J, L)-regular LDPC block code constructed from circulant matrices

$$H^{(\mathrm{c})} = \begin{pmatrix} I_{w_{00}} & I_{w_{01}} & \cdots & I_{w_{0(c-1)}} \\ I_{w_{10}} & I_{w_{11}} & \cdots & I_{w_{1(c-1)}} \\ \cdots & \cdots & \cdots & \cdots \\ I_{w_{(c-b-1)0}} & I_{w_{(c-b-1)1}} & \cdots & I_{w_{(c-b-1)(c-1)}} \end{pmatrix} \tag{3.3}$$

where w_{ij} are the entries of the degree matrix W and $I_{w_{ij}}$ denotes an $M \times M$ *circulant matrix*, that is, an identity matrix with its rows shifted cyclically to the left by w_{ij} positions. Note, that the (J, L)-regular LDPC block code determined by $H^{(c)}$ is not quasi-cyclic, although it is equivalent to the QC block code determined by $H^{(\mathrm{tb})}$.

■ **Example 3.1:**
Consider a rate $R = 1/4$ convolutional code $\mathcal{C}$ whose parity-check matrix consists only of monomial entries, for example,

$$H(D) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & D & D \\ 1 & D & 1 & D \end{pmatrix} \tag{3.4}$$

with degree matrix

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Tailbiting (3.4) to length $M = 2$, yields the tailbiting 6×8 parity-check matrix of a QC LDPC block code

$$H^{(\mathrm{tb})} = \begin{array}{c} \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \left(\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}\right) \end{array} \tag{3.5}$$

In particular, every cyclic shift of a codeword by $c = 4$ positions modulo $Mc = 8$ yields another codeword. By reordering the columns as v_0, v_4, v_1, v_5, v_2, v_6, v_3, v_7 and the rows as c_0, c_3, c_1, c_4, c_2, c_5, we obtain an equivalent rate $R = 1 - 6/8$ $(3,4)$-regular LDPC block code whose parity-check matrix is based on circulants, that is,

$$H^{(\mathrm{c})} = \begin{array}{c} \\ c_0 \\ c_3 \\ c_1 \\ c_4 \\ c_2 \\ c_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_0 & v_4 & v_1 & v_5 & v_2 & v_6 & v_3 & v_7 \end{array} \\ \left(\begin{array}{cc|cc|cc|cc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}\right) \end{array} \tag{3.6}$$

Interpreting the parity-check matrix $H^{(\mathrm{tb})}$ (3.5) as a biadjacency matrix, yields a Tanner graph $\mathcal{G}$ which is illustrated in Figure 3.1 and consists of 8 symbol nodes and 6 constraint nodes with girth $g = 4$.

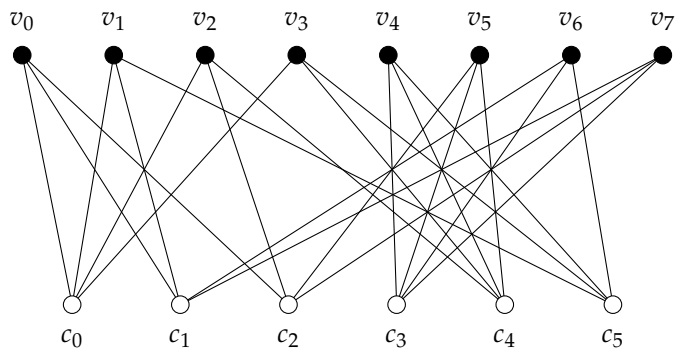


Figure 3.1: Tanner graph with girth $g = 4$ consisting of 8 symbol nodes v_i, $i = 0, 1, \ldots, 7$, and 6 constraint nodes c_j, $j = 0, 1, \ldots, 5$.

3.2 BASE MATRICES, VOLTAGES, AND THEIR GRAPHS

A binary matrix B is called *base matrix* for a tailbiting LDPC block code $\mathcal{B}$ if its parent convolutional code $\mathcal{C}$ with parity-check matrix $H(D)$ has only monomial or zero entries and satisfies

$$B = H(D)\big|_{D=1} \tag{3.7}$$

that is, all nonzero entries in $H(D)$ are replaced by $D^0 = 1$. Note that different tailbiting LDPC block codes can have the same base matrix B.

The corresponding *base graph*[1] $\mathcal{G}_\mathrm{B}$ follows as the bipartite graph, whose biadjacency matrix is given by the base matrix B. Denote the girth of such a base graph by g_B.

Next, consider the additive group $(\Gamma, +)$ where $\Gamma = \{\gamma\}$. The corresponding *voltage graph* $\mathcal{G}_\mathrm{V} = \{\mathcal{E}_\mathrm{B}, \mathcal{V}_\mathrm{B}, \Gamma\}$ is obtained from a base graph $\mathcal{G}_\mathrm{B} = \{\mathcal{E}_\mathrm{B}, \mathcal{V}_\mathrm{B}\}$ by assigning a voltage value $\gamma(e, \zeta, \zeta')$ to every edge e which connects the vertices ζ and ζ' and satisfies the property $\gamma(e, \zeta, \zeta') = -\gamma(e, \zeta', \zeta)$. The definitions of a path and a cycle in a voltage graph coincide with those in a regular graph, except for the additional restriction that two neighboring edges may not connect the same nodes in reversed order. Moreover, the *voltage of a path* and the *voltage of a cycle* within a voltage graph, follows as the sum of all edge voltages involved. Note that even though the edges of a voltage graph are not directed, their edge voltage depends on the passing direction.

Let $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$ be a *lifted graph* obtained from a voltage graph $\mathcal{G}_\mathrm{V}$, where $\mathcal{E} = \mathcal{E}_\mathrm{B} \times \Gamma$ and $\mathcal{V} = \mathcal{V}_\mathrm{B} \times \Gamma$. Two vertices (ζ, γ) and (ζ', γ') are connected in

[1] The terminology »base graph« originates from graph theory and is equivalent to the terminology *protograph* or *seed graph*.

the lifted graph by an edge if and only if ζ and ζ' are connected in the voltage graph $\mathcal{G}_{\mathrm{V}}$ with an edge having the voltage value $\gamma(e, \zeta, \zeta') = \gamma - \gamma'$.

Clearly, every cycle in the lifted graph corresponds to a cycle in the voltage graph with zero voltage. Consequently, the girth g_{V} of a voltage graph follows as the length of its shortest cycle with voltage zero, which is equal to its free girth g_{free}. Note that such a cycle does not necessarily have to be simple.

When searching for QC LDPC block codes, as for example described in Section 3.4, we construct a base graph $\mathcal{G}_{\mathrm{B}}$ and determine its corresponding voltage graph $\mathcal{G}_{\mathrm{V}}$ by assigning suitable edge voltages. In particular, these edge voltages are identical to the monomial degrees w_{ij} of the degree matrix W. To be more precise, denote the edge voltage from symbol node v_i to constraint node c_j and vice versa by

$$\begin{array}{lll} \mu_{v_i c_j} & \text{with } c_j \in \mathcal{A}(v_i) & (v_i \to c_j) \\ \mu_{c_j v_i} & \text{with } v_i \in \mathcal{A}(c_j) & (c_j \to v_i) \end{array}$$

where for parity-check matrices $H(D)$ with only monomial entries, the edge voltage $\mu_{v_i c_j}$ determines the monomial degree w_{ij} according to

$$\begin{cases} \mu_{v_i c_j} = -w_{ji} \\ \mu_{c_j v_i} = w_{ji} \end{cases} \tag{3.8}$$

When searching for LDPC convolutional codes with a given free girth g_{free}, integer edge voltages are used, that is, the monomial degrees are chosen from an infinite additive group. However, when searching for QC LDPC block codes with a given girth g, obtained by tailbiting a parent convolutional code to length M, a group of modulo M residues is used and (3.8) is replaced by

$$\begin{cases} \mu_{v_i c_j} = -w_{ji} \mod M \\ \mu_{c_j v_i} = w_{ji} \mod M \end{cases} \tag{3.9}$$

■ **Example 3.1 (Cont'd):**
Consider the rate $R = 1/4$ $(3,4)$-regular LDPC convolutional code $\mathcal{C}$ whose parity-check matrix is given (3.4). Its base matrix follows as

$$B = H(D)\Big|_{D=1} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{3.10}$$

while the corresponding base graph $\mathcal{G}_{\mathrm{B}}$ is illustrated in Figure 3.2.

By labeling the edges of the base graph $\mathcal{G}_{\mathrm{B}}$ according to (3.8), the corresponding voltage graph $\mathcal{G}_{\mathrm{V}}$ is obtained as shown in Figure 3.3. The edge

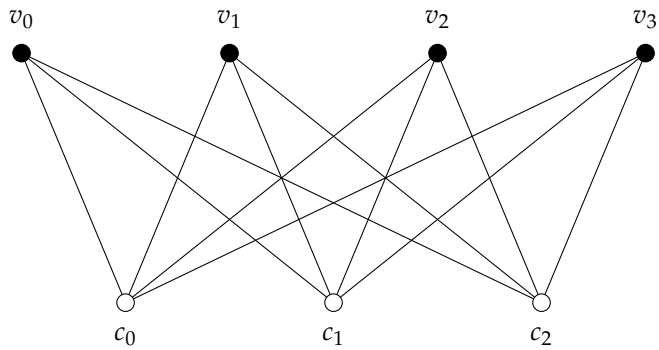


Figure 3.2: Base graph $\mathcal{G}_{\mathrm{B}}$ for the rate $R = 1/4$ QC $(3,4)$-regular LDPC block code used in Example 3.1.

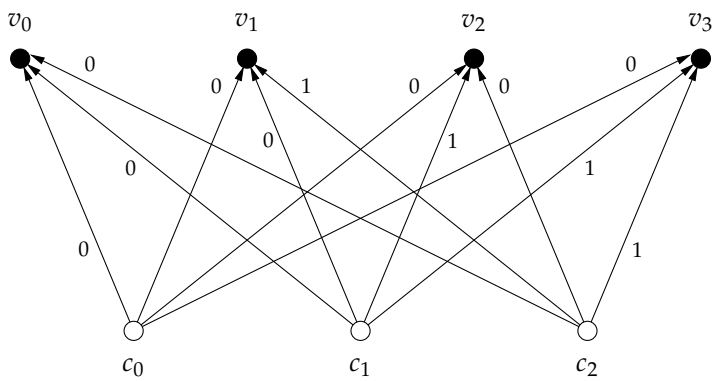


Figure 3.3: Voltage graph $\mathcal{G}_{\mathrm{V}}$ for the rate $R = 1/4$ QC $(3,4)$-regular LDPC block code used in Example 3.1.

from, for example, constraint node c_1 to symbol node v_2 is labeled according to

$$\mu_{c_1 v_2} = -\mu_{v_2 c_1} = w_{12} = 1$$

and hence the parity-check polynomial $h_{12}(D) = D^{w_{12}} = D$. The girth of this voltage graph follows as $g_{\mathrm{V}} = 4$ (for example, $v_0 \to c_0 \to v_1 \to c_1 \to v_0$). The free girth of the infinite Tanner graph, corresponding to the parent convolutional code $\mathcal{C}$, is determined by the convolutional parity-check matrix $H(D)$ and is equal to the girth of the voltage graph; and hence we conclude that $g_{\mathrm{free}} = g_{\mathrm{V}} = 4$.

Tailbiting the parity-check matrix $H(D)$ to length $M = 2$, yields the 6×8 tailbiting parity-check matrix of a QC LDPC block code given in (3.5), whose lifted graph is equivalent to the Tanner graph illustrated in Figure 3.1. From a graph theoretical point of view this corresponds to duplicating the base graph

$M = 2$ times and directly connecting the node ζ in the γth copy with another node ζ' in the γ'th copy if and only if ζ and ζ' are connected in the voltage graph with an edge having the voltage value $\gamma - \gamma'$.

For example, the constraint node c_1 in each of the $M = 2$ copies is cyclically connected to the symbol node v_2 in the next copy, since these two nodes are connected in the voltage graph by an edge labeled with the edge voltage $\mu_{c_1 v_2} = 1$.

3.3 BOUNDS ON THE GIRTH AND THE MINIMUM DISTANCE

Every voltage graph corresponds to a convolutional parity-check matrix $H(D)$, and hence describes an infinite sequence of tailbiting QC LDPC block codes with different truncations lengths M in a rather compact form. Consequently, such codes can be efficiently analyzed using their voltage graph representation, and search algorithms can directly consider an infinite sequence of related QC LDPC block codes. In the following, different relations between parameters of the base graph $\mathcal{G}_{\mathrm{B}}$, voltage graph $\mathcal{G}_{\mathrm{V}}$, and the corresponding infinite sequence of QC LDPC block codes will be established:[2]

Theorem 3.1 The minimum distance $d_{\min}$ and the girth g of a rate $R = K/N$ QC LDPC block code $\mathcal{B}$ obtained from a rate $R = b/c$ convolutional code $\mathcal{C}$ with free distance d_{free} and girth g_{free} by tailbiting to length M are upper-bounded by the inequalities

$$\begin{aligned} d_{\min} &\leq d_{\text{free}} \\ g &\leq g_{\text{free}} \end{aligned}$$

Proof. The first statement follows directly from the fact that any code sequence $\boldsymbol{v}(D)$ of the tailbiting block code $\mathcal{B}$, obtained from the parity-check matrix $H(D)$ of the parent convolutional code $\mathcal{C}$, satisfies

$$\boldsymbol{v}(D)H^{\mathrm{T}}(D) = \mathbf{0} \mod (D^M - 1) \tag{3.11}$$

Since the parent convolutional code $\mathcal{C}$ satisfies (3.11) without reduction modulo $(D^M - 1)$ and reducing modulo $(D^M - 1)$ does not increase the weight of a polynomial, the first statement follows.

For the second statement we consider the voltage graph $\mathcal{G}_{\mathrm{V}}$ representation of the parent convolutional code $\mathcal{C}$ with girth $g_{\mathrm{V}} = g_{\text{free}}$ together with the Tanner graph representation of the QC LDPC block code $\mathcal{B}$ with girth g. Similar to the relations between the free distance d_{free} and the minimum distance

[2]Note that similar relations were previously derived in [BKS09] for the special case of QC $(J = 2, L)$-regular LDPC block codes.

$d_{\min}$, there exists a relation between each cycle within the voltage graph $\mathcal{G}_{\mathrm{V}}$ of the parent convolutional code and the Tanner graph $\mathcal{G}$ of the corresponding block code obtained by tailbiting to length M. As mentioned previously, the sum of all edge voltages for every cycle in $\mathcal{G}_{\mathrm{V}}$ is equal to zero. Similarly, every cycle in $\mathcal{G}$ corresponds to a cycle in $\mathcal{G}_{\mathrm{V}}$ such that its edge voltages sum up to zero modulo M. From the same argument as before it follows that

$$g \leq g_{\mathrm{V}} = g_{\mathrm{free}} \qquad \blacksquare$$

Consider the base graph of a QC $(J \geq 3, L)$-regular LDPC convolutional code with girth g_{B} and let d_s denote the sth generalized minimum Hamming distance of the linear $\big(JMc,\ M((J-2)c+b)\big)$ block code $\mathcal{B}_{\mathrm{T}}$ determined by the parity-check matrix which corresponds to the incidence matrix of the Tanner graph. That is, d_s corresponds to the minimal number of nontrivial code symbols of an s-dimensional linear subcode.

Theorem 3.2 There exists a tailbiting length M and a voltage assignment, such that the girth g of the Tanner graph for the corresponding tailbiting block code of length Mc satisfies the inequality

$$g \geq 2 \max \left\{ g_{\mathrm{B}} + \lceil g_{\mathrm{B}}/2 \rceil, d_2 \right\} \tag{3.12}$$

where d_2 is the second generalized minimum Hamming distance, that is, the minimum support of a subcode of $\mathcal{B}_{\mathrm{T}}$ having dimension two. We have equality in (3.12), if the underlying base graph consists of two connected cycles, having at least one common vertex.

Proof. According to Theorem 3.1, any cycle in the Tanner graph of a QC LDPC block code corresponds to a cycle of the same length in the voltage graph. Since the labels of the voltage Tanner graph can be freely chosen, it is enough to prove that there is no zero cycle whose length is shorter than $2(g_{\mathrm{B}} + \lceil g_{\mathrm{B}}/2 \rceil)$, that is, no such cycle whose voltage is zero regardless of the voltage assignment of the base graph[3].

The number of times each edge in such a cycle of the voltage graph is passed in different directions has to be even. Such a cycle can not be simple, since in a simple cycle each edge is passed in one direction only. Hence, this cycle passes through the vertices of a subgraph which contains at least two different cycles, corresponding to two different nonzero codewords. The minimum distance of the code $\mathcal{B}_{\mathrm{T}}$ determined by the parity-check matrix which is the incident matrix to the base graph is equal to the girth g_{B} (cf. [BKS09]). According to the Griesmer bound, the smallest length of a linear code with

[3]Note that a cycle whose voltage is zero regardless of the voltage assignment of the base graph is also known as an *inevitable cycle* [KNCS07] or *balanced cycle* [O'S06].

two nonzero codewords of minimum distance d is $d + \lceil d/2 \rceil$, and hence the first lower bound of inequality (3.12) follows.

Next, consider the second lower bound. The definition of the second generalized minimum Hamming distance implies that the smallest subgraph with two cycles consists of at least d_2 edges. Hence, the second of the two lower bounds gives the precise value of the girth of a subgraph containing two connected cycles, having at least one common symbol node. Otherwise, the second generalized minimum Hamming distance d_2 is a lower bound. ■

These bounds are tighter than the $3g_{\mathrm{B}}$ bound published in [KNCS07] and [KW08] but not tight if the shortest nonsimple cycle consists of two simple cycles connected by a path.

In 2004, Fossorier [Fos04] derived an upper bound on the achievable girth g for LDPC block codes constructed from all-one base matrices. Using the same approach, this upper bound can be reformulated to include base matrices with zero elements as follows:

Theorem 3.3 Consider the parity-check matrix $H(D)$ of a rate $R = b/c$ convolutional code with base matrix B. Denote the corresponding base graph by $\mathcal{G}_{\mathrm{B}}$ and let B' be the 2×3 submatrix

$$B' = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3.13}$$

If the base matrix B, after possibly reordering its rows and columns, contains the submatrix B', then the girth g_{V} of the corresponding voltage graph $\mathcal{G}_{\mathrm{V}}$ is upper-bounded by

$$g_{\mathrm{V}} \leq 12 \tag{3.14}$$

regardless of the voltage assignment.

Proof. The subgraph determined by the 2×3 submatrix B' contains 3 symbol nodes, 2 constraint nodes, and 6 edges. Moreover, there exist 3 shortest cycles of length 4. Thus, the base graph $\mathcal{G}_{\mathrm{B}}$ has girth $g_{\mathrm{B}} = 4$ and its second generalized Hamming distance is $d_2 = 6$. Applying Theorem 3.2, we obtain the precise value of the achievable girth as $2d_2 = 12$, which completes the proof. ■

The following upper bound on the minimum distance, was initially derived by MacKay and Davey [MD99] for LDPC block codes constructed from all-one base matrices and later reformulated by Bocharova et al. [BKSS09] to include base matrices with zero elements. Consider the parity-check matrix $H(D)$ of a rate $R = b/c$ (J, L)-regular LDPC convolutional code $\mathcal{C}$ with free distance

d_{free}. Tailbiting to length M yields a QC LDPC block code of block length Mc whose minimum distance $d_{\min}$ can be upper-bounded by

$$d_{\min} \leq d_{\text{free}} \leq (c-b+1)! \tag{3.15}$$

which simplifies to $(J+1)!$ for LDPC block codes constructed from all-one base matrices.

Next, we shall consider an upper bound on the free distance d_{free} of rate $R = b/c$ (J, L)-regular LDPC convolutional codes by Smarandache et al. [SV11][4]. Denote by $\mathcal{J}$ a subset of $c-b+1$ columns of $H(D)$ and consider its convolutional subcode, whose code c-tuples have zeros at all positions whose corresponding columns are not included in $\mathcal{J}$. Deleting those zero positions yields the shortened convolutional code $\tilde{\mathcal{C}}^{\mathcal{J}}$ with rate $\tilde{R} = 1/(c-b+1)$, free distance $\tilde{d}_{\text{free}} \geq d_{\text{free}}$ and generator matrix

$$\widetilde{G}^{\mathcal{J}}(D) = \begin{pmatrix} \tilde{\boldsymbol{g}}_0^{\mathcal{J}}(D) & \dots & \tilde{\boldsymbol{g}}_{c-b-1}^{\mathcal{J}}(D) & \tilde{\boldsymbol{g}}_{c-b}^{\mathcal{J}}(D) \end{pmatrix} \tag{3.16}$$

Without loss of generality we can assume that $\tilde{\boldsymbol{g}}_{c-b}^{\mathcal{J}}(D) \neq 0$, and hence an equivalent systematic generator matrix follows from Cramer's rule and (1.66) as

$$\widetilde{G}_{\text{sys}}^{\mathcal{J}}(D) = \begin{pmatrix} \dfrac{\Delta_0^{\mathcal{J}}}{\Delta_{c-b}^{\mathcal{J}}} & \dots & \dfrac{\Delta_{c-b-1}^{\mathcal{J}}}{\Delta_{c-b}^{\mathcal{J}}} & 1 \end{pmatrix} \tag{3.17}$$

or, alternatively,

$$\widetilde{G}^{\mathcal{J}}(D) = \begin{pmatrix} \Delta_0^{\mathcal{J}} & \dots & \Delta_{c-b-1}^{\mathcal{J}} & \Delta_{c-b}^{\mathcal{J}} \end{pmatrix} \tag{3.18}$$

where $\Delta_i^{\mathcal{J}}$ denotes the determinant of the square matrix obtained from $\tilde{H}(D)$ by removing column i. Thus, the free distance $\tilde{d}_{\text{free}}$ can be upper-bounded by

$$\tilde{d}_{\text{free}} \leq \sum_{i=0}^{c-b} w_{\text{H}}\left(\Delta_i^{\mathcal{J}}\right) \tag{3.19}$$

where $w_{\text{H}}(\Delta_i^{\mathcal{J}})$ denotes the Hamming weight of the polynomial determinant $\Delta_i^{\mathcal{J}}$. In particular, the free distance d_{free} of the rate $R = b/c$ (J, L)-regular LDPC convolutional code $\mathcal{C}$ follows as

$$d_{\text{free}} \leq \min_{\mathcal{J}} \sum_{i=0}^{c-b} w_{\text{H}}\left(\Delta_i^{\mathcal{J}}\right) \tag{3.20}$$

where the minimization is carried out over all possible subsets of $c-b+1$ columns of $H(D)$.

[4]This upper bound is a generalized version of the upper bound by Bocharova et al. [BKSS09] which is only valid for LDPC convolutional codes obtained from all-one base matrices.

3.4 SEARCHING FOR QC LDPC BLOCK CODES WITH LARGE GIRTH

The problem of finding QC LDPC block codes with large girth for a wide range of code rates was previously considered by several authors, for example, in [Fos04], [TSS+04], [MKL06], [O'S06], [KNCS07], and [EG10]. Codes with girth smaller than or equal to 12 are constructed in [TSF01], [Fos04], [TSS+04], and [MKL06], while [O'S06] gives examples of rather short codes with girth 14. QC (J, L)-regular LDPC block codes with girth up to 18 and $J \geq 3$ are presented in [EG10]. In particular it is shown that QC LDPC block codes with girth greater than or equal to 14 and block length between 34000 and 92000 outperform random codes of the same block length and code rate.

When searching for QC (J, L)-regular LDPC block codes with large girth, we start with a base graph corresponding to an all-one base matrix of size $J \times L$. Then, suitable voltage assignments, that is, edge voltages based on the group of nonnegative integers are determined, such that the girth of this voltage graph is greater than or equal to a given target girth g. Since shorter block codes are of particular interest, we replace all edge voltages by their corresponding modulo M residuals, where the value M is minimized under the restriction of preserving the girth g.

Applying the previously introduced concept of biadjacency matrices yields the corresponding degree matrix W, and hence the parity-check matrix $H(D)$ of a convolutional code whose bipartite graph has girth $g = g_{\text{free}}$. By tail-biting the convolutional parity-check matrix to lengths M, we obtain a rate $R = Mb/Mc$ QC LDPC block code whose parity-check matrix is equal to the biadjacency matrix of a bipartite graph with girth g.

Note that the process of determining suitable voltage assignments for a given base graph with target girth g can be split into the following two separate steps:

(i) Constructing a list of all (voltage) inequalities which describe all cycles of length smaller than g within the base graph (cf. Subsection 3.4.1).

(ii) Searching for a suitable voltage assignment of the base graph such that all inequalities are satisfied (cf. Subsection 3.4.2).

The efficiency of the second step, searching for a suitable voltage assignment, depends on the chosen representation for the list of inequalities determined during the first step. In general, when searching for all cycles of length g roughly $(J-1)^g$ different paths have to be considered. However, using a similar idea as in Section 2.4 when searching for a path within a trellis, a tree of maximum depth $g/2$ is created, used to search for identical nodes, and hence reducing the complexity to roughly $(J-1)^{g/2}$.

3.4.1 STEP I: CREATING A TREE STRUCTURE

In the first step of the algorithm, we consider a base graph $\mathcal{G}_{\mathrm{B}}$ with c symbol nodes and $c-b$ constraint nodes which corresponds to an all-one base matrix of size $(c-b) \times c$. For each of its c symbol nodes v_i, $i = 0, 1, \ldots, c-1$, we create a separate subtree according to the following description.

First, we will introduce some notations. A node in such a tree is denoted by ζ, has a unique parent node ζ^{p}, and is connected to other nodes according to the underlying base graph $\mathcal{G}_{\mathrm{B}}$. Since the base graph $\mathcal{G}_{\mathrm{B}}$ is bipartite (cf. Section 2.5), every node $\zeta \in \mathcal{V}_i$, is only connected to nodes $\zeta' \in \mathcal{V}_j$ with $i, j \in \{0, 1\}$, $i \neq j$, where $\mathcal{V}_0$ and $\mathcal{V}_1$ denote the sets of symbol and constraint nodes, respectively. In other words, a symbol node is only connected to a constraint nodes and vice versa.

Moreover, every node ζ is characterized by its depth $\ell(\zeta)$ and its number $n(\zeta)$, where $n(\zeta) = i$ follows from $\zeta = v_i \in \mathcal{V}_0$ or $\zeta = c_i \in \mathcal{V}_1$ depending on whether its depth $\ell(\zeta)$ is even or odd. In particular, note that every node satisfies

$$\zeta \in \mathcal{V}_i \quad \text{where } i = \ell(\zeta) \text{ modulo-2}$$

Next, c separate subtrees $\mathcal{T}_i$, $i = 0, 1, \ldots, c-1$, are grown, where the root node $\zeta_{i,\mathrm{root}}$ of the ith subtree is initialized with the ith symbol node $\zeta_{i,\mathrm{root}} = v_i \in \mathcal{V}_0$ and depth $\ell(\zeta_{i,\mathrm{root}}) = 0$. Since the girth of a bipartite graph is always even, it is sufficient to grow each subtree up to depth $(g-2)/2$ in order to obtain all possible cycles of length smaller than g. In other words, every node $\zeta \in \mathcal{V}_i$ at depth $\ell(\zeta) = n < (g-2)/2$ with $i = n$ modulo 2 is connected by different branches to all nodes $\zeta' \in \mathcal{V}_{(i+1) \bmod 2}$ at depth $n+1$ according to the underlying base graph $\mathcal{G}_{\mathrm{B}}$, except for the node ζ^{p} which is already connected to ζ at depth $n-1$.

Clearly, all subtrees together contain all possible paths of length up to $g-2$ in the voltage graph. Thus, voltage inequalities for all cycles of length at most $g-2$ can be constructed by labeling all edges with their edge voltages according to (3.8). Then, the voltage for node ζ in the ith subtree $\mathcal{T}_i$ follows as the sum of the edge voltages along the path $\zeta_{i,\mathrm{root}} \to \zeta$. Moreover, for every node pair (ζ, ζ') in the same subtree i with the same number $n(\zeta) = n(\zeta')$ and depth $\ell(\zeta) = \ell(\zeta')$ but different parent nodes $\zeta^{\mathrm{p}} \neq \zeta'^{\mathrm{p}}$, the corresponding voltage inequality follows as the difference between the voltages for the paths from $\zeta_{i,\mathrm{root}}$ to ζ and ζ', respectively, that is,

$$(\zeta_{i,\mathrm{root}} \to \zeta) - (\zeta_{i,\mathrm{root}} \to \zeta') \neq 0 \tag{3.21}$$

If and only if there exists a cycle of length $g' < g$, then at depth $g'/2$ there exists at least one such pair of nodes (ζ, ζ'), whose corresponding voltage inequality is not satisfied; otherwise all voltage inequalities are satisfied.

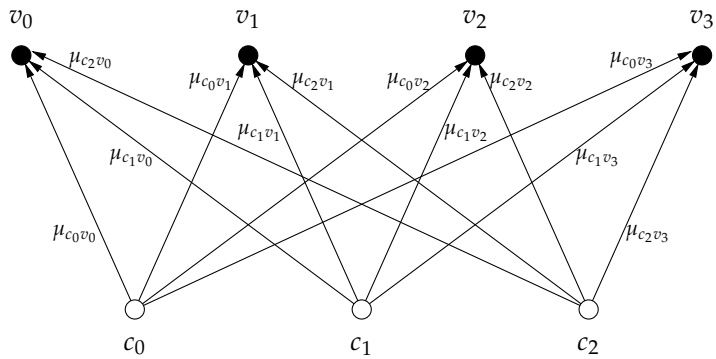


Figure 3.4: General voltage graph $\mathcal{G}_{\mathrm{V}}$ for the rate $R = 1/4$ QC $(3,4)$-regular LDPC block code used in Example 3.2.

■ **Example 3.2:**
Consider the rate $R = 1/4$ $(3,4)$-regular LDPC convolutional code from Example 3.1 whose base graph with four symbol nodes $v_i \in \mathcal{V}_0$, $i = 0,1,2,3$, and three constraint nodes $c_i \in \mathcal{V}_1$, $i = 0,1,2$, is given in Figure 3.2.

By labeling its edges with the edge voltages $\mu_{c_j v_i}$ the corresponding general voltage graph as shown in Figure 3.4 is obtained. Based on this graph, $c = 4$ separate subtrees are created, where the root node ζ_{root} of the ith subtree is initialized with the symbol nodes v_i, $i = 0,1,2,3$.

Within this example, we assume that we are interested in finding a voltage assignment, that is, entries of the degree matrix W, such that the corresponding voltage graph has at least girth $g = 6$. Hence, it is sufficient to extend each of the $c = 4$ subtrees up to depth $(g-2)/2 = 2$.

Consider for example the subtree with root node v_0 as illustrated in Figure 3.5 where for each node ζ at depth $\ell(\zeta) = 2$ the corresponding voltage for the path $\zeta_{i,\mathrm{root}} \to \zeta$ is given. Clearly, at depth $\ell(\zeta) = 1$ there are no identical nodes, while at depth $\ell(\zeta) = 2$ there are $3 \times \binom{3}{2} = 9$ pairs of identical nodes $(n(\zeta) = n(\zeta'))$ with different parents.

For example, the voltage inequalities obtained by checking all such node pairs (ζ, ζ') with $\zeta = v_1$, that is, $n(\zeta) = 1$, at depth $\ell(\zeta) = 2$ in the subtree starting with symbol node v_0 as illustrated in Figure 3.5, are

$$\begin{aligned}
&\big(-\mu_{c_0v_0} + \mu_{c_0v_1}\big) - \big(-\mu_{c_1v_0} + \mu_{c_1v_1}\big) \neq 0 && {\scriptstyle (v_0 \to c_0 \to v_1) - (v_0 \to c_1 \to v_1) \neq 0} \\
&\big(-\mu_{c_0v_0} + \mu_{c_0v_1}\big) - \big(-\mu_{c_2v_0} + \mu_{c_2v_1}\big) \neq 0 && {\scriptstyle (v_0 \to c_0 \to v_1) - (v_0 \to c_2 \to v_1) \neq 0} \\
&\big(-\mu_{c_1v_0} + \mu_{c_1v_1}\big) - \big(-\mu_{c_2v_0} + \mu_{c_2v_1}\big) \neq 0 && {\scriptstyle (v_0 \to c_1 \to v_1) - (v_0 \to c_2 \to v_1) \neq 0}
\end{aligned}$$

Taking into account that similar subtrees are constructed using the remaining three symbol nodes v_1, v_2 and v_3 as their root node ζ_{root}, there exist in total $4 \times 9 = 36$ node pairs, that is voltage inequalities. However, among all those 36 voltage inequalities, there are only 18 unique ones.

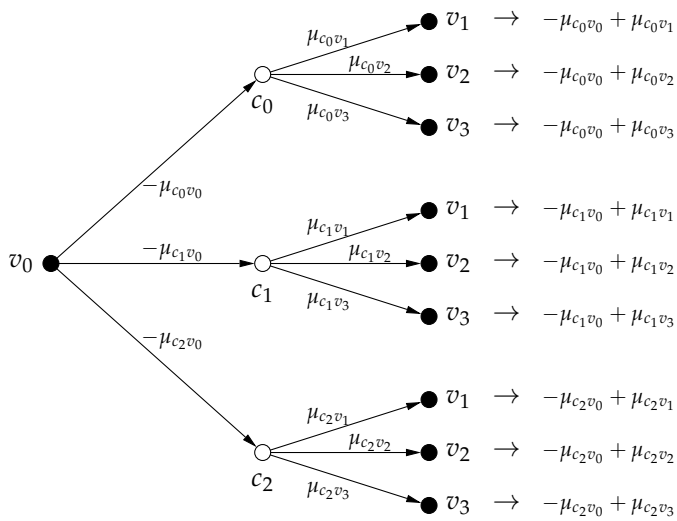


Figure 3.5: The subtree $\mathcal{T}_0$ with maximum depth two, starting with symbol node v_0, and constructed from the general voltage graph in Figure 3.4 used in Example 3.2.

Algorithm TR (Constructing a tree representation)

---

1. Grow c separate subtrees according to the underlying base graph up to depth $g/2-1$, with the root node $\zeta_{i,\text{root}}$ of the ith subtree being initialized with $\zeta_{\text{root}} = v_i \in \mathcal{V}_0$ and depth $\ell(\zeta_{\text{root}}) = 0$.
2. Extend every node $\zeta \in \mathcal{V}_i$ at depth $\ell(\zeta) = n < g/2-1$ with $i = n \bmod 2$ with branches connected to all nodes $\zeta' \in \mathcal{V}_{i+1 \bmod 2}$ at depth $n+1$ according to the underlying base graph, except ζ^{P} which is already connected to ζ at depth $n-1$. Denote the set of all nodes within the ith subtree by $\mathcal{T}_i$.

3.4.2 STEP II: SEARCHING FOR A SUITABLE VOLTAGE ASSIGNMENT

As mentioned previously, all c subtrees $\mathcal{T}_i$, $i = 0, 1, \ldots, c-1$, with maximum depth $g/2-1$, represent all possible cycles of length smaller than or equal to $g-2$, together with their corresponding voltage inequalities.

Note however, that the same cycle might be found several times within all c subtrees. Additionally, different cycles can correspond to the same voltage inequality.

To determine a suitable voltage assignment, a list $\mathcal{L}$ of node pairs (ζ, ζ') is created, containing all unique voltage inequalities of all c subtrees $\mathcal{T}_i$, $i =$

$0, 1, \ldots, c-1$. In other words, among all identical voltage inequalities we keep only one. Using this list, every subtree $\mathcal{T}_i$, can be reduced in a similar way by removing all nodes, not participating in any of the cycles in $\mathcal{L}$. Denote such a reduced subtree by $\mathcal{T}_{i,\min}$. That is, remove all nodes of the subtree $\mathcal{T}_i$, which only participate in already known cycles or in new cycles with already known voltage inequalities.

In the following, we present two different approaches for finding suitable voltage assignments. They are based on using either the list $\mathcal{L}$ or the reduced subtrees $\mathcal{T}_{i,\min}$, and hence referred to as the VA-L or the VA-T algorithm, respectively.

The VA-L algorithm labels all edges of the reduced subtrees $\mathcal{T}_{i,\min}$, $i = 0, 1, \ldots, c-1$, with a set of randomly chosen edge voltages. For every node pair (ζ, ζ') within the list $\mathcal{L}$, the voltage of the corresponding cycle is calculated according to (3.21) as the difference of the path voltages $\zeta_{i,\text{root}} \to \zeta$ and $\zeta_{i,\text{root}} \to \zeta'$. If all inequalities are satisfied, that is, if none of the path voltage differences is equal to zero, the girth of the LDPC block code obtained from combining the underlying base graph with such a voltage assignment is greater than or equal to g.

Contrary, the VA-T algorithm discards the list $\mathcal{L}$ and considers only the c reduced subtrees $\mathcal{T}_{i,\min}$. After labeling their edges with a set of randomly chosen voltages, the nodes ζ of each subtree are sorted according to their path voltage $\zeta_{i,\text{root}} \to \zeta$. If there exists no pair of nodes (ζ, ζ') with the same path voltage, number $n(\zeta) = n(\zeta')$, and depth $\ell(\zeta) = \ell(\zeta')$, but different parent nodes $\zeta^{\text{p}} \neq \zeta'^{\text{p}}$, the girth of the LDPC code obtained by combining the underlying base graph with such a voltage assignment is greater than or equal to g.

Algorithm VA-L (Constructing a system of voltage inequalities and searching for an optimum voltage assignment using a list)

1. Create a list $\mathcal{L}$ of unique voltage inequalities for all node pairs (ζ, ζ') within all c subtrees $\mathcal{T}_i$, $i = 0, 1, \ldots, c-1$, with the same number $n(\zeta) = n(\zeta')$, depth $\ell(\zeta) = \ell(\zeta')$, but different parent nodes $\zeta^{\text{p}} \neq \zeta'^{\text{p}}$.
2. Reduce each of the c subtrees $\mathcal{T}_i$ by removing all nodes, which do not participate in any of the found cycles corresponding to the voltage inequalities in $\mathcal{L}$, and denote the reduced subtree by $\mathcal{T}_{i,\min}$.
3. Assign randomly chosen voltages to the edges of all trees and perform the following steps:
   (i) Find the path voltages for all paths leading from the root node $\zeta_{i,\text{root}}$ of the ith reduced subtree $\mathcal{T}_{i,\min}$ to all its nodes $\zeta \in \mathcal{T}_{i,\min}$, $i = 0, 1, \ldots, c-1$.

(ii) Determine the voltage inequality for all cycles $(\zeta, \zeta') \in \mathcal{L}$, as the difference of the corresponding path voltages in $\mathcal{T}_{i,\min}$, $i = 0, 1, \ldots, c-1$, computed previously.

(iii) If all voltage inequalities are satisfied, the girth of the voltage graph with such a voltage assignment is greater than or equal to the given girth g. Otherwise, assign new random voltages to all edges and go to step **VA-L-3**(i).

Algorithm VA-T (Constructing a system of voltage inequalities and searching for an optimum voltage assignment using a tree)

1. Construct the list $\mathcal{L}$ and the reduced subtrees $\mathcal{T}_{i,\min}$, $i = 0, 1, \ldots, c-1$, according to the steps **VA-L-1** and **VA-L-2**. However note that the corresponding list $\mathcal{L}$ is not needed to be stored.

2. Assign randomly chosen voltages to the edges of all trees and perform the following steps:

(i) Find the voltages for all paths from the root node $\zeta_{i,\text{root}}$ to all nodes within $\mathcal{T}_{i,\min}$, $i = 0, 1, \ldots, c-1$, and sort all elements within $\mathcal{T}_{i,\min}$ according to their voltages.

(ii) Search for a pair of nodes (ζ, ζ') in the sorted list with the same path voltage, number $n(\zeta) = n(\zeta')$, and depth $\ell(\zeta) = \ell(\zeta')$, but different parent nodes $\zeta^{\text{p}} \neq \zeta'^{\text{p}}$.

(iii) If no such pair exists, then the girth of the corresponding voltage graph with such a voltage assignment is greater than or equal to the given girth g. Otherwise, assign new random voltages to all edges and go to step **VA-T-2**(i).

COMPLEXITY CONSIDERATIONS

Denote the number of nodes within all reduced subtrees $\mathcal{T}_{i,\min}$, $i = 0, 1, \ldots, c-1$, and the number of unique inequalities within the list $\mathcal{L}$ by N_{T} and N_{L}, respectively, that is,

$$N_{\text{T}} = \sum_{i=1}^{c} |\mathcal{T}_{i,\min}| \quad \text{and} \quad N_{\text{L}} = |\mathcal{L}|$$

where $|\mathcal{X}|$ denotes the number of entries in the set $\mathcal{X}$.

The VA-L algorithm requires N_{T} summations for computing the path voltages and N_{L} comparisons for finding all possible cycles of length at most $g-2$, yielding a complexity estimate of $N_{\mathrm{T}}+N_{\mathrm{L}}$. On the other hand, the VA-T algorithm requires the same number of N_{T} summations for computing the path voltages, roughly $N_{\mathrm{T}}\log_2 N_{\mathrm{T}}$ operations for sorting the set, and N_{T} comparisons, leading to a total complexity estimate of $N_{\mathrm{T}}\log_2 N_{\mathrm{T}}$.

	$g=8$		$g=10$		$g=12$	
L	N_{T}	N_{L}	N_{T}	N_{L}	N_{T}	N_{L}
4	53	42	150	231	269	519
5	93	90	286	645	581	1905
6	142	165	485	1470	1060	5430
7	200	273	759	2919	1742	12999
8	267	420	1120	5250	2663	27426
9	343	612	1580	8766	3859	52614
10	428	855	2151	13815	5358	93735
11	522	1155	2845	20790	7210	157410
12	625	1518	3674	30129	9446	251889

Table 3.1: Complexity of a search for suitable voltage assignment for QC $(J=3,L)$-regular LDPC block codes with girth $g \leq 12$.

In Table 3.1 the values of N_{T} and N_{L} are given when searching for a suitable voltage assignment for (J,L)-regular rate $R=1-J/L$ QC LDPC convolutional codes with $J=3$, arbitrary $L \geq 4$, and girth g constructed from all-one base matrices. In this case, up to girth $g=10$, the VA-L algorithm should be used, while when searching for a voltage assignment with girth $g \geq 12$, the VA-T algorithm yields a better performance.

In the general case, all node pairs have to be considered and as N_{L} can be approximated by N_{T}^2, we conclude that the VA-T algorithm performs asymptotically better (when $N_{\mathrm{T}} \to \infty$).

3.5 MINIMUM DISTANCE OF QC LDPC BLOCK CODES

Usually the girth of the Tanner graph for a QC LDPC block code is considered to be the most important parameter affecting the BER performance of the BP decoding algorithm, since it determines the number of independent decoding iterations (cf. Section 2.6). Hence, most effort is focused on finding QC LDPC block codes with large girth, while their corresponding minimum distance is mostly unknown. Dolecek et al. showed in 2009 that the BER performance of the BP decoding algorithm at high SNRs depends on the structure and the size of the smallest absorbing sets, where the latter can be upper-bounded by the minimum distance of the corresponding block code.

From (1.26) it follows that the minimum distance $d_{\min}$ of any linear block code $\mathcal{B}$ with parity-check matrix H is equal to the minimum number of columns of H which sum up to zero. Hence, exploiting the regularity and sparsity of the parity-check matrix for QC LDPC block codes, the subset of all linear combination with a limited number of columns can be easily computed.

Consider an $M(c-b) \times Mc$ parity-check matrix $H^{(\mathrm{tb})}$ of a (J, L)-regular rate $R = Mb/Mc$ tailbiting block code $\mathcal{B}$ with block-length Mc (1.79). As mentioned in Section 3.1, the first c columns of $H^{(\mathrm{tb})}$ are repeated throughout the whole matrix in a cyclicly shifted manner. Hence, it is sufficient to construct c separate trees, starting with each of the first c columns of $H^{(\mathrm{tb})}$ as a root node, where each node ζ is characterized by its depth $\ell(\zeta)$ and its partial syndrome state column vector $\boldsymbol{\sigma}(\zeta)$.

The partial syndrome state column vector of the root node $\zeta_{i,\mathrm{root}}$ of the ith tree is initialized with the ith column of the corresponding parity-check matrix $H^{(\mathrm{tb})}$, that is, $\boldsymbol{\sigma}(\zeta_{i,\mathrm{root}}) = \boldsymbol{h}_i$, $i = 0, 1, \ldots, c-1$. Then, each tree is grown such, that every branch between any two nodes ζ and ζ' is labeled by a column vector $\boldsymbol{h}_j$, $j \neq i$, with $\boldsymbol{\sigma}(\zeta') = \boldsymbol{\sigma}(\zeta) + \boldsymbol{h}_j$, where every branch label on the path $\zeta_{i,\mathrm{root}} \to \zeta'$ occurs at most once.

Assuming that the kth position of the partial syndrome state column vector $\boldsymbol{\sigma}(\zeta)$ is nonzero, there exist at most $L-1$ columns which can cancel this nonzero position and have not been considered previously. Hence, at most $L-1$ branches per nonzero position are stemming from each node ζ.

However, such a tree would grow indefinitely, until all possible linear combinations have been found. Assuming the minimum distance to be restricted by $d_{\min} < t$, the maximum depth of each of the c trees is limited to $t-1$. Consequently, a node ζ at depth $\ell(\zeta)$ will not be extended, if the number of nonzero positions in its partial syndrome state column vector $\boldsymbol{\sigma}(\zeta)$ exceeds $J(t-\ell(\zeta)-1)$, since at most J nonzero entries can be canceled by each branch. Moreover, note that the partial syndrome state column vector for a node ζ can be expressed as $\boldsymbol{\sigma}(\zeta) = (\boldsymbol{\sigma}_0^{\mathrm{T}}(\zeta)\ \boldsymbol{\sigma}_1^{\mathrm{T}}(\zeta) \ldots \boldsymbol{\sigma}_{(c-b-1)}^{\mathrm{T}}(\zeta))^{\mathrm{T}}$, where $\boldsymbol{\sigma}_j(\zeta)$, $j = 0, 1, \ldots, c-b-1$ is a column vector of length M. Hence, by ini-

tially reordering the columns of the parity-check matrix $H^{(\mathrm{tr})}$ such that each of the $c-b$ nonoverlapping blocks of M rows contains at most one nonzero entry per column, the stopping criterion can be strengthened as follows: A node ζ at depth $\ell(\zeta)$ will not be extended, if the number of nonzero positions in each of its partial syndrome state column vectors $\boldsymbol{\sigma}_j(\zeta)$, $j = 0, 1, \ldots, c-b-1$ exceeds $t - \ell(\zeta) - 1$, since at most one nonzero entry in each block can be canceled by each branch.

Note that such a reordering of the parity-check matrix $H^{(\mathrm{tb})}$ corresponds to the parity-check matrix $H^{(c)}$ (3.3) of the equivalent (J, L)-regular LDPC block code constructed from circulant matrices, and hence is only feasible for such LDPC block codes.

Algorithm MD (Determine the minimum distance of a rate $R = b/c$ (J, L)-regular LDPC block code)

1. Assume a suitable restriction t on the minimum distance $d_{\min} < t$.
2. Grow c separate trees as follows:
   (i) Initialize the root node of the ith tree by $\boldsymbol{\sigma}(\zeta_{\mathrm{root},i}) = \boldsymbol{h_i}$ with depth $\ell(\zeta) = 0$, with $i = 0, 1, \ldots, c-1$.
   (ii) Extend all nodes ζ as long as the Hamming weights of their partial syndrome state column vector $w_{\mathrm{H}}(\boldsymbol{\sigma}(\zeta)) \leq J(t - \ell(\zeta) - 1)$.
   Note that, for QC LDPC block codes with blocks of M rows containing only a single nonzero entry, this criterion can be strengthen to $w_{\mathrm{H}}(\boldsymbol{\sigma}_j(\zeta)) \leq t - \ell(\zeta) - 1$, $j = 0, 1, \ldots, c-b-1$.
   (iii) The minimum distance $d_{\min}$ follows as
   $$d_{\min} = \min_{\zeta} \{\ell(\zeta) \mid \boldsymbol{\sigma}(\zeta) = \mathbf{0}\}$$
   If there exists no node ζ at any depth within any of the c trees having a zero partial syndrome state column vector, then the minimum distance of the corresponding block code is lower-bounded by $d_{\min} \geq t$.

3.6 ALL-ONE BASED QC LDPC BLOCK CODES

Using the previously introduced algorithms TR, VA-L, and VA-T, we obtain QC $(3, L)$-regular LDPC block codes with girth $g = 6$, 8, 10, and 12 as given in Tables 3.2 – 3.5. These LDPC block codes are constructed from all-one base matrices, and hence their parent convolutional parity-check matrices $H(D)$ contain only monomial entries.

To reduce the number of possible (permuted) voltage assignments, the following restrictions have been applied during the search process:

(i) Since the girth of a voltage graph $\mathcal{G}_{\mathrm{V}}$ is defined as the length of the shortest cycle with voltage zero, while the sign of the voltage depends on the direction of the edge, we can add an arbitrary offset to the voltages of all edges being connected to the same node. Hence, without loss of generality, the voltages of all edges connected to one symbol node as well as those connected to one constraint node can be set to zero by adjusting the edge voltages of their neighboring edges. (For consistency with QC LDPC block codes constructed using alternative base matrices, which shall be introduced later, we choose always the first symbol node and the last constraint node. This corresponds to a degree matrix with zeros in its first column and last row.)

From a graph theoretical point of view, this procedure is equivalent to creating a spanning tree $\mathcal{S}$ of the base graph $\mathcal{G}_{\mathrm{B}}$ and assigning the zero edge voltage all edges within $\mathcal{S}$ by adjusting the edge voltages of all adjacent edges not in $\mathcal{S}$. In particular, the spanning tree above is created such that it contains all edges connected to the first symbol node as well as all edges connected to the last constraint node.

For example, the degree matrix of the $(J = 3, L = 4)$ QC LDPC block code with girth $g = 8$ from Table 3.3 follows as

$$W = \begin{pmatrix} 0 & 1 & 4 & 6 \\ 0 & 5 & 2 & 3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

(ii) To further reduce the number of only permuted degree matrices, the following restrictions are applied:

- The first row is sorted in ascending order.
- When sorting the first and the second row in ascending order, the first row is lexicographically less than the second row.
- The maximum degree is less than the tailbiting length M for which there exists a QC $(J = 3, L)$-regular LDPC block code with given girth g.

(iii) QC $(J = 3, L = 4)$-regular LDPC block codes were found by exhaustive search over the previously defined set of restricted edge voltages.

(iv) QC $(J = 3, L = U)$-regular LDPC block codes with $U > 4$ were obtained by adding one additional, randomly chosen column to the best degree matrices of QC LDPC block codes with $L = U - 1$ and same girth g, obtained during the previous step. The maximum degree within this additional column is limited by twice the maximum degree of the previous code.

The obtained QC $(J = 3, L)$-regular LDPC block codes with girth $g = 6$, 8, 10, and 12 satisfying these restrictions are presented in Tables 3.2 – 3.5.

The first column contains L, the number of nonzero elements per row, which corresponds to the number of columns of the matrices $H(D)$ and W. Since all entries in the first column and the last row of the degree matrix W are zero, they are omitted in the submatrix W' which is given in the last column.

Next, consider the parity-check matrix $H(D)$ of the rate $R = 1 - J/L$ convolutional code $\mathcal{C}$ with only monomial entries corresponding to the degree matrix W. By tailbiting the semi-infinite parity-check matrix H to length M (given in the fourth column), we obtain the parity-check matrix $H^{(\mathrm{tb})}$ of an (N, K) block code $\mathcal{B}$ with minimum distance $d_{\min}$, where (N, K) and $d_{\min}$ follow from the second and third column, respectively. Due to linearly dependent rows in $H^{(\mathrm{tb})}$ the rate of the block code $\mathcal{B}$ might be larger than $1 - J/L$.

The codes presented in Tables 3.2 and 3.3 coincide with the QC LDPC block codes found by the »hill-climbing« algorithm in [WYD08], but we determined their previously unknown minimum distance using the MD algorithm. Tables 3.4 and 3.5 contain new QC $(J = 3, L)$-regular LDPC block codes, which, to the best of our knowledge, are shorter than the previously known codes obtained from all-one base matrices and presented in [TSF01], [O'S06], [WYD08], and [ZW10]. In particular, these codes are significantly shorter than the codes presented in [EG10], which are obtained from base matrices with zeros. However, due to the zeros in their base matrix, the minimum distance of their LDPC block codes can exceed $(J + 1)!$.

For example, using the MD algorithm, the minimum distance $d_{\min}$ of the $(444, 111)$ QC $(3, 4)$-regular LDPC block code with girth $g = 12$ in [EG10] is determined to be 28, while the corresponding code in Table 3.5, that is, the $(292, 75)$ QC $(3, 4)$-regular LDPC block code, has only minimum distance $d_{\min} = 24$, but shorter block length. Using the BEAST, the free distance of the parent convolutional code for the QC LDPC block code from [EG10] is calculated to be $d_{\mathrm{free}} = 46$. Hence, using the approach described in Section 3.4 together with a larger tailbiting length, it is possible to construct additional QC $(3, 4)$-regular LDPC block codes with minimum distance up to 46.

3.7 ALTERNATIVE CONSTRUCTIONS

In order to find QC $(J = 3, L)$-regular LDPC block codes with girth $g \geq 14$, the restriction on only monomial entries in the convolutional parity-check matrix $H(D)$ has to be relaxed to additionally include zero entries. Theorem 3.3 states as a necessary condition that the corresponding base matrix B must not contain a 2×3 all-one submatrices, while according to Theorem 3.2 such a code exists if the base graph has the girth g_{B} satisfying (3.12).

L	(N, K)	$d_{\min}$	M	W'
short codes				
4	$(20, 7)$	6	5	$1, 2, 4$ $3, 1, 2$
5	$(25, 12)$	6	5	$1, 2, 3, 4$ $3, 1, 4, 2$
6	$(42, 23)$	4	7	$1, 2, 3, 4, 6$ $3, 5, 2, 1, 4$
7	$(49, 30)$	4	7	$1, 2, 3, 4, 5, 6$ $3, 5, 2, 1, 6, 4$
8	$(72, 47)$	4	9	$1, 2, 3, 4, 5, 7, 8$ $3, 6, 2, 1, 8, 5, 4$
9	$(81, 56)$	4	9	$1, 2, 3, 4, 5, 6, 7, 8$ $3, 6, 2, 1, 8, 7, 5, 4$
10	$(110, 79)$	6	11	$1, 2, 3, 4, 5, 6, 8, 9, 10$ $3, 1, 7, 2, 10, 9, 4, 6, 5$
11	$(121, 90)$	4	11	$1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ $3, 1, 7, 2, 10, 9, 8, 4, 6, 5$
12	$(156, 119)$	6	13	$1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12$ $3, 1, 8, 2, 9, 12, 4, 11, 5, 7, 6,$
large distance codes				
4	$(92, 25)$	22	23	$1, 2, 4$ $5, 3, 12$
5	$(245, 100)$	22	49	$1, 3, 10, 14$ $40, 31, 33, 30$
6	$(414, 209)$	22	69	$3, 4, 21, 26, 67$ $34, 15, 64, 33, 44$
7	$(763, 438)$	22	109	$1, 3, 11, 15, 45, 93$ $101, 34, 18, 9, 1, 4$
8	$(1224, 767)$	22	153	$2, 10, 26, 57, 89, 4, 49$ $22, 19, 5, 23, 61, 90, 123$

Table 3.2: Degree matrices for QC $(J = 3, L)$-regular LDPC codes with girth $g = 6$.

L	(N, K)	$d_{\min}$	M	W'
		short codes		
4	$(36, 11)$	6	9	$1, 4, 6$ $5, 2, 3$
5	$(65, 28)$ $\big((75, 30)$ [EG10]$\big)$	10	13	$1, 3, 7, 11$ $10, 4, 5, 6$
6	$(108, 56)$ $\big((156, 78)$ [EG10]$\big)$	10	18	$2, 3, 5, 7, 9$ $4, 6, 13, 1, 16$
7	$(147, 86)$	10	21	$2, 3, 8, 15, 17, 20$ $4, 6, 7, 9, 12, 13$
8	$(200, 127)$	8	25	$1, 3, 4, 10, 14, 15, 19$ $5, 6, 11, 24, 2, 9, 12$
9	$(270, 182)$	8	30	$1, 3, 10, 16, 23, 25, 26, 28$ $2, 6, 5, 9, 8, 12, 14, 22$
10	$(350, 247)$	8	35	$2, 6, 7, 18, 19, 26, 29, 31, 34$ $4, 5, 3, 13, 10, 16, 12, 11, 23$
11	$(451, 330)$	8	41	$1, 4, 8, 20, 27, 28, 29, 33, 39, 40$ $5, 7, 6, 9, 10, 19, 13, 21, 14, 35$
12	$(564, 425)$	8	47	$3, 7, 8, 22, 24, 27, 29, 35, 40, 41, 43$ $6, 2, 4, 5, 14, 16, 1, 21, 28, 9, 34$
		large distance codes		
4	$(116, 31)$	24	29	$3, 14, 21$ $7, 1, 17$
5	$(225, 92)$	24	45	$1, 3, 10, 14$ $40, 31, 33, 30$
6	$(431, 218)$	24	72	$3, 4, 21, 26, 67$ $34, 15, 64, 33, 44$
7	$(777, 446)$	24	111	$3, 11, 15, 45, 93, 110$ $34, 18, 9, 1, 4, 101$
8	$(1280, 802)$	24	160	$2, 4, 10, 26, 49, 57, 89$ $22, 90, 19, 5, 123, 23, 61$
9	$(1386, 926)$	20	154	$6, 9, 26, 65, 79, 99, 124, 153$ $24, 16, 14, 1, 46, 62, 137, 84$

Table 3.3: Degree matrices for QC $(J = 3, L)$-regular LDPC codes with girth $g = 8$.

L	(N, K)	$d_{\min}$	M	W'
			short codes	
4	$(148, 39)$ $\big((144, 36)$ [EG10]$\big)$	14	37 $\big(39$ [WYD08]$\big)$	$1, 14, 17$ $11, 6, 2$
5	$(305, 124)$ $\big((550, 220)$ [EG10]$\big)$	24	61 $\big(61$ [TSF01]$\big)$	$2, 20, 54, 60$ $26, 16, 31, 48$
6	$(606, 305)$ $\big((780, 390)$ [EG10]$\big)$	24	101 $\big(103$ [WYD08]$\big)$	$2, 24, 25, 54, 85$ $21, 15, 11, 8, 59$
7	$(1113, 638)$	24	159 $\big(160$ [WYD08]$\big)$	$2, 14, 27, 67, 97, 130$ $21, 24, 1, 6, 75, 58$
8	$(1752, 1097)$	24	219 $\big(233$ [WYD08]$\big)$	$3, 14, 26, 63, 96, 128, 183$ $24, 6, 19, 46, 4, 77, 107$
9	$(2871, 1916)$	24	319 $\big(329$ [WYD08]$\big)$	$6, 9, 26, 65, 99, 153, 233, 278$ $24, 16, 14, 1, 62, 84, 200, 137$
10	$(4300, 2912)$	24	430 $\big(439$ [WYD08]$\big)$	$9, 11, 26, 67, 101, 161, \ldots$ $233, 302, 395$ $23, 5, 1, 54, 33, 96, \ldots$ $120, 104, 244$
11	$(6160, 4482)$	24	560 $\big(577$ [WYD08]$\big)$	$2, 11, 25, 62, 101, 162, 225, \ldots$ $268, 421, 492$ $24, 21, 5, 55, 6, 59, 178, \ldots$ $132, 204, 311$
12	$(8844, 6635)$	22	737 $\big(758$ [WYD08]$\big)$	$2, 22, 23, 63, 101, 147, 219, \ldots$ $322, 412, 569, 601$ $16, 9, 6, 58, 34, 91, 126, \ldots$ $155, 185, 298, 232$
			large distance codes	
4	$(176, 46)$	24	44	$1, 14, 17$ $11, 6, 2$

Table 3.4: Degree matrices for QC $(J = 3, L)$-regular LDPC codes with girth $g = 10$.

L	(N, K)	$d_{\min}$	M	W'
		short codes		
4	$(292, 75)$ $\big((444, 111)$ [EG10]$\big)$	24	73 $\big(97$ [O'S06]$\big)$	$2, 25, 33$ $18, 6, 5$
5	$(815, 328)$ $\big((1700, 680)$ [EG10]$\big)$	24	163 $\big(181$ [TSF01]$\big)$	$5, 33, 42, 117$ $36, 35, 25, 57$
6	$(1860, 932)$ $\big((4680, 2340)$ [EG10]$\big)$	24	310 $\big(393$ [ZW10]$\big)$	$1, 24, 38, 145, 246$ $16, 36, 5, 82, 110$
6	$(1836, 920)$	24	306 $\big(393$ [ZW10]$\big)$	$9, 36, 38, 154, 204$ $33, 1, 13, 54, 123$
7	$(3962, 2266)$	24	566 $\big(881$ [O'S06]$\big)$	$3, 10, 33, 147, 297, 442$ $31, 22, 4, 93, 133, 219$
8	$(6784, 4242)$	24	848 $\big(1493$ [O'S06]$\big)$	$4, 24, 31, 143, 303, 498, 652$ $32, 9, 6, 70, 130, 193, 222$
9	$(12384, 8258)$	24	1376 $\big(2087$ [O'S06]$\big)$	$4, 20, 32, 160, 284, \ldots$ $569, 794, 1133$ $30, 7, 1, 92, 169, \ldots$ $350, 437, 645$
10	$(21030, 14723)$	24	2103	$6, 13, 28, 150, 291, 565, \ldots$ $678, 1258, 1600$ $30, 16, 5, 64, 225, 207, \ldots$ $491, 838, 746$
11	$(34507, 25098)$	24	3137	$9, 11, 24, 150, 306, 508, \ldots$ $666, 1279, 1765, 1958$ $31, 28, 1, 83, 131, 160, \ldots$ $429, 550, 956, 1391$
12	$(56760, 42572)$	24	4730	$3, 15, 22, 140, 286, 537, \ldots$ $811, 1113, 1878, 2524, 3349$ $31, 26, 1, 66, 95, 210, 373, \ldots$ $729, 878, 1365, 1644$

Table 3.5: Degree matrices for QC $(J = 3, L)$-regular LDPC codes with girth $g = 12$.

In the following, different approaches for creating such base matrices shall be described, utilizing Steiner Triple Systems, short QC LDPC block codes obtain from all-one matrices, or (double) Hamming codes. Moreover, the extension from monomial to binomial entries is discussed shortly. Note that when searching for QC LDPC block codes with short block lengths, the shortest possible base matrix should be considered.

3.7.1 STEINER TRIPLE SYSTEMS BASED QC LDPC BLOCK CODES

A Steiner Triple System (STS) is a set $\mathcal{S}$ of triples such that every 2-subset occurs in exactly one triple of $\mathcal{S}$. In 1846, Kirkman [Kir47] showed that an STS of order n (STS(n)) exists if and only if n modulo 6 is equal to 1 or 3, where n denotes the total number of unique elements within all its triples.

In [JW01a], [JW01b], and [TAD04], a (shortened) STS of order $c-b$ with c different triples is used to construct a $(c-b) \times c$ base matrix $B_{\mathrm{STS}(c-b)}$ such that its nonzero positions within the ith column are determined by the ith triple. In particular note that such a base matrix does not contain an all-one 2×3 submatrix and thus satisfies the necessary condition imposed by Theorem 3.3. Applying the previously described algorithms to such a base matrix yields QC LPDC block codes with girth $g \geq 14$.

As before, a certain subset of edges of the voltage graphs can be labeled with the zero edge voltage, such that the number of possible voltage assignments is decreased. From a graph theoretical point of view, this corresponds to creating a spanning tree $\mathcal{S}$ and assigning the zero edge voltage to all its edges by adjusting the edge voltages of all adjacent edges not in $\mathcal{S}$.

The following algorithm deterministically creates such a spanning tree, by reordering the rows and columns of the base matrix such that the number of zero elements in its lower left corner is maximized. In such a base matrix, it is always possible to label the last nonzero entry in each column with degree zero. Moreover, in each of the remaining rows at the top of the base matrix, at least one nonzero entry can be labeled with degree zero[5]. Parity-check matrices fulfilling these criteria are for example given by (3.22) and (3.23).

Algorithm STS (Construction of a (J, L)-regular $(c-b) \times c$ base graph B obtained from an STS$(c-b)$)

1. Initialize a counter u with the number of nonzero entries per row, that is, $u = L$. Moreover, denote the current row and column by s and t, respectively, starting from the right-most entry in the last row, that is, $s = c - b - 1$ and $t = c - 1$.

[5]Hereinafter the first element in each of the remaining rows shall always be labeled with zero voltage.

2. Set the u entries in row s and column t, $t-1, \ldots, t-u+1$ to one, that is, $b_{ij} = 1$ with $i = s$ and $j = t, t-1, \ldots, t-u+1$.
3. Choose the remaining $J-1$ nonzero positions in each of those u columns such that the properties of an STS are satisfied. If possible, choose the positions b_{ij} to minimize i. That is, avoid using the lowest rows $s-1$, $s-2, \ldots$, if possible due to the restrictions imposed by the STS.
4. Finally, decrease t by u. If $t < 0$, then stop as all c columns have been used. Otherwise, set s to $s-1$, denote the number of nonzero elements this new row s by w, set $u = L - w$, and go to step **STS-2**.

By removing the last row and the last L columns of such a (J, L)-regular $(c-b) \times c$ base matrix B constructed using an STS$(c-b)$, we obtain a shortened $(c-b-1) \times (c-L)$ $(J, L-1)$-regular base matrix B'. Deleting additional columns and rows, yields base matrices of intermediate sizes, which are, however, irregular.

■ **Example 3.3:**
In the following, $(J = 3, L)$-regular base matrices of dimension 9×12 $(L = 4)$, dimension 13×26 $(L = 6)$, and dimension 25×100 $(L = 7)$ shall be constructed. Using the STS algorithm, the following STSs (base matrices) of order 9 (STS(9)), 13 (STS(13)) and 25 (STS(25)) are obtained:

$$\begin{aligned}
\mathbf{STS}(9) = \Big\{ & \{1,2,4\}, \quad \{0,3,5\}, \quad \{0,2,6\}, \quad \{1,5,6\}, \quad \{3,4,6\}, \\
& \{0,1,7\}, \quad \{4,5,7\}, \quad \{2,3,7\}, \quad \{0,4,8\}, \quad \{1,3,8\}, \\
& \{2,5,8\}, \quad \{6,7,8\} \Big\} \\
\mathbf{STS}(13) = \Big\{ & \{0,3,6\}, \quad \{0,2,7\}, \quad \{1,5,7\}, \quad \{3,4,7\}, \quad \{3,5,8\}, \\
& \{1,4,8\}, \quad \{2,6,8\}, \quad \{2,4,9\}, \quad \{5,6,9\}, \quad \{0,1,9\}, \\
& \{1,3,10\}, \quad \{0,4,10\}, \quad \{6,7,10\}, \quad \{2,5,10\}, \quad \{8,9,10\}, \\
& \{7,8,11\}, \quad \{4,6,11\}, \quad \{1,2,11\}, \quad \{0,5,11\}, \quad \{3,9,11\}, \\
& \{10,11,12\}, \quad \{7,9,12\}, \quad \{0,8,12\}, \quad \{1,6,12\}, \quad \{4,5,12\}, \\
& \{2,3,12\} \Big\} \\
\mathbf{STS}(25) = \Big\{ & \{4,5,10\}, \quad \{1,9,10\}, \quad \{7,8,11\}, \quad \{1,6,11\}, \quad \{2,3,12\}, \\
& \{0,9,12\}, \quad \{6,8,12\}, \quad \{8,9,13\}, \quad \{6,7,13\}, \quad \{0,5,13\}, \\
& \{2,10,13\}, \quad \{3,4,14\}, \quad \{1,12,14\}, \quad \{0,2,14\}, \quad \{7,9,14\}, \\
& \{5,11,14\}, \quad \{5,6,15\}, \quad \{3,10,15\}, \quad \{4,12,15\}, \quad \{1,7,15\}, \\
& \{0,8,15\}, \quad \{11,13,16\}, \quad \{5,7,16\}, \quad \{6,10,16\}, \quad \{2,8,16\},
\end{aligned}$$

$$
\begin{array}{lllll}
\{3,9,16\}, & \{0,4,16\}, & \{9,11,17\}, & \{12,13,17\}, & \{1,3,17\}, \\
\{4,7,17\}, & \{0,6,17\}, & \{2,5,17\}, & \{8,17,18\}, & \{3,11,18\}, \\
\{2,4,18\}, & \{13,15,18\}, & \{0,10,18\}, & \{1,16,18\}, & \{6,14,18\}, \\
\{9,18,19\}, & \{4,8,19\}, & \{14,15,19\}, & \{10,11,19\}, & \{0,3,19\}, \\
\{2,7,19\}, & \{12,16,19\}, & \{1,5,19\}, & \{17,19,20\}, & \{9,15,20\}, \\
\{10,12,20\}, & \{0,11,20\}, & \{5,8,20\}, & \{1,4,20\}, & \{13,14,20\}, \\
\{3,7,20\}, & \{2,6,20\}, & \{5,18,21\}, & \{4,6,21\}, & \{1,13,21\}, \\
\{16,17,21\}, & \{10,14,21\}, & \{2,9,21\}, & \{3,8,21\}, & \{11,15,21\}, \\
\{7,12,21\}, & \{19,21,22\}, & \{18,20,22\}, & \{0,7,22\}, & \{10,17,22\}, \\
\{3,5,22\}, & \{6,9,22\}, & \{2,15,22\}, & \{1,8,22\}, & \{11,12,22\}, \\
\{4,13,22\}, & \{14,16,22\}, & \{20,21,23\}, & \{0,1,23\}, & \{6,19,23\}, \\
\{15,16,23\}, & \{2,11,23\}, & \{7,18,23\}, & \{5,12,23\}, & \{14,17,23\}, \\
\{4,9,23\}, & \{8,10,23\}, & \{3,13,23\}, & \{0,21,24\}, & \{22,23,24\}, \\
\{1,2,24\}, & \{16,20,24\}, & \{7,10,24\}, & \{8,14,24\}, & \{13,19,24\}, \\
\{3,6,24\}, & \{12,18,24\}, & \{15,17,24\}, & \{5,9,24\}, & \{4,11,24\}\Big\}
\end{array}
$$

Note that the set of all c triples in an STS$(c-b)$, contains each number $0, 1, \ldots, c-b-1$ exactly L times, but is not necessarily uniquely determined.

The corresponding base matrices of dimension 9×12 STS(9), dimension 13×26 STS(13), and dimension 25×100 STS(25) are sparse matrices with nonzero elements in column i and row j, if and only if the ith triple contains the value j. The 9×12 $(3,4)$-regular base matrix constructed from the STS(9) is denoted by $B_{\mathrm{STS}(9)}$ and is for example given by

$$
B_{\mathrm{STS}(9)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array} \\
\begin{pmatrix}
0 & \mathbf{1} & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{1} & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
\mathbf{1} & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\mathbf{1} & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}
\end{pmatrix}
\end{array}
\tag{3.22}
$$

Entries which are part of the spanning tree $\mathcal{S}$, determined by the STS algorithm, are marked in bold and are hereinafter labeled with degree zero.

Removing its last row as well as its last $L = 4$ columns, yields the shortened 8×8 $(3,3)$-regular base matrix $B_{\text{S-STS}(9)}$

$$B_{\text{S-STS}(9)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 0 & \mathbf{1} & 1 & 0 & 0 & 1 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \mathbf{1} & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & 1 \\ \mathbf{1} & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & \mathbf{1} & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix} \end{array} \tag{3.23}$$

This corresponds to removing the four triples of the STS(9) containing the number (8) of the last row of (3.22). Shortening the 9×12 base matrix $B_{\text{STS}(9)}$ to obtain the shortened 8×8 base matrix $B_{\text{S-STS}(9)}$ is unpractical as its code rate becomes $R = 1 - 8/8 = 0$. However, shortening the 13×25 base matrix $B_{\text{STS}(13)}$ in the same way, yields the 12×20 base matrix $B_{\text{S-STS}(13)}$ with feasible code rate $R = 8/20$.

In Table 3.6, newly found QC $(J = 3, L)$-regular LDPC block codes with girth $g = 14$, 16, and 18 constructed from STSs are presented. While the number of nonzero elements in each column is fixed to $J = 3$, the number of nonzero elements in each row L depends on the underlying STS and is given in the first column. The second column specifies the obtained girth g, while in the next columns the dimensions of the block code (N, K) after tailbiting to length M are presented. The fifth column contains the STS used to construct the underlying base graph. Since the corresponding degree matrices W are too large, those matrices are omitted, but are available in [BHJ$^+$12] and [BHJ$^+$].

3.7.2 ITERATIVE QC LDPC BLOCK CODES

Besides using base matrices constructed from STSs, previously obtained QC $(J = 3, L)$-regular LDPC block codes of smaller block size and smaller girth can be reused as base matrices for larger QC LDPC block codes.

In the following, QC $(J = 3, L)$-regular LDPC block codes with girth $g = 8$ and binary parity-check matrices of size 27×36, 39×65, and 54×108 are used as base matrices (cf. Table 3.3). (Re-)applying the previously described algorithms yields QC $(J = 3, L)$-regular LDPC block codes with girth $g = 20$, 22, and 24, respectively, as presented in Table 3.7. As before, the first column denotes L, the number of nonzero elements in each column; the obtained girth g and the dimensions of the block code (N, K) after tailbiting to length M are given in the following columns. Since the corresponding degree matrices are too large, they are omitted in Table 3.7, but are available at [BHJ$^+$].

L	g	(N, K)	M	Base graph
4	14	$(1812, 453)$ $\big((2208, 732)$ [EG10]$\big)$	151 $\big(184$ [EG10]$\big)$	(9×12), STS(9)
5	14	$(9720, 3888)$ $\big((11525, 4612)$ [EG10]$\big)$	486	(12×20), S-STS(13)
6	14	$(29978, 14989)$ $\big((37154, 18577)$ [EG10]$\big)$	1153 $\big(1429$ [EG10]$\big)$	(13×26), STS(13)
12	14	$(80000000, 60000000)$	800000	(25×100), STS(25)
4	16	$(7980, 1995)$	665	(9×12), STS(9)
5	16	$(51240, 20496)$ $\big((62500, 25000)$ [EG10]$\big)$	2562	(12×20), S-STS(13)
6	16	$(227032, 113516)$ $\big((229476, 114738)$ [EG10]$\big)$	8732 $\big(8826$ [EG10]$\big)$	(13×26), STS(13)
4	18	$(32676, 8169)$	2723 $\big(2855$ [EG10]$\big)$	(9×12), STS(9)
5	18	$(271760, 108704)$ $\big((371100, 148440)$ [EG10]$\big)$	13588	(12×20), S-STS(13)

Table 3.6: Degree matrices for QC $(J = 3, L)$-regular LDPC codes with girth between $g = 14$ and $g = 18$.

Note that these codes are (probably) not practical due to their long block length. However, the table illustrates that by interpreting QC (J, L)-regular LDPC block codes as new base matrices and (re-)applying our algorithms it is possible to find QC (J, L)-regular LDPC block codes of »any« girth g.

3.7.3 DOUBLE-HAMMING BASED QC LDPC BLOCK CODES

Next, a new class of QC (J, L)-regular LDPC block codes constructed from binary base matrices of size $2J \times 2L$ shall be considered. This class of QC LDPC block codes will be referred to as *Double-Hamming based QC LDPC block codes* since its base matrix B_{DH} is constructed by using the parity-check matrix of a Hamming code twice, that is,

$$B_{\mathrm{DH}} = \begin{pmatrix} I_J & P & \mathbf{1} & \mathbf{0} & W_1 \\ P_{\mathrm{p}} & I_J & \mathbf{0} & \mathbf{1} & W_2 \end{pmatrix} \tag{3.24}$$

L	g	(N, K)	M	Base graph (Table 3.3)
4	20	$(1296000, 324002)$	36000	(27×36), $g = 8$
5	20	$(31200000, 12480002)$	480000	(39×65), $g = 8$
6	20	$(518400000, 259200002)$	4800000	(54×108), $g = 8$
4	22	$(7200000, 1800002)$	200000	(27×36), $g = 8$ [EG10]
5	22	$(325000000, 130000002)$	5000000	(39×65), $g = 8$
4	24	$(39600000, 9900002)$	1100000	(27×36), $g = 8$

Table 3.7: Properties of QC $(J = 3, L)$-regular LDPC codes with girth $g \geq 20$.

where I_J is the $J \times J$ identity matrix, and $\mathbf{0}$, $\mathbf{1}$ denote the all-zero and all-one column vectors, respectively. The submatrix $(P \ \mathbf{1})$ corresponds to the parity part of the parity-check matrix of the $(2^J - 1, J)$ Hamming code (cf. Example 1.3), while P_{p} denotes a permutation of P.

Depending on the desired code rate of the QC (J, L)-regular LDPC block code, the dimensions of the matrices W_1 and W_2 are adjusted correspondingly. These matrices can be chosen arbitrarily, but with the restriction that the number of nonzero elements in each column and in each row of the base matrix B has to be equal to J and L, respectively, avoiding identical columns in B.

■ **Example 3.4:**
The base matrix B_{DH} for an $R = 2/8$ $(3, 4)$-regular QC LDPC code is given by

$$B_{\mathrm{DH}} = \begin{array}{c} \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \left(\begin{array}{ccc|ccc|c|c} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right) \end{array} \tag{3.25}$$

where the matrices W_1 and W_2 are not present.

Since there always exist at least two columns within the base matrix B_{DH} which coincide in at least two positions, the girth of the corresponding base graph is $g_{\mathrm{B}} = 4$. Hence, it follows from Theorem 3.2 that the achievable girth of such a code construction is $g \geq 3g_{\mathrm{B}} = 12$. However, since the second generalized Hamming distance d_2 of the $R = 10/24$ convolutional code, whose parity-check matrix coincides with the incidence matrix of the base graph specified by (3.25), is equal to 7, it follows from the same theorem that a Double-Hamming based QC LDPC block code with $g \geq 2d_2 = 14$ exists.

■ **Example 3.4 (Cont'd):**
Applying the VA-B algorithm to the base matrix given in (3.25) yields a $(2112, 528)$ QC LDPC block code with girth $g = 14$ by labeling the edges of its base graph with $(0,0,0,0)$, $(0,13,0,181)$, $(0,87,66,101)$, $(7,260,245,0)$, $(0,154,33,6)$, $(107,0,130,85)$, where the ith entry within the jth 4-tuple corresponds to the monomial degree of the ith nonzero entry in the jth row of the base matrix. Hitherto, the shortest published QC LDPC code of rate $R = 1/4$ with $g = 14$ had length 2208 and was reported in 2010 by Esmaeili and Gholami [EG10].

Additionally, this construction can be generalized to $J \geq 3$ in a straight forward manner as seen in the following example:

■ **Example 3.5:**
Consider, for example, the rate $R = 8/16$ base matrix of a $(4,8)$-regular QC LDPC code, given by

$$B_{\mathrm{DH}} = \begin{pmatrix} I_4 & P_1 & P_2 & \mathbf{1} & \mathbf{0} \\ P_{2\mathrm{p}} & P_{1\mathrm{p}} & I_4 & \mathbf{0} & \mathbf{1} \end{pmatrix} \tag{3.26}$$

where the parity part P of the Hamming code in (3.24) has been split into two submatrices P_1 and P_2 for notational convenience. The submatrices of the parity part of the corresponding Hamming code are $P_2 = P_{2\mathrm{p}} = I_4^{\mathrm{c}}$, where I_4^{c} denotes the complement of the identity matrix I_4, as well as,

$$P_1 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{3.27}$$

and

$$P_{1\mathrm{p}} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.28}$$

$(N, K, d_{\min})$	$d_{\text{free}}(\widehat{d}_{\text{free}})$	g	M	Edge Voltages
Rate $R = 1/4$				
$(168, 42, 30)$	54 (≤ 66)	8	21	$7,3,0,5;1,0,4,10;0,4,7,9;$ $7,6,0,3;0,0,7,6;7,7,0,10$
$(160, 40, 32)$	76 (≤ 102)	10	20	$0,6,15,11;6,9,0,0;6,0,2,14;$ $19,0,12,4;13,4,0,4;12,0,5,0$
Rate $R = 2/5$				
$(380, 144, 26)$	≥ 40 (≤ 72)	8	38	$0,5,0,19,9;3,0,0,11,0;$ $3,12,9,0,4;0,0,12,9,14;$ $6,5,13,0,2;5,0,5,0,0$
$(370, 148, \geq 30)$ $(d_{\min} \leq 36)$	≥ 38 (≤ 86)	10	37	$0,22,28,6,24;0,7,0,0,11;$ $0,8,25,0,0;19,6,0,15,0;$ $14,8,0,31,21;0,11,5,27,6$
Rate $R = 1/2$				
$(1080, 540, \geq 28)$	(≤ 90)	10	90	$40,47,17,77,36,10;19,74,43,$ $24,86,31;86,56,3,83,52,56;$ $26,38,0,22,81,25;77,47,13,$ $6,6,70;76,0,56,11,20,57$

Table 3.8: Parameters of new Double-Hamming based rate $R = 1/4$ $(J = 3, L = 4)$-regular, $R = 2/5$ $(J = 3, L = 5)$-regular, and $R = 1/2$ $(J = 3, L = 6)$-regular QC LDPC codes.

Using the base matrix (3.26), applying the VA-B algorithm, and tailbiting the obtain convolutional parity-check matrix to length $M = 1168$, yields a $(18688, 9344)$ QC $(J = 4, L = 8)$-regular LDPC block code with girth $g = 10$. The corresponding voltage assignment is omitted, but is available at [BHJ$^+$].

Additional examples of newly found Double-Hamming based QC LDPC block with base matrix (3.24) are presented in Table 3.8. The first column specifies its dimension K and block length N, as well as its minimum distance $d_{\min}$, while the free distance d_{free} of its parent convolutional code is, if possible, given together with the corresponding upper bound $\widehat{d}_{\text{free}}$ (3.20) in the second column.

Tailbiting to length M according to the fourth column yields a QC LDPC block code with girth g, as specified in the third column. The voltage assignment for the base graph B, obtained by applying the VA-B algorithm, are given in the last column, where different rows are separated by a semicolon,

while zero entries in the base matrix are omitted. Hence, the jth edge voltage in kth block corresponds to the monomial degree of the jth nonzero entry in the kth row of its base matrix B_{DH}.

For comparison, we would like to mention the hitherto shortest published $R = 2/5$ $(3, 5)$-regular QC LDPC code with $g = 10$ was found by Esmaeili and Gholami in 2010 [EG10] and has block length $N = 550$.

3.7.4 BINOMIAL QC LDPC BLOCK CODES

Generalizing the previously discussed construction to allow parallel edges between two vertices is straight forward. Consider the base matrix $B = B' + B''$, where B' and B'' are two »ordinary« base matrices of the same size. Hence, if two vertices are connected by an edge in both base matrices B' and B'', they will be connected by two parallel edges in B. In particular, such a base matrix B contains the integer entries $\{0, 1, 2\}$, where 2 denotes a binomial entry, corresponding to a parallel edge in the underlying base graph.

In [SV11], Smarandache et al. derive different upper bounds on the girth and on the minimum distance of such binomial QC LDPC block codes. For rate $R = 1/4$ QC LDPC block codes it is shown that, depending on the chosen edge voltages, their minimum distance is upper-bounded either by 32 with girth ≤ 8 or by 30 and 28 with girth ≤ 10. Moreover, for codes of rate $R = 2/5$, the minimum distance is less than or equal to 28 with girth ≤ 8. In particular, a $(184, 47)$ QC LDPC block code with minimum distance $d_{\min} = 32$ and girth $g = 8$ is presented, achieving the derived upper bounds.

Applying our previously presented VA-B algorithm to the base matrix

$$B_{\mathrm{Bin}} = \begin{pmatrix} 2 & 0 & 1 & 1 \\ 1 & 1 & 2 & 0 \\ 0 & 2 & 0 & 2 \end{pmatrix} \tag{3.29}$$

yields QC $(3, 4)$-regular LDPC block codes with minimum distance up to 30 as well as very short block codes of length $N = 96$ and minimum distance $d_{\min} = 24$ as summarized in Table 3.9.

The code length N, dimension K, and minimum distance $d_{\min}$ of the obtained QC LDPC block code are given in the first column of Table 3.9, followed by the tailbiting length M, the free distance d_{free} of the parent LDPC convolutional code, and the achievable girth g. Additionally, the obtained voltage assignment for the base graph B_{Bin} is specified in the last column, where the edge voltages for each row are separated by a semicolon. Note that a tuple (a, b) corresponds to a binomial entry $(D^a + D^b)$ while a single value (a) refers to a monomial entry (D^a). As before, zero entries of the base matrix are omitted, that is, the jth edge voltage in the kth block corresponds to the degree(s) of the jth nonzero entry in the kth row of the base matrix B_{Bin}.

$(N, K, d_{\min})$	M	d_{free}	g	Edge Voltages
$(96, 25, 24)$	24	30	8	$(4,0),(13),(4)$; $(4),(3),(0,1)$; $(10,0),(3,0)$
$(112, 29, 26)$	28	30	8	
$(124, 32, 28)$	31	30	8	
$(144, 37, 30)$	36	30	8	
$(108, 28, 24)$	27	28	8	$(0,2),(13),(2)$; $(10),(3),(0,1)$; $(13,0),(0,3)$
$(116, 30, 26)$	29	30	8	
$(136, 35, 28)$	34	30	8	
$(152, 39, 30)$	38	30	8	

Table 3.9: Parameters for new binomial rate $R = 1/4$ $(J = 3,\ L = 4)$-regular QC LDPC codes.

3.8 CASE STUDY: IEEE 802.16 WIMAX

Due to their low decoding complexity when decoded with the BP algorithm (cf. Section 2.6) and their good BER performance close to the theoretical limit, LDPC block codes are a suitable choice for modern communication standards [IEE05] [Eur08] [Eur09]. For example, within the IEEE 802.16 WiMAX standard [IEE05], QC irregular LDPC block codes with code rate from $1/2$ up to $5/6$ and block length between 576 and 2304 are defined. To the best of our knowledge, nobody reported QC LDPC block codes of such rates with better BER performance, especially for block lengths as short as $N \approx 576$.

Using the algorithms described in Section 3.4, we search for QC $(J = 3, L)$-regular LDPC block codes of same rate R and similar block length, that is, $N \approx 576$, but larger girth g, based on all-one base matrices or STSs (only for $R = 1/2$). In Table 3.10, newly found QC (J, L)-regular LDPC block codes with rate $R = 1/2$ and $R = 2/3$ are presented, together with the parameters of the corresponding QC irregular LDPC block code of the same rate as defined within the IEEE 802.16 WiMAX standard. The block length of all codes is $N = 576$, except for the QC LDPC block code constructed from STSs, whose block length is slightly shorter with $N = 572$. Additionally, a QC LDPC block code constructed from an all-one base matrix with randomly chosen edge voltages is included. For each code, its girth g, tailbiting length M,

Base Matrix	Class	g	M	Edge Voltages
$R = 1/2$				
12×24 IEEE 802.16	irregular	4	24	see [IEE05]
13×26 STS(13)*	$(3,6)$-reg.	8	22	see [BHJ+]
3×6 All-one	$(3,6)$-reg.	8	96	$4,15,24,69,86$ $22,58,79,23,25$
3×6 All-one	$(3,6)$-reg. (random)	4	96	$13,46,52,92,93$ $76,91,26,91,15$
$R = 2/3$				
8×24 IEEE 802.16 (A)	irregular	6	24	see [IEE05]
8×24 IEEE 802.16 (B)	irregular	4	24	see [IEE05]
3×9 All-one	$(3,9)$-reg.	8	64	$6,7,13,26,36,42,44,58$ $37,12,16,51,14,34,35,43$
3×9 All-one	$(3,9)$-reg. (random)	4	64	$2,2,20,24,31,41,50,52$ $17,60,44,28,11,28,48,6$

Table 3.10: Parameters of rate $R = 1/2$ and $R = 2/3$ QC LDPC block codes with block length $N = 576$ ($N^* = 572$).

and, if possible, its edge voltages are specified. The QC LDPC block codes specified within the IEEE 802.16 WiMAX standard are irregular, and hence have a higher degree of freedom, possibly yielding a better BER performance.

Using the BP decoding algorithm (Section 2.6), the BER performance for those LDPC block codes is simulated and illustrated in Figure 3.6 and Figure 3.7 for rate $R = 1/2$ and rate $R = 2/3$, respectively. Similarly, newly found QC (J, L)-regular rate $R = 3/4$ and $R = 5/6$ LDPC block codes with block length $N = 572$ are given in Table 3.11 together with the corresponding QC irregular LDPC block codes defined within the IEEE 802.16 WiMAX standard, while their BER performance is compared in Figure 3.8 and Figure 3.9 for rate $R = 3/4$ and rate $R = 5/6$, respectively. All simulations results are, except if stated otherwise, based on at least 100 different block error events.

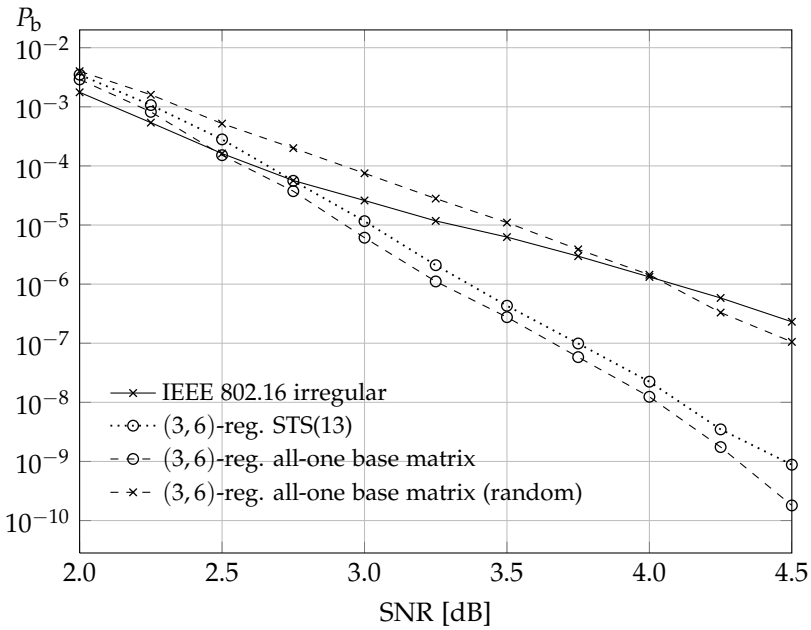


Figure 3.6: BER performance for $R = 1/2$ LDPC block codes.

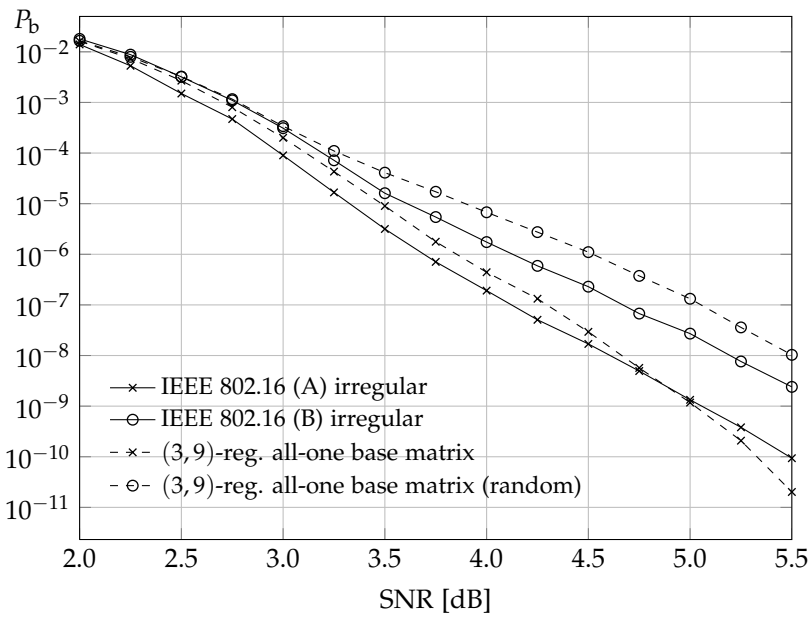


Figure 3.7: BER performance for $R = 2/3$ LDPC block codes, where the simulation results for 5.25 dB and 5.5 dB are based only on at least 25 and 8 block error events, respectively.

Base Matrix	Class	g	M	Edge Voltages
$R = 3/4$				
6×24 IEEE 802.16 (A)	irregular	4	24	see [IEE05]
6×24 IEEE 802.16 (B)	irregular	4	24	see [IEE05]
3×12 All-one	$(3, 12)$-reg.	6	64	$1, 7, 8, 23, 26, 32, 37, \ldots$ $43, 44, 46, 47$ $4, 38, 42, 36, 10, 17, 44, \ldots$ $3, 41, 25, 19$
3×12 All-one	$(3, 12)$-reg. (random)	4	64	$7, 9, 13, 17, 32, 35, 37, \ldots$ $41, 45, 48, 57$ $31, 16, 21, 43, 44, 8, 14, \ldots$ $10, 48, 16, 13$
$R = 5/6$				
4×24 IEEE 802.16	irregular	4	24	see [IEE05]
3×18 All-one	$(3, 18)$-reg.	6	64	$1, 5, 6, 7, 8, 12, 13, 14, 16, \ldots$ $17, 19, 21, 23, 24, 25, 27, 30$ $31, 20, 2, 4, 24, 21, 23, 19, 9, \ldots$ $7, 11, 10, 27, 15, 12, 3, 18$
3×18 All-one	$(3, 18)$-reg. (random)	4	64	$0, 0, 2, 3, 4, 5, 5, 7, 8, 9, \ldots$ $14, 16, 17, 18, 22, 26, 29$ $10, 24, 14, 30, 26, 25, 19, 29, \ldots$ $20, 16, 2, 24, 31, 15, 23, 27, 4$

Table 3.11: Parameters of rate $R = 3/4$ and $R = 5/6$ QC LDPC block codes with block length $N = 576$.

Somewhat surprisingly, our newly found QC regular LDPC block codes yield a better BER performance for rate $R = 1/2$, $3/4$, and $5/6$; typically around $0.5 - 1\,\text{dB}$ for small and medium SNR values. For the code rate $R = 2/3$ we were only able to improve the BER performance for one of the two QC irregular LDPC block codes defined in the IEEE 802.16 WiMAX standard, while the second one still yielded the overall best BER performance for small SNRs. Compared to the QC regular LDPC block codes with random edge voltages, the advantages by choosing suitable edge voltages is eminent. Moreover, applying our algorithms to irregular base matrices, should potentially yield QC LDPC block codes with further improved BER performance.

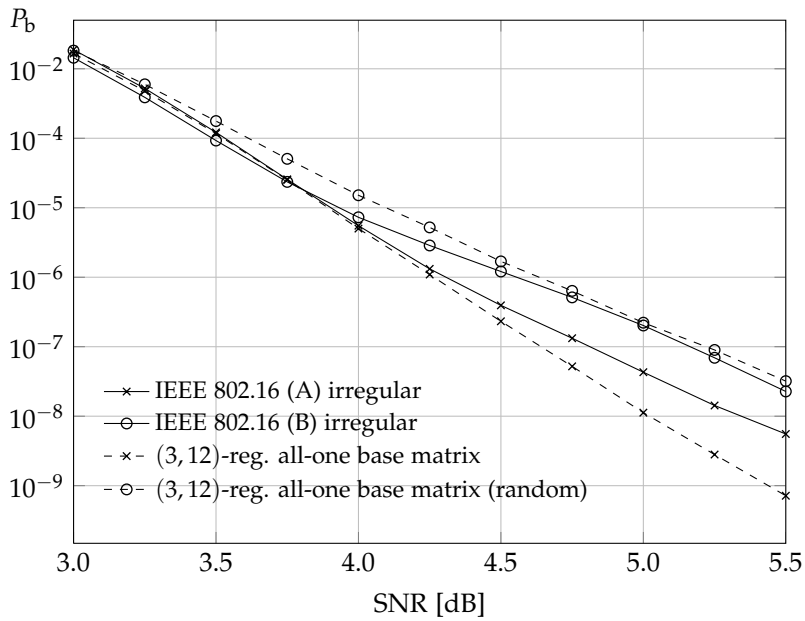


Figure 3.8: BER performance for $R = 3/4$ LDPC block codes.

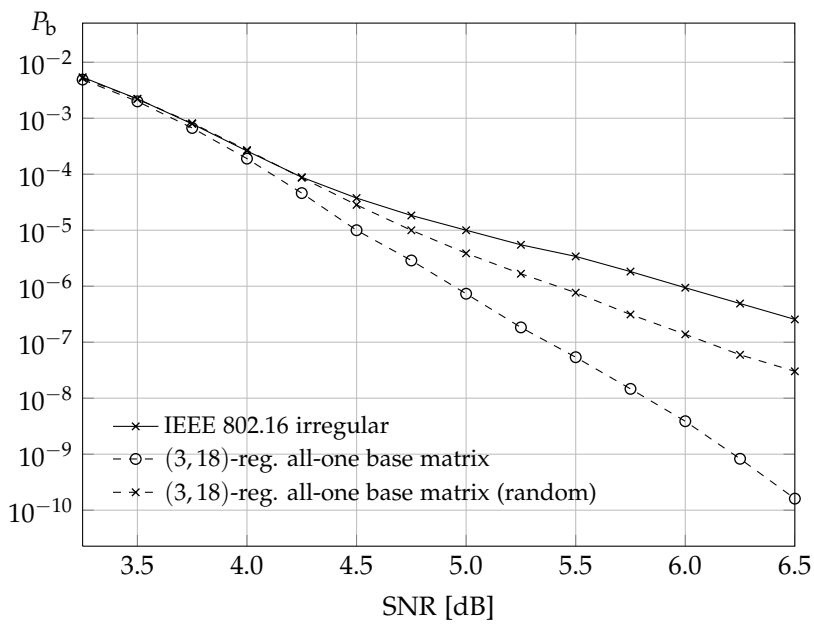


Figure 3.9: BER performance for $R = 5/6$ LDPC block codes.

4

Woven Graph Codes

Woven graph codes are generalizations of (J, L)-regular graph-based codes with either constituent block or convolutional codes [BKJZ07] [BKJZ10]. Their distinguishing feature is that the codeword length of the constituent code is a multiple of the constraint node degree L, that is, their length is equal to LM, where M is an integer. In particular, when M tends to infinity we obtain woven graph codes with constituent convolutional codes. While, for example, serial concatenated convolutional codes are obtained by combining their generator matrices, woven graph codes are obtained by combining their corresponding parity-check matrices.

Within the ensemble of woven graph codes with constituent block codes, based on bipartite graphs, the existence of codes satisfying the Varshamov-Gilbert (VG) bound has been proven [BKJZ10]. Due to the simple structure of woven graph codes, such codes can be analyzed with low computational complexity while their minimum distance is rather close to the minimum distance of the best known linear block codes of same length and dimension.

Graph-based block codes with constituent codes will be introduced in Section 4.1, while Section 4.2 focuses on their generalization to woven graph codes with either constituent block codes or constituent convolutional codes. An asymptotic bound on the free distance of woven convolutional graph codes, namely, the Costello lower bound, is proven in Section 4.3. Examples of promising woven convolutional graph codes are given in Section 4.4, including a rate $R = 5/20$ woven convolutional graph code with overall constraint length $\nu = 67$ and free distance $d_{\text{free}} = 120$. This chapter is concluded in Section 4.5 with a short BER comparison of a woven convolutional graph code with overall constraint length $\nu = 26$ and free distance d_{free} with (near-) optimal convolutional codes of same complexity or free distance.

4.1 GRAPH-BASED BLOCK CODES WITH CONSTITUENT CODES

A graph-based (J, L)-regular block code with a rate $\widehat{R} = \hat{b}/\hat{c}$ constituent block code $\widehat{\mathcal{B}}$ is based on a bipartite graph $\mathcal{G}$ whose set of constraint nodes $\mathcal{V}_1$ is s-partite with equally sized partitions. Using the concept of biadjacency matrices, the corresponding parity-check matrix for such a graph $\mathcal{G}$ can be expressed as

$$H = \begin{pmatrix} \widetilde{H}_0 \\ \widetilde{H}_1 \\ \vdots \\ \widetilde{H}_{s-1} \end{pmatrix} \tag{4.1}$$

where each of the s parity-check submatrices $\widetilde{H}_i$ of size $n(\hat{c} - \hat{b}) \times n\hat{c}$, $i = 0, 1, \ldots, s-1$, corresponds to one disjoint set of $n(\hat{c} - \hat{b})$ constraint nodes $\mathcal{V}_1^{(i)}$. By reordering its rows and columns, it is possible, without loss of generality, to represent the first parity-check submatrix $\widetilde{H}_0$ as an $n \times n$ block structure

$$\widetilde{H}_0 = \begin{pmatrix} \widehat{H} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \widehat{H} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \widehat{H} \end{pmatrix} \tag{4.2}$$

where $\widehat{H}$ is a $(\hat{c} - \hat{b}) \times \hat{c}$ parity-check matrix of the constituent block code $\widehat{\mathcal{B}}$. The remaining $s-1$ parity-check matrices $\widetilde{H}_k$ are column permutations of $\widetilde{H}_0$, determined by the underlying graph $\mathcal{G}$. In particular, note that for this construction we obtain $J = s(\hat{c} - \hat{b})$ and $L = \hat{c}$.

Assigning different constituent block codes $\widehat{\mathcal{B}}$ of rate $\widehat{R} = \hat{b}/\hat{c}$ to the same bipartite graph $\mathcal{G}$, whose set of constraint node is s-partite with equally sized partitions, yields a set of related graph-based block codes. Since the total number of parity-checks of such a block code is at most $sn(\hat{c} - \hat{b})$, its code rate R follows as

$$R \geq \frac{n\hat{c} - sn(\hat{c} - \hat{b})}{n\hat{c}} = 1 - s(1 - \widehat{R}) \tag{4.3}$$

with equality if and only if all parity-checks are linearly independent.

In particular, QC (J, L)-regular LDPC block codes (cf. Chapter 3) are graph-based codes with constituent single parity-check block codes of rate $\widehat{R} = (L-1)/L$, constructed from a bipartite graph $\mathcal{G}$ with a J-partite set of constraint nodes. Following Chapter 3, it is convenient to represent such bipartite graph-based codes by their parent convolutional parity-check matrix. Although such a representation is not necessary for the following proofs and theorems, all examples of bipartite graphs will be given in the form of their parent convolutional parity-check matrices.

■ **Example 4.1:**
Consider the rate $R = 1/4$ convolutional code C with free distance $d_{\text{free}} = 8$ determined by its monomial parity-check matrix

$$H(D) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & D & D^2 & D^3 \\ 1 & D^3 & D & D^2 \end{pmatrix} \tag{4.4}$$

which is constructed by combining the 3×4 all-one base matrix B with the corresponding weight matrix W, where

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{4.5}$$

and

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix} \tag{4.6}$$

respectively. Tailbiting $H(D)$ to length $M = 4$ yields the 12×16 tailbiting parity-check matrix of a QC $(J = 3, L = 4)$-regular LDPC block code

$$H^{(\text{tb})} = \begin{array}{c} \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \\ c_9 \\ c_{10} \\ c_{11} \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \end{array} \\ \left(\begin{array}{cccc|cccc|cccc|cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}\right) \end{array} \tag{4.7}$$

Interpreting this parity-check matrix as a biadjacency matrix yields the corresponding Tanner graph $\mathcal{G}$ with 16 symbol nodes, 12 constraint nodes, and girth $g = 4$ as illustrated in Figure 4.1. Constructing such a bipartite graph $\mathcal{G}$ based on a monomial parity-check matrix of a rate $R = b/c$ parent convolutional code, ensures that its set of constraint nodes is at least $(c - b)$-partite. Hence, in our example, the set of constraint nodes $\mathcal{V}_1$ is 3-partite, that is, it consists of the 3 disjoint subsets $\mathcal{V}_1^{(i)}$, $i = 0, 1, 2$.

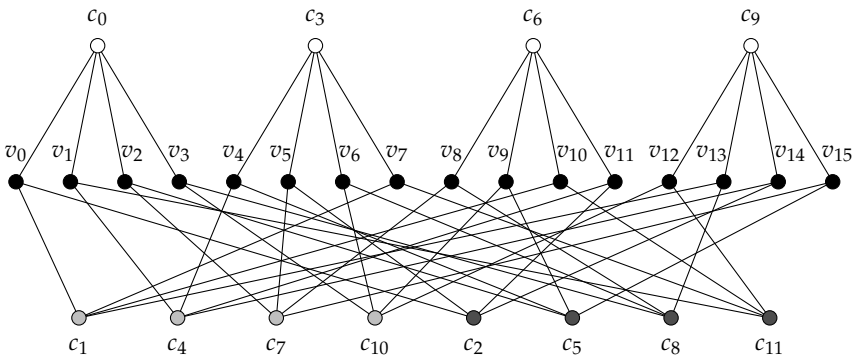


Figure 4.1: Tanner graph with 16 symbol nodes and 12 constraint nodes. The three disjoint sets of constraint nodes are represented by white, light gray, and dark gray vertices.

Reordering the rows of (4.7) as c_0, c_3, c_6, c_9, c_1, c_4, $c_7, \ldots$, yields an equivalent parity-check matrix

$$H^{(\mathrm{tb})\prime} = \begin{array}{c} \\ c_0 \\ c_3 \\ c_6 \\ c_9 \\ c_1 \\ c_4 \\ c_7 \\ c_{10} \\ c_2 \\ c_5 \\ c_8 \\ c_{11} \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \end{array} \\ \left(\begin{array}{cccc|cccc|cccc|cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}\right) \end{array} \tag{4.8}$$

where the three submatrices $\widetilde{H}_k$, $k = 0, 1, 2$, are separated by horizontal lines. Its constituent block code $\widehat{\mathcal{B}}$ of rate $\widehat{R} = (L-1)/L$ is a single parity-check block code with block length L, determined by the parity-check matrix

$$\widehat{H} = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \tag{4.9}$$

Since the rows in (4.7) and (4.8) are linearly dependent, two parity-checks have to be removed, yielding a $(16, 6)$ linear graph-based block code. Clearly, its rate satisfies inequality (4.3) since

$$R = 6/16 \geq 1 - s\,(1 - \widehat{R}\,) = 1 - 3\,(1 - 3/4\,) = 1/4 \tag{4.10}$$

4.2 WOVEN GRAPH CODES

Woven graph codes are generalizations of graph-based codes with either constituent block or convolutional codes. Consider a binary $(M\hat{c}, M\hat{b})$ linear block code, determined by the parity-check matrix

$$\widehat{H} = \begin{pmatrix} \widehat{H}_{00} & \widehat{H}_{01} & \cdots & \widehat{H}_{0(\hat{c}-1)} \\ \widehat{H}_{10} & \widehat{H}_{11} & \cdots & \widehat{H}_{1(\hat{c}-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{H}_{(\hat{c}-\hat{b}-1)0} & \widehat{H}_{(\hat{c}-\hat{b}-1)1} & \cdots & \widehat{H}_{(\hat{c}-\hat{b}-1)(\hat{c}-1)} \end{pmatrix}$$

where $\widehat{H}_{ij} \in \mathcal{B}_{M\times M}$, $i = 0, 1, \ldots, \hat{c} - \hat{b} - 1$, $j = 0, 1, \ldots, \hat{c} - 1$, is a size $M \times M$ binary submatrix and $\mathcal{B}_{M\times M}$ denotes the set of all possible binary matrices of size $M \times M$. The corresponding graph-based code of length $M\hat{c}n$ with a constituent block code, determined by its $M(\hat{c} - \hat{b}) \times M\hat{c}$ parity-check matrix $\widehat{H}$, is called a *woven graph code* with a constituent block code. In other words, we generalize graph-based codes to woven graph codes by assigning a length M subblock of a length $M\hat{c}$ codeword to each node within a group of $\hat{c}$ consecutive symbol nodes. Similarly, a block of M parity-check equations is assigned to each constraint node. As a special case, we obtain graph-based block codes for $M = 1$.

Woven graph codes with constituent convolutional codes, hereinafter referred to as *woven convolutional graph codes*, can be considered as a straightforward generalization of woven graph codes with constituent block codes.

Let $\widehat{H}(D)$ denote the minimal-basic $(\hat{c} - \hat{b}) \times \hat{c}$ parity-check matrix of a rate $\widehat{R} = \hat{b}/\hat{c}$ convolutional code

$$\widehat{H}(D) = \begin{pmatrix} \hat{h}_{00}(D) & \hat{h}_{01}(D) & \cdots & \hat{h}_{0(\hat{c}-1)}(D) \\ \hat{h}_{10}(D) & \hat{h}_{11}(D) & \cdots & \hat{h}_{1(\hat{c}-1)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{h}_{(\hat{c}-\hat{b}-1)0}(D) & \hat{h}_{(\hat{c}-\hat{b}-1)1}(D) & \cdots & \hat{h}_{(\hat{c}-\hat{b}-1)(\hat{c}-1)}(D) \end{pmatrix} \tag{4.11}$$

where

$$\hat{h}_{ij}(D) = \hat{h}_{ij}^{(0)} + \hat{h}_{ij}^{(1)} D + \hat{h}_{ij}^{(2)} D^2 + \cdots \tag{4.12}$$

with $i = 0, 1, \ldots, \hat{c} - \hat{b} - 1$, and $j = 0, 1, \ldots, \hat{c} - 1$, is a binary polynomial. Denote by $\mathbb{F}_2((D))$ the field of binary Laurent series and regard a rate $\widehat{R} = \hat{b}/\hat{c}$ convolutional code as a block code of same rate over the field of binary Laurent series. Then, the corresponding codewords are elements of $\mathbb{F}_2^{\hat{c}}((D))$, which is a $\hat{c}$-dimensional vector space over the field of binary Laurent series.

In other words, we assign a length $\hat{c}$ subblock to a group of $\hat{c}$ consecutive symbol nodes, which corresponds to a time-cut of a length $M\hat{c}$ code sequence, where M tends towards infinity.

Exploiting the above definitions in connection with the Tanner graph representation in Figure 4.1, the n constituent convolutional codes at the top can be regarded as a warp with $n\hat{c}$ threads. Each of the n constituent convolutional codes at the bottom are tacked on $\hat{c}$ of the threads in the warp such that each thread of the warp is tacked on exactly once. Thus, this construction is a special case of a woven code [HJZ02].

■ **Example 4.2:**
The parity-check matrix $H_{\text{wg}}(D)$ of a woven convolutional graph code, based on the Tanner graph with parity-check matrix (4.8) from Example 4.1, is for example given by

$$H_{\text{wg}}(D) = \begin{array}{c} \\ c_0 \\ c_3 \\ c_6 \\ c_9 \\ c_1 \\ c_4 \\ c_7 \\ c_{10} \\ c_2 \\ c_5 \\ c_8 \\ c_{11} \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \end{array} \\ \left(\begin{array}{cccc|cccc|cccc|cccc} \hat{h}_0 & \hat{h}_1 & \hat{h}_2 & \hat{h}_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \hat{h}_0 & \hat{h}_1 & \hat{h}_2 & \hat{h}_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \hat{h}_0 & \hat{h}_1 & \hat{h}_2 & \hat{h}_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \hat{h}_0 & \hat{h}_1 & \hat{h}_2 & \hat{h}_3 \\ \hline \hat{t}_0 & 0 & 0 & 0 & 0 & 0 & 0 & \hat{t}_3 & 0 & 0 & \hat{t}_2 & 0 & 0 & \hat{t}_1 & 0 & 0 \\ 0 & \hat{t}_1 & 0 & 0 & \hat{t}_0 & 0 & 0 & 0 & 0 & 0 & 0 & \hat{t}_3 & 0 & 0 & \hat{t}_2 & 0 \\ 0 & 0 & \hat{t}_2 & 0 & 0 & \hat{t}_1 & 0 & 0 & \hat{t}_0 & 0 & 0 & 0 & 0 & 0 & 0 & \hat{t}_3 \\ 0 & 0 & 0 & \hat{t}_3 & 0 & 0 & \hat{t}_2 & 0 & 0 & \hat{t}_1 & 0 & 0 & \hat{t}_0 & 0 & 0 & 0 \\ \hline \hat{l}_0 & 0 & 0 & 0 & 0 & \hat{l}_1 & 0 & 0 & 0 & 0 & 0 & \hat{l}_3 & 0 & 0 & \hat{l}_2 & 0 \\ 0 & 0 & \hat{l}_2 & 0 & \hat{l}_0 & 0 & 0 & 0 & 0 & \hat{l}_1 & 0 & 0 & 0 & 0 & 0 & \hat{l}_3 \\ 0 & 0 & 0 & \hat{l}_3 & 0 & 0 & \hat{l}_2 & 0 & \hat{l}_0 & 0 & 0 & 0 & 0 & \hat{l}_1 & 0 & 0 \\ 0 & \hat{l}_1 & 0 & 0 & 0 & 0 & 0 & \hat{l}_3 & 0 & 0 & \hat{l}_2 & 0 & \hat{l}_0 & 0 & 0 & 0 \end{array}\right) \end{array} \tag{4.13}$$

$$\begin{array}{llll} \hat{h}_0 = \hat{h}_0(D) & \hat{h}_1 = \hat{h}_1(D) & \hat{h}_2 = \hat{h}_2(D) & \hat{h}_3 = \hat{h}_3(D) \\ \hat{t}_0 = \hat{t}_0(D) & \hat{t}_1 = \hat{t}_1(D) & \hat{t}_2 = \hat{t}_2(D) & \hat{t}_3 = \hat{t}_3(D) \\ \hat{l}_0 = \hat{l}_0(D) & \hat{l}_1 = \hat{l}_1(D) & \hat{l}_2 = \hat{l}_2(D) & \hat{l}_3 = \hat{l}_3(D) \end{array}$$

where the parity-check matrix of the rate $\widehat{R} = 3/4$ constituent convolutional code with free distance $\hat{d}_{\text{free}} = 5$ is determined by

$$\widehat{H}(D) = \begin{pmatrix} \hat{h}_0(D) & \hat{h}_1(D) & \hat{h}_2(D) & \hat{h}_3(D) \end{pmatrix} \tag{4.14}$$

with

$$\begin{array}{ll} \hat{h}_0(D) = 1 + D + D^2 + D^4 & \hat{h}_1(D) = 1 + D + D^2 + D^3 + D^4 \\ \hat{h}_2(D) = 1 + D + D^3 + D^5 & \hat{h}_3(D) = 1 + D^2 + D^5 \end{array}$$

and $\big(\hat{t}_0(D), \hat{t}_1(D), \hat{t}_2(D), \hat{t}_3(D)\big)$ and $\big(\hat{l}_0(D), \hat{l}_1(D), \hat{l}_2(D), \hat{l}_3(D)\big)$ are two out of 24 possible permutations of $\big(\hat{h}_0(D), \hat{h}_1(D), \hat{h}_2(D), \hat{h}_3(D)\big)$.

Combining the parity-check matrix (4.4) of the parent convolutional code of the underlying graph with the parity-check matrix (4.14) of the constituent convolutional code and its permutations, yields the two-dimensional, rate $R = 1/4$ graph-based parent woven convolutional code with parity-check matrix

$$H(D,Z) = \begin{pmatrix} \hat{h}_1(D) & \hat{h}_2(D) & \hat{h}_3(D) & \hat{h}_4(D) \\ \hat{t}_1(D) & \hat{t}_2(D)Z & \hat{t}_3(D)Z^2 & \hat{t}_4(D)Z^3 \\ \hat{l}_1(D) & \hat{l}_2(D)Z^3 & \hat{l}_3(D)Z & \hat{l}_4(D)Z^2 \end{pmatrix} \tag{4.15}$$

over the two formal variables D and Z. In particular, by tailbiting (4.15) in the Z-dimension to length $M = 4$, we return to the rate $R = 4/16$ woven convolutional graph code (4.13).

4.3 ASYMPTOTIC BOUND ON THE FREE DISTANCE OF WOVEN CONVOLUTIONAL GRAPH CODES

Consider a woven graph code whose constituent block code is obtained by tailbiting a rate $\widehat{R} = \hat{b}/\hat{c}$ convolutional code $\widehat{\mathcal{C}}$ with overall constraint length $\hat{\nu}$, memory $\widehat{m}$, and syndrome memory $\widehat{m}_{\mathrm{s}}$, where in general $\widehat{m} \neq \widehat{m}_{\mathrm{s}}$. Denote the length of a codeword of the tailbiting block code in $n\hat{c}$-tuples by M and let the minimum distance of the woven graph code be $d_{\min}^{\mathrm{wg}}$, while its rate is given by $R_{\mathrm{wg}} = 1 - s(1 - \widehat{R})$ according to (4.3). Considering tailbiting codes (instead of zero-tail or other termination techniques) simplifies the analysis since their code rate coincides with the rate of the parent convolutional code. Moreover, if M tends towards infinity, the minimum distance of a tailbiting code coincides with the free distance of its parent convolutional code, that is, $d_{\min}^{\mathrm{wg}} = d_{\mathrm{free}}^{\mathrm{wg}}$ if $M \to \infty$.

Since the overall constraint length ν_{wg} of such a woven convolutional graph code is at most $sn\hat{\nu}$, its memory m_{wg} can be upper-bounded by

$$m_{\mathrm{wg}} \leq \nu_{\mathrm{wg}} \leq sn\hat{\nu} \leq sn\hat{b}\widehat{m} \tag{4.16}$$

Theorem 4.1 (Costello lower bound) For any $\epsilon > 0$, some $m_0 > 0$, and for all $m_{\mathrm{wg}} > m_0$ within the random ensemble of rate $R_{\mathrm{wg}} = 1 - s(1 - \widehat{R})$ woven graph codes over bipartite graphs with an s-partite set of constraint nodes with equally sized partitions, as well as constituent convolutional codes of rate $\widehat{R} = \hat{b}/\hat{c}$ and memory $\hat{m}$, there exists a woven convolutional graph code such that its relative free distance $\delta_{\mathrm{free}}^{\mathrm{wg}}$ satisfies the Costello lower bound [JZ99],

$$\delta_{\mathrm{free}}^{\mathrm{wg}} = \frac{d_{\mathrm{free}}^{\mathrm{wg}}}{\hat{c}m_{\mathrm{wg}}} \geq -\frac{R_{\mathrm{wg}}}{\log_2\left(2^{1-R_{\mathrm{wg}}} - 1\right)} - \epsilon \tag{4.17}$$

if

$$s \geq \begin{cases} 2, & \text{if } R_{\mathrm{wg}} \leq 0.402 \\ 3, & \text{if } R_{\mathrm{wg}} > 0.402 \end{cases} \tag{4.18}$$

Proof. Analogously to the derivations within the proof of the Varshamov-Gilbert bound in [BKJZ10], let w denote the Hamming weight of the codeword $\boldsymbol{v}$ of the random binary woven graph code determined by the time-varying random parity-check matrix

$$H_{\mathrm{wg}} = \begin{pmatrix} \pi_1\left(\widetilde{H}_0\right) \\ \pi_2\left(\widetilde{H}_1\right) \\ \vdots \\ \pi_s\left(\widetilde{H}_{s-1}\right) \end{pmatrix} \tag{4.19}$$

where $\pi_i(\widetilde{H}_i)$ denotes a random column permutation of $\widetilde{H}_i$. Each of the s submatrices $\widetilde{H}_i$, $i = 0, 1, \ldots, s-1$, is an $n \times n$ block matrix (or in other words a binary matrix of size $nM(\hat{c} - \hat{b}) \times nM\hat{c}$)

$$\widetilde{H}_i = \begin{pmatrix} \widehat{H}_i^{(0)} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \widehat{H}_i^{(1)} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \widehat{H}_i^{(n-1)} \end{pmatrix}$$

where n denotes the number of constituent block codes within each subset of constraint nodes.

Note that within the initial construction in (4.1), the submatrix $\widetilde{H}_0$ had a regular block structure, while all other submatrices $\widetilde{H}_i$, $i = 1, 2, \ldots, s-1$, were random column permutation. In (4.19), we consider its generalization where all submatrices are independent, random column permutations of their corresponding submatrices $\widetilde{H}_i$ with regular block structure.

Moreover, each of the n nonzero blocks within $\widetilde{H}_i$ denotes a random parity-check matrix $\widehat{H}_i^{(t)}$, $t = 0, 1, \ldots, n-1$, $i = 0, 1, \ldots, s-1$, given by

$$\widehat{H}_i^{(t)} = \begin{pmatrix} \widehat{H}_{00} & & & & \widehat{H}_{\hat{m}_{\mathrm{s}}(M-\hat{m}_{\mathrm{s}})} & \ldots & \widehat{H}_{2(M-2)} & \widehat{H}_{1(M-1)} \\ \widehat{H}_{10} & \widehat{H}_{01} & & & & \ddots & \vdots & H_{2(M-1)} \\ \widehat{H}_{20} & \widehat{H}_{11} & \ddots & & & & \widehat{H}_{\hat{m}_{\mathrm{s}}(M-2)} & \vdots \\ \vdots & \widehat{H}_{21} & \ddots & \widehat{H}_{0(M-\hat{m}_{\mathrm{s}}-1)} & & & & \widehat{H}_{\hat{m}_{\mathrm{s}}(M-1)} \\ \widehat{H}_{\hat{m}_{\mathrm{s}}0} & \vdots & \ddots & \widehat{H}_{1(M-\hat{m}_{\mathrm{s}}-1)} & \widehat{H}_{0(M-\hat{m}_{\mathrm{s}})} & & & \\ & \widehat{H}_{\hat{m}_{\mathrm{s}}1} & \ddots & \widehat{H}_{2(M-\hat{m}_{\mathrm{s}}-1)} & \widehat{H}_{1(M-\hat{m}_{\mathrm{s}})} & \ddots & & \\ & & \ddots & \ldots & \ldots & \ddots & \widehat{H}_{0(M-2)} & \\ & & & \widehat{H}_{\hat{m}_{\mathrm{s}}(M-\hat{m}_{\mathrm{s}}-1)} & \widehat{H}_{(\hat{m}_{\mathrm{s}}-1)(M-\hat{m}_{\mathrm{s}})} & \ldots & \widehat{H}_{1(M-2)} & \widehat{H}_{0(M-1)} \end{pmatrix} \tag{4.20}$$

which can be interpreted as the parity-check matrix of a rate $R = M\hat{b}/M\hat{c}$ tailbiting block code with tailbiting length M obtained from a parent rate

$\widehat{R} = \hat{b}/\hat{c}$ convolutional code $\widehat{\mathcal{C}}$ with memory $\widehat{m}$ and syndrome memory $\widehat{m}_{\mathrm{s}}$. All binary matrices $\widehat{H}_{kl}$, $k = 0, 1, \ldots, \widehat{m}_{\mathrm{s}}$, $l = 0, 1, \ldots, M-1$, in (4.20) have size $(\hat{c} - \hat{b}) \times \hat{c}$ and can be obtained separately for each block matrix $\widehat{H}_i^{(t)}$ by randomly and independently choosing 0s and 1s from an equiprobable distribution.

In the following, we are going to find the parameter d, such that the probability $P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \mid w\right)$ that there exists a woven graph code with a codeword $\boldsymbol{v}$ of Hamming weight $w_{\mathrm{H}}(\boldsymbol{v}) = w$ tends to zero for $w < d$, when M tends to infinity. In particular note that for $M \to \infty$ the constituent block code becomes a constituent convolutional code.

Clearly, $P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \mid w\right)$ can be written as

$$\begin{aligned} P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \,\middle|\, w\right) &= \sum_{h} P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \,\middle|\, w, h\right) P\left(h \,\middle|\, w\right) \\ &\leq \max_{h} \left\{ P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \,\middle|\, h, w\right) \right\} \end{aligned} \tag{4.21}$$

where $h \leq M$ denotes the number of nontrivial subblocks of length $\hat{c}$ in each of the constituent codewords $\hat{\boldsymbol{v}}$ of length $M\hat{c}$. The conditional probability in the last inequality can be expressed as

$$P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \,\middle|\, h, w\right) = \sum_{\boldsymbol{j}} P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \,\middle|\, h, w, \boldsymbol{j}\right) P\left(\boldsymbol{j} \mid h, w\right) \tag{4.22}$$

where $\boldsymbol{j} = (j_0, j_1, \ldots, j_{s-1})$, whose elements j_i, $i = 0, 1, \ldots, s-1$, denote the number of nontrivial constituent codewords of length $M\hat{c}$ which are part of the woven graph codeword of length $nM\hat{c}$ within the ith subset of constraint nodes $\mathcal{V}_1^{(i)}$.

The events that a random codeword $\boldsymbol{v}$ and a random matrix $\widetilde{H}_i$ satisfy the ith subset of parity checks, that is, $\boldsymbol{v}\widetilde{H}_i^{\mathrm{T}} = \mathbf{0}$, for different i are stochastically dependent in the product space of random equiprobable sequences $\boldsymbol{v}$ and random parity-check matrices, because the same fragments of $\boldsymbol{v}$ participate in different sets of parity checks. However, for all $\boldsymbol{v}$ satisfying the conditions w, h, and $\boldsymbol{j}$, the probabilities of $\boldsymbol{v}\widetilde{H}_i^{\mathrm{T}} = \mathbf{0}$ depend only on $\widetilde{H}_i$, and thus the events are *conditionally* independent for given $\boldsymbol{v}$, w, h, and $\boldsymbol{j}$. Taking into account that there exist not more than $\binom{n\hat{c}h}{w}$ codewords $\boldsymbol{v}$ satisfying the mentioned conditions, we obtain the upper bounds

$$P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \,\middle|\, h < M, w, \boldsymbol{j}\right) \leq M^{2Ms/\widehat{m}} \binom{n\hat{c}h}{w} \left(\prod_{i=0}^{s-1} 2^{j_i} (h - \widehat{m}) \hat{b} 2^{-j_i h \hat{c}} \right) \tag{4.23}$$

and

$$P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \,\middle|\, h = M, w, \boldsymbol{j}\right) \leq \binom{nM\hat{c}}{w} \left(\prod_{i=0}^{s-1} 2^{j_i M \hat{b}} 2^{-j_i M \hat{c}} \right) \tag{4.24}$$

Within the derivation of (4.23) we used that a constituent codeword $\hat{\boldsymbol{v}}$ of length $M\hat{c}$ which contains h nontrivial $\hat{c}$-subblocks is generated by at most $h - \widehat{m}$ nontrivial binary information $\hat{b}$-tuples. Moreover, the number of possible locations of these h out of M subblocks can be upper-bounded by $M^{2M/\widehat{m}}$, taking into account that every codeword contains one or more (cyclic) trivial bursts which originate from at least $\widehat{m}$ consecutive all-zero information $\hat{b}$-tuples. Thus, the number of bursts can not be larger than $M/\widehat{m}$. Additionally, since there are at most M possible starting positions for each burst and less than M possible ending positions within each of the s subsets of constraint nodes, the maximum total number of possible choices for h nontrivial subblocks is upper-bounded by $(M^{2(M/\widehat{m})})^s$.

On the other hand, if all M subblocks are nontrivial, that is if $h = M$, these considerations do not have to be taken into account and the corresponding probability can be upper-bounded by the tighter expression (4.24).

Before continuing with the proof we shall recall the following Lemma, proven in [BKJZ10], in order to estimate the probability $P(\boldsymbol{j} \mid h, w)$ for a woven graph code with a constituent block code.

Lemma 4.2 For the random ensemble of binary woven graph codes with constituent block codes the probability $P(\boldsymbol{j} \mid h, w)$ that a woven graph codeword of length $nM\hat{c}$ with h nontrivial subblocks of length $n\hat{c}$ and Hamming weight w consists of $\boldsymbol{j} = (j_0, j_1, \ldots, j_{s-1})$ nonzero constituent codewords of length $M\hat{c}$ with h nontrivial subblocks of length $\hat{c}$ within the s subsets of constraint nodes can be upper-bounded by

$$P(\boldsymbol{j} \mid h, w) \leq \prod_{i=0}^{s-1} \frac{\binom{n}{j_i}\binom{\hat{c}h}{w/j_i}^{j_i}\binom{w-1}{j_i-1}}{\binom{n\hat{c}h}{w}}$$

Combining (4.22) and (4.23) with Lemma 4.2 yields

$$\begin{aligned}
&P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \boldsymbol{0} \;\middle|\; h < M, w\right) \\
&\quad \leq M^{2Ms/\widehat{m}} \sum_{\boldsymbol{j}} \binom{n\hat{c}h}{w}^{1-s} \prod_{i=0}^{s-1} 2^{j_i\hat{b}(h-\widehat{m})-j_i\hat{c}h} \binom{n}{j_i}\binom{\hat{c}h}{w/j_i}^{j_i}\binom{w-1}{j_i-1} \\
&\quad \leq M^{2Ms/\widehat{m}} \binom{n\hat{c}h}{w}^{1-s} (n+1)^s \\
&\qquad \times \max_{(j_0,\ldots,j_{s-1})} \prod_{i=1}^{s-1} 2^{j_i b(h-\widehat{m})-j_i\hat{c}h} \binom{n}{j_i}\binom{\hat{c}h}{w/j_i}^{j_i}\binom{w-1}{j_i-1} \\
&\quad \leq M^{2Ms/\widehat{m}} (n+1)^s \binom{n\hat{c}h}{w}^{1-s} \\
&\qquad \times \max_{j \leq n} \left\{ \left(2^{j\hat{b}(h-\widehat{m})-j\hat{c}h} \binom{n}{j}\binom{\hat{c}h}{w/j}^{j}\binom{w-1}{j-1} \right)^s \right\}
\end{aligned} \tag{4.25}$$

If all M codeword blocks are nontrivial, (4.23) can be replaced by (4.24), and hence it follows from (4.22), (4.24), and Lemma 4.2 in a similar way that

$$P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \;\middle|\; h = M, w\right) \leq (n+1)^s \binom{nM\hat{c}}{w}^{1-s} \max_{j \leq n} \left\{ \left(2^{jM(\hat{b}-\hat{c})} \binom{n}{j} \binom{\hat{c}l}{w/j}^j \binom{w-1}{j-1} \right)^s \right\} \quad (4.26)$$

Letting the tailbiting length M tend to infinity, we obtain a woven convolutional graph code (with constituent convolutional code). In the following, we will prove that in such a case the probabilities in (4.25) and (4.26) are strictly below one for a sufficiently large memory $\hat{m}$ of the constituent code, that is, there exists a woven convolutional graph code with free distance $d_{\mathrm{free}}^{\mathrm{wg}} > w$. When expressing these probabilities by their exponents to base 2, normalized by m_{wg}, while the memory $\hat{m}$ of their constituent code tends to infinity, this holds if and only if at least one of the corresponding exponents is strictly positive. Hence, we obtain

$$\begin{aligned} F_{h<M}(\delta) &= \lim_{\hat{m}\to\infty} \frac{-\log_2 P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \;\middle|\; h < M, w\right)}{\hat{c}sn\hat{b}\hat{m}} \\ &\geq \min_{\gamma\in(0,1],\, \mu\geq 1} \left\{ \left(1 - \frac{1}{s}\right) \mu h\left(\frac{\delta s}{\mu}\right) + \gamma\left(1 + \frac{\mu-1}{s}(1 - R_{\mathrm{wg}})\right) \right. \\ &\qquad \left. -\gamma\mu h\left(\frac{\delta s}{\gamma\mu}\right) \right\} \end{aligned} \quad (4.27)$$

$$\begin{aligned} F_{h=M}(\delta) &= \lim_{\hat{m}\to\infty} \frac{-\log_2 P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \;\middle|\; h = M, w\right)}{\hat{c}sn\hat{b}\hat{m}} \\ &\geq \max_{\theta>1} \min_{\gamma\in(0,1]} \left\{ \left(1 - \frac{1}{s}\right) \theta h\left(\frac{\delta s}{\theta}\right) + \gamma\theta\frac{1 - R_{\mathrm{wg}}}{s} - \gamma\theta h\left(\frac{\delta s}{\gamma\theta}\right) \right\} \end{aligned} \quad (4.28)$$

where we neglected all terms not depending on M (and hence h) and used the fact that

$$\log_2 \binom{b}{a} \simeq bh\left(\frac{a}{b}\right)$$

where $h(x)$ is the binary entropy function (1.3), and introduced the following abbreviations

$$\delta = \frac{w}{\hat{c}sn\hat{b}\hat{m}}, \quad \mu = \frac{h}{\hat{b}\hat{m}}, \quad \theta = \frac{M}{\hat{b}\hat{m}}, \quad \text{and} \quad \gamma = \frac{j}{n}$$

Similarly, the exponent of (4.21) can be lower-bounded by combining (4.27) and (4.28) as

$$F(\delta) = \lim_{\hat{m}\to\infty} \frac{-\log_2 P\left(\boldsymbol{v}H_{\mathrm{wg}}^{\mathrm{T}} = \mathbf{0} \;\middle|\; w\right)}{\hat{c}sn\hat{b}\hat{m}} = \min\left\{F_{h<M}(\delta), F_{h=M}(\delta)\right\}$$

Since the truncation length M tends towards infinity, that is, $\theta \to \infty$, it is straightforward to verify that $F_{h=M} \to \infty$ and only $F_{h<M}$ determines the probability exponent $F(\delta)$. Moreover, to find the minimum of $F_{h<M}(\delta)$ over μ and γ, we use the fact that the function

$$f(x) = \beta x - xh\left(\frac{\alpha}{x}\right)$$

is convex if $x > \alpha > 0$ and achieves its minimum

$$f(x_0) = \alpha \log_2(2^\beta - 1)$$

at the point

$$x_0 = \frac{\alpha}{1 - 2^{-\beta}}$$

Minimizing (4.27) similarly over $0 < \gamma \leq 1$, leads to

$$\gamma_{\text{opt}} = \min\left\{1, \frac{\delta s}{\mu(1 - 2^{-\beta})}\right\} \tag{4.29}$$

with

$$\beta = \frac{s + (\mu - 1)(1 - R_{\text{wg}})}{s\mu} \tag{4.30}$$

Obviously, if s is chosen large enough, γ_{opt} follows from (4.29) as $\gamma_{\text{opt}} = 1$ and the probability exponent in (4.27) can be expressed by

$$F_{h<M}(\delta) \geq \min_{\mu \geq 1}\left\{-\frac{\mu}{s}h\left(\frac{\delta s}{\mu}\right) + \frac{\mu}{s}(1 - R_{\text{wg}})\right\} + 1 - \frac{1 - R_{\text{wg}}}{s} \tag{4.31}$$

Minimizing this expression over μ yields

$$\mu_{\text{opt}} = \frac{\delta s}{1 - 2^{R_{\text{wg}} - 1}} \tag{4.32}$$

and thus

$$F_{h<M}(\delta) \geq \delta \log_2\left(2^{1 - R_{\text{wg}}} - 1\right) + 1 - \frac{1 - R_{\text{wg}}}{s} \tag{4.33}$$

As mentioned previously, for the Costello bound (4.17) to hold, $F(\delta)$ needs to be strictly positive, such that the probability of codewords with relative Hamming weight below the Costello bound is strictly below one if s is large enough. To complete the proof we determine the minimal value of s for which this is satisfied. The lower limit on s within our derivation is imposed by the assumption that $\gamma_{\text{opt}} = 1$, which holds as long as

$$\frac{\delta s}{\mu(1 - 2^{-\beta})} \geq 1 \tag{4.34}$$

where β is defined by (4.30). By combining (4.17), (4.29), and (4.32), it follows that (4.34) is fulfilled if s is selected according to (4.18), which completes the proof. ■

Graph	(N, K)	Constituent convolutional code	Perm.	ν	d_{free}	Spectrum
Rate $R_{\text{wg}} = 1/4$						
girth $g = 4$ $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & D & D^2 & D^3 \\ 1 & D^3 & D & D^2 \end{pmatrix}$	$(16, 4)$	$\hat{h}_0(D) = 1 + D + D^2 + D^4$ $\hat{h}_1(D) = 1 + D + D^2 + D^3 + D^4$ $\hat{h}_2(D) = 1 + D + D^3 + D^5$ $\hat{h}_3(D) = 1 + D^2 + D^5$ $\hat{d}_{\text{free}} = 5$	$(1, 3, 2, 4)$ $(3, 4, 1, 2)$	49	68	1,0,0,0,1,0,0,0,0,0,...
	$(20, 5)$		$(0, 2, 3, 1)$ $(2, 3, 0, 1)$	67	120	1,...
Rate $R_{\text{wg}} = 1/3$						
girth $g = 8$ $\begin{pmatrix} 1 & 1 & 1 \\ 1 & D & D^2 \end{pmatrix}$	$(9, 3)$	$\hat{h}_0(D) = 1 + D + D^4$ $\hat{h}_1(D) = 1 + D + D^3 + D^4 + D^5$ $\hat{h}_2(D) = 1 + D^2 + D^3 + D^4 + D^5$ $\hat{d}_{\text{free}} = 6$	$(2, 0, 1)$	26	30	4,0,0,0,0,0,3,0,6,0,...
girth $g = 12$ $\begin{pmatrix} 1 & 1 & 1 \\ 1 & D & D^3 \end{pmatrix}$	$(21, 7)$		$(0, 2, 1)$	64	32	7,0,0,0,0,0,7,0,7,0,...
Rate $R_{\text{wg}} = 1/2$						
girth $g = 8$ $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & D & D^2 & D^3 \end{pmatrix}$	$(16, 8)$	$\hat{h}_0(D) = 1 + D + D^2 + D^4$ $\hat{h}_1(D) = 1 + D + D^2 + D^3 + D^4$ $\hat{h}_2(D) = 1 + D + D^3 + D^5$ $\hat{h}_3(D) = 1 + D^2 + D^5$ $\hat{d}_{\text{free}} = 5$	$(2, 3, 1, 0)$	38	31	8,14,16,16,112,185,...

Table 4.1: Examples of promising woven convolutional graph codes.

4.4 EXAMPLES

Parameters for some promising examples of woven convolutional graph codes with rate $R = 1/4$, $R = 1/3$, and $R = 1/2$ and free distance up to $d_{\text{free}} = 120$ are presented in Table 4.1.

For each entry, the underlying graph is specified in the first column by the parity-check matrix $H(D)$ of its parent convolutional code together with its girth g. Tailbiting $H(D)$ yields the parity-check matrix H of the graph-based block code $\mathcal{B}$, whose dimensions (N, K) are specified in the second column. Interpreting the tailbiting parity-check matrix H as a biadjacency matrix yields the corresponding bipartite graph $\mathcal{G}$ (cf. Section 3.1).

The corresponding rate $\hat{R} = \hat{b}/\hat{c}$ constituent convolutional code is determined by its parity-check polynomials in column three together with the used permutation matrices in column four. Interpreting these codes as block codes of the same rate over the field of binary Laurent series yields the final woven convolutional graph code.

In column five, the overall constraint length ν of the corresponding minimal-basic generator matrix $G_{\text{wg}}(D)$ of the woven convolutional graph code is specified. Its free distance d_{free} as well as its first Viterbi spectral components, determined by the BEAST (cf. Section 2.4), are given in the last two columns of Table 4.1. Even though the girth of the underlying graph as well as the free distance $\hat{d}_{\text{free}}$ of the constituent convolutional code are in general rather small, it is possible to construct woven convolutional graph codes with free distances up to at least 120.

Consider for example the second entry in Table 4.1, whose parent convolutional code is determined by the parity-check matrix

$$H(D) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & D & D^2 & D^3 \\ 1 & D^3 & D & D^2 \end{pmatrix} \tag{4.35}$$

By tailbiting its semi-infinite parity-check matrix H to length $M = 5$, we obtain the parity-check matrix $H^{(\text{tb})}$ of a rate $R = 5/20$ block code as specified in column two. Using the concept of biadjacency matrices yields the corresponding bipartite graph $\mathcal{G}$ with 15 symbol nodes and 20 constraint nodes, whose set of constraint nodes is 3-partite with equally sized partitions.

While we assign the four parity-check polynomials $\hat{h}_i(D)$ of the rate $R = 3/4$ constituent convolutional code from column three to the first set of constraint nodes in their natural order, that is, $(\hat{h}_0(D), \hat{h}_1(D), \hat{h}_2(D), \hat{h}_3(D))$, the constituent parity-check polynomials for the second and third set of constraint nodes are permuted according to $(0, 2, 3, 1)$ and $(2, 3, 0, 1)$, that is, $(\hat{h}_0(D), \hat{h}_2(D), \hat{h}_3(D), \hat{h}_1(D))$ and $(\hat{h}_2(D), \hat{h}_3(D), \hat{h}_0(D), \hat{h}_1(D))$, respectively.

This yields a rate $R_{\text{wg}} = 5/20$ woven convolutional graph code, whose minimal-basic generator matrix has an overall constraint length of $\nu = 67$. It is well known that the row distances d_i^{r}, $i = 0, 1, \ldots$, upper-bound the free distance (cf. Subsection 7.1.2). For the considered minimal-basic generator matrix we obtain $d_0^{\text{r}} = d_1^{\text{r}} = d_2^{\text{r}} = 130$ and $d_3^{\text{r}} = \ldots = d_6^{\text{r}} = 120$, and hence we conclude that its free distance is upper-bounded by $d_{\text{free}} \leq 120$.

For further code analysis, we use the BEAST (cf. Section 2.4) . Calculating the free distance for such a code would take prohibitively long time without splitting the computational workload on several processors. Using parallel computing, the free distance of this woven convolutional graph code is determined to be as large as $d_{\text{free}} = 120$, where the individual forward and backward sets are sorted and merged on individual processors. Moreover, there exists only one codeword of weight 120 with corresponding length $(1 + 3 + 11)20 = 300$. Based on our experience obtained from studying less complex woven convolutional graph codes, we conjecture that the next nonzero spectral component occurs at weight 130 (cf. the sequence of row distances shown above).

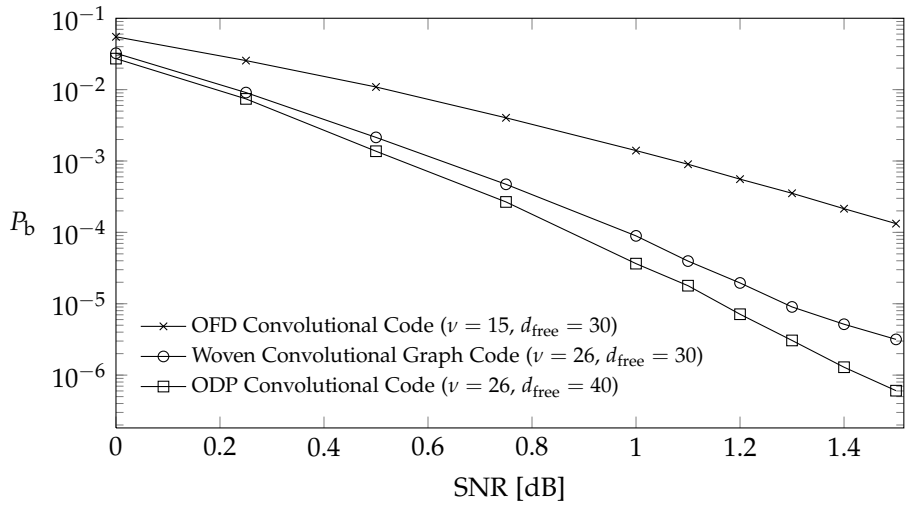


Figure 4.2: Comparison of BER performances for Viterbi decoding for a rate $R_{\text{wg}} = 3/9$ woven convolutional graph code with $\nu = 26$ and $d_{\text{free}} = 30$ in comparison to a rate $R = 1/3$ ODP convolutional code with $\nu = 26$ and $d_{\text{free}} = 40$ as well as a rate $R = 1/3$ OFD convolutional code with $\nu = 15$ and $d_{\text{free}} = 30$.

4.5 SIMULATION RESULTS

To demonstrate the error-correcting capabilities of woven convolutional graph codes, the BER performance for the minimal-basic encoding matrix of a rate $R_{\text{wg}} = 3/9$ woven convolutional graph code with overall constraint length $\nu = 26$ and free distance $d_{\text{free}} = 30$ is simulated and illustrated in Figure 4.2. For comparison we include the corresponding BER performance for the minimal-basic encoding matrices of the rate $R = 1/3$ *optimum distance profile* (ODP) convolutional code (cf. Chapter 7) with the same overall constraint length $\nu = 26$ but with larger free distance $d_{\text{free}} = 40$, as well as the rate $R = 1/3$ *optimum free distance* (OFD) convolutional code (cf. Chapter 7) with the same free distance $d_{\text{free}} = 30$ but with smaller overall constraint length $\nu = 15$.

For low SNRs (0.0–0.5 dB) the woven convolutional graph code yields almost the same BER performance as the ODP convolutional code, despite the large difference in their free distances. However, for higher SNRs, with BER around 10^{-5}, the woven convolutional graph codes loses about 0.2 dB compared to the ODP convolutional code. The OFD convolutional code, on the other hand, has a significantly worse BER performance over the whole range of SNRs.

5

A Closed Form Expression for the Exact Bit Error Probability

The achievable BER performance when using a convolutional code to communicate over a BSC or a quantized AWGN channel together with an ML decoding algorithm like the Viterbi algorithm (cf. Section 2.3) is of particular interest. In 1971, Viterbi [Vit71] published a nowadays classical upper bound on the bit error probability P_b, derived from the extended path weight enumerators obtained by using a signal flow chart technique for convolutional encoders. Later, van de Meeberg [Van74] used a very clever observation to tighten Viterbi's bound for large signal-to-noise ratios (SNRs).

The challenging problem of deriving an expression for the *exact* (decoding) bit error probability was first addressed by Morrissey in 1970 [Mor70] for a suboptimal feedback decoding algorithm. He obtained the same expression for the exact bit error probability of the rate $R = 1/2$, memory $m = 1$ (2-state) convolutional encoder with generator matrix $G(D) = (1 \quad 1+D)$ that Best et al. [BBL+95] obtained for the Viterbi algorithm. The latter method is based on considering a Markov chain of the so-called metric states of the Viterbi decoder; an approach due to Burnashev and Cohn [BC90]. In 2004, Lentmaier et al. [LTZ04] published among other contributions an extension of this method to the rate $R = 1/2$ memory $m = 2$ (4-state) convolutional encoder with generator matrix $G(D) = (1+D^2 \quad 1+D+D^2)$.

Within this chapter we shall use a different and more general approach to derive a closed form expression for the exact (decoding) bit error probability for Viterbi decoding of convolutional encoders, when communicating over a BSC as well as a quantized AWGN channel. This new method allows the calculation of the exact bit error probability for more complex encoders in a wider range of code rates than the methods of [BBL+95] and [LTZ04]. By considering a random tie-breaking strategy, we average the information

weights over the sequences of channel noises and random coin-flipping decisions (where the coin may have more than two sides, depending on the code rate). Unlike the backward recursion in [BBL+95] and [LTZ04], the bit error probability averaged over time is obtained by deriving and solving a recurrent matrix equation for the average information weights at the current and previous states of a trellis section, when the maximum-likelihood branches are decided by the Viterbi decoder at the current state.

To illustrate our method, we use a rate $R = 2/3$ systematic convolutional 2-state encoder whose minimal realization is given in OCF, since this encoder is both general and simple.

In Section 5.1, the problem of computing the exact bit error probability is reformulated via the average information weights and the concept of normalized cumulative metrics is introduced. A recurrent matrix equation for these average information weights is derived in Section 5.2. The derivations used to solve this general equation are discussed in Section 5.3, before presenting additional examples of various rate $R = 1/2$ and $R = 2/3$ convolutional encoders with different memories in Section 5.4. Furthermore, a rate $R = 1/2$ (4-state) encoder used to communicate over a quantized AWGN channel is given an its bit error probability is compared for different quantization schemes.

Before proceeding, we would like to emphasize that the bit error probability is an encoder property, neither a generator matrix property nor a convolutional code property.

5.1 EXPRESSING THE BIT ERROR PROBABILITY USING THE AVERAGE INFORMATION WEIGHTS

Assume that the all-zero sequence is transmitted over a BSC with crossover probability p and let $W_t(\boldsymbol{\sigma})$ denote the Hamming weight of the information sequence corresponding to the code sequence decided by the Viterbi decoder (cf. Section 2.3) at state $\boldsymbol{\sigma}$ and time instant t, where the state $\boldsymbol{\sigma}$ is given by (2.1). If the initial value $W_{t=0}(\boldsymbol{\sigma})$ is known, then the random process $W_t(\boldsymbol{\sigma})$, $t = 0, 1, 2 \ldots$ is a function of both the random sequence of the received c-tuples $\boldsymbol{r}_\tau$, $\tau = 0, 1, \ldots, t-1$, and the coin-flippings used to resolve ties.

In the following, the mathematical expectation of the random variable $W_t(\boldsymbol{\sigma})$ over this ensemble shall be determined, since for rate $R = b/c$ minimal convolutional encoders the bit error probability can be computed as the limit

$$P_\mathrm{b} = \lim_{t\to\infty} \frac{E\left[W_t(\boldsymbol{\sigma} = \mathbf{0})\right]}{tb} \tag{5.1}$$

assuming that this limit exists.

Note that if we consider nonminimal encoders, we have to additionally take all states equivalent to the all-zero state into account.

In Section 2.3, the cumulative Viterbi branch metric for a rate $R = b/c$ convolutional code is computed recursively in (2.38) as

$$\mu_{\mathrm{V}}\left(\boldsymbol{r}_{[0,t+1)}, \boldsymbol{v}_{[0,t+1)}\right) = \mu_{\mathrm{V}}\left(\boldsymbol{r}_{[0,t)}, \boldsymbol{v}_{[0,t)}\right) + \mu_{\mathrm{V}}\left(\boldsymbol{r}_t, \boldsymbol{v}_t\right) \tag{5.2}$$

where $\boldsymbol{r}_t$ and $\boldsymbol{v}_t$ denote the received c-tuple and code c-tuple at time instant t, while the sequences $\boldsymbol{r}_{[0,t)}$ and $\boldsymbol{v}_{[0,t)}$, $t > 0$, correspond to the segments $(\boldsymbol{r}_0\, \boldsymbol{r}_1 \ldots \boldsymbol{r}_{t-1})$ and $(\boldsymbol{v}_0\, \boldsymbol{v}_1 \ldots \boldsymbol{v}_{t-1})$, respectively, and μ_{V} refers to the cumulative Viterbi branch metric.

Let Σ denote the total number of different encoder states $\boldsymbol{\sigma}$ in either CCF and OCF (cf. Subsection 1.3.2). To simplify notations, we will index the encoder states by ζ and refer to the ith encoder state $\boldsymbol{\sigma}$ as $\zeta = i$, with $\zeta \in \{0, 1, \ldots, |\Sigma| - 1\}$, where the state $\boldsymbol{\sigma} = \mathbf{0}$ corresponds to $\zeta = 0$.

Denote by $\mu_{t-1}(\zeta)$ and $\mu_{(t)}(\zeta')$ the cumulative Viterbi branch metrics for the code sequence segments $\boldsymbol{v}_{[0,t)}$ and $\boldsymbol{v}_{[0,t+1)}$ leading to the encoder states ζ and ζ' at time instants $t-1$ and t, respectively. Then the calculation of the cumulative Viterbi branch metric (5.2) can be alternatively expressed as

$$\mu_{(t)}(\zeta') = \mu_{(t-1)}(\zeta) + \mu_{\mathrm{V}}\left(\boldsymbol{r}_t, \boldsymbol{v}_t\right) \tag{5.3}$$

where the branch between the two encoder states ζ and ζ' is labeled by the corresponding code c-tuple $\boldsymbol{v}_t$.

Combining all cumulative Viterbi branch metrics for time instant t yields the cumulative Viterbi branch metric vector $\boldsymbol{\mu}_t$ given by

$$\boldsymbol{\mu}_t = \left(\mu_t(0)\ \ \mu_t(1)\ \ \ldots\ \ \mu_t(|\Sigma| - 1)\right) \tag{5.4}$$

It is convenient to normalize these metrics such that the cumulative Viterbi branch metrics at every all-zero state are equal to zero, that is, we subtract the value $\mu_t(0)$ from $\mu_t(1)$, $\mu_t(2), \ldots, \mu_t(|\Sigma| - 1)$ and introduce the *normalized cumulative (Viterbi) branch metric vector*

$$\begin{aligned}\boldsymbol{\phi}_t &= \left(\phi_t(1)\ \ \phi_t(2)\ \ \ldots\ \ \phi_t(|\Sigma| - 1)\right) \\ &= \left(\mu_t(1) - \mu_t(0)\ \ \mu_t(2) - \mu_t(0)\ \ \ldots\ \ \mu_t(|\Sigma| - 1) - \mu_t(0)\right)\end{aligned}$$

For example, using a 2-state encoder we obtain the scalar

$$\phi_t = \mu_t(1) - \mu_t(0) = \phi_t(1)$$

while for a 4-state encoder we have the vector

$$\begin{aligned}\boldsymbol{\phi}_t &= \left(\mu_t(1) - \mu_t(0)\ \ \mu_t(2) - \mu_t(0)\ \ \mu_t(3) - \mu_t(0)\right) \\ &= \left(\phi_t(1)\ \ \phi_t(2)\ \ \phi_t(3)\right)\end{aligned}$$

The elements of the random vector $\boldsymbol{\phi}_t$ belong to a set whose cardinality M depends on the channel model, encoder structure, and the tie-breaking rule. Enumerate the vectors $\boldsymbol{\phi}_t$ by numbers ϕ_t which are random variables taking on M different integer values $\phi^{(0)}, \phi^{(1)}, \ldots, \phi^{(M-1)}$. The sequence of numbers ϕ_t forms an M-state Markov chain Φ_t with transition probability matrix $\Phi = \left(\phi_{jk}\right)$, where

$$\phi_{jk} = \Pr\left(\phi_{t+1} = \phi^{(k)} \;\middle|\; \phi_t = \phi^{(j)}\right) \tag{5.5}$$

Let $\boldsymbol{W}_t$ be the vector of information weights at time instant t that depends both on the $|\Sigma|$ encoder states ζ_t and on the M normalized cumulative metrics ϕ_t; that is, $\boldsymbol{W}_t$ is expressed as the following vector with $M\,|\Sigma|$ entries

$$\boldsymbol{W}_t = \Big(\boldsymbol{W}_t(\zeta=0)\;\; \boldsymbol{W}_t(\zeta=1)\;\; \ldots\;\; \boldsymbol{W}_t(\zeta=|\Sigma|-1)\Big) \tag{5.6}$$

where

$$\boldsymbol{W}_t(\zeta) = \Big(W_t(\phi^{(0)},\zeta)\;\; W_t(\phi^{(1)},\zeta)\;\; \ldots\;\; W_t(\phi^{(M-1)},\zeta)\Big) \tag{5.7}$$

Then (5.1) can be rewritten as

$$\begin{aligned} P_{\mathrm{b}} &= \lim_{t\to\infty} \frac{E[W_t(\boldsymbol{\sigma}=0)]}{tb} = \lim_{t\to\infty} \frac{E[W_t(\zeta=0)]}{tb} \\ &= \lim_{t\to\infty} \frac{\sum_{i=0}^{M-1} E[W_t(\phi^{(i)},\zeta=0)]}{tb} = \lim_{t\to\infty} \frac{E[\boldsymbol{W}_t(\zeta=0)]\mathbf{1}_{1,M}^{\mathrm{T}}}{tb} \\ &= \lim_{t\to\infty} \frac{\boldsymbol{w}_t(\zeta=0)\mathbf{1}_{1,M}^{\mathrm{T}}}{tb} = \lim_{t\to\infty} \frac{\boldsymbol{w}_t}{tb}\left(\mathbf{1}_{1,M}\,\mathbf{0}_{1,M}\ldots\,\mathbf{0}_{1,M}\right)^{\mathrm{T}} \end{aligned} \tag{5.8}$$

where $\mathbf{1}_{1,M}$ and $\mathbf{0}_{1,M}$ denote the all-one and the all-zero row vectors of length M, respectively, $\boldsymbol{w}_t$ represents the length $M\,|\Sigma|$ vector of the average information weights, while the length M vector of average information weights at the state ζ is given by $\boldsymbol{w}_t(\zeta)$. Note that the mathematical expectations in (5.8) are computed over sequences of channel noises and coin-flipping decisions.

To illustrate the introduced notations, we use the rate $R = 2/3$ memory $m = 1$ minimal encoder with overall constraint length $\nu = 2$ and systematic generator matrix

$$G(D) = \begin{pmatrix} 1 & 0 & 1+D \\ 0 & 1 & 1+D \end{pmatrix} \tag{5.9}$$

whose 2-state realization in OCF is illustrated in Figure 5.1.

Assuming that the normalized cumulative metric is $\phi_t = 0$, we obtain the eight trellis sections given in Figure 5.2, where bold branches correspond to branches decided by the Viterbi decoder at time instant $t+1$. When we have more than one branch with the maximum normalized cumulative metric entering the same state, we have a tie, which we resolve in this analysis by fair coin-flipping.

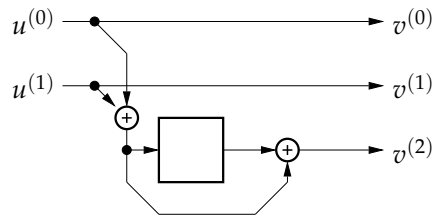


Figure 5.1: A minimal encoder for the generator matrix given in equation (5.9).

The eight trellis sections in Figure 5.2 yield the normalized cumulative metrics $\{-1, 0, 1\}$. Starting with $\phi_t = -1$ and $\phi_t = 1$, yields 16 additional trellis sections and the two additional normalized cumulative metrics $\{-2, 2\}$. From the metrics $\phi_t = -2$ and $\phi_t = 2$, we get another 16 trellis sections but those will not yield any new metrics.

Thus, in total we have $M = 5$ normalized cumulative metrics $\phi_t \in \{-2, -1, 0, 1, 2\}$. Together with the eight different received triples, $\boldsymbol{r}_t = 000$, 001, 010, 100, 011, 101, 110, and 111, they correspond to in total 40 different trellis sections. Hence, the normalized cumulative metric Φ_t is a 5-state Markov chain with transition probability matrix $\Phi = (\phi_{jk})$, $1 \leq j, k \leq 5$.

From the four trellis sections Figure 5.2(a), (b), (g), and (h), we obtain for example that

$$\begin{aligned}\phi_{0(-1)} &= P(\boldsymbol{r}_t = 000) + P(\boldsymbol{r}_t = 001) + P(\boldsymbol{r}_t = 110) + P(\boldsymbol{r}_t = 111) \\ &= q^3 + pq^2 + p^2q + p^3 = p^2 + q^2 \end{aligned} \tag{5.10}$$

while the four trellis sections, Figure 5.2(c), (d), (e), and (f) yield

$$\phi_{01} = pq^2 + pq^2 + p^2q + p^2q = 2pq \tag{5.11}$$

where $q = 1 - p$. Similarly, we obtain the remaining transition probabilities from the 32 trellis sections that are not included in Figure 5.2.

Finally, their transition probability matrix follows as

$$\Phi = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \\ \phi^{(j)} \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \quad \phi^{(k)} \\ \begin{pmatrix} q^3 + p^2q & 0 & p^3 + 3pq^2 & 0 & 2p^2q \\ q^3 + p^2q & 0 & p^3 + 3pq^2 & 0 & 2p^2q \\ 0 & p^2 + q^2 & 0 & 2pq & 0 \\ p^3 + pq^2 & 0 & q^3 + 3p^2q & 0 & 2p^2q \\ p^3 + pq^2 & 0 & q^3 + 3p^2q & 0 & 2p^2q \end{pmatrix} \end{array} \tag{5.12}$$

whose state diagram of the metric Markov chain is shown in Figure 5.3.

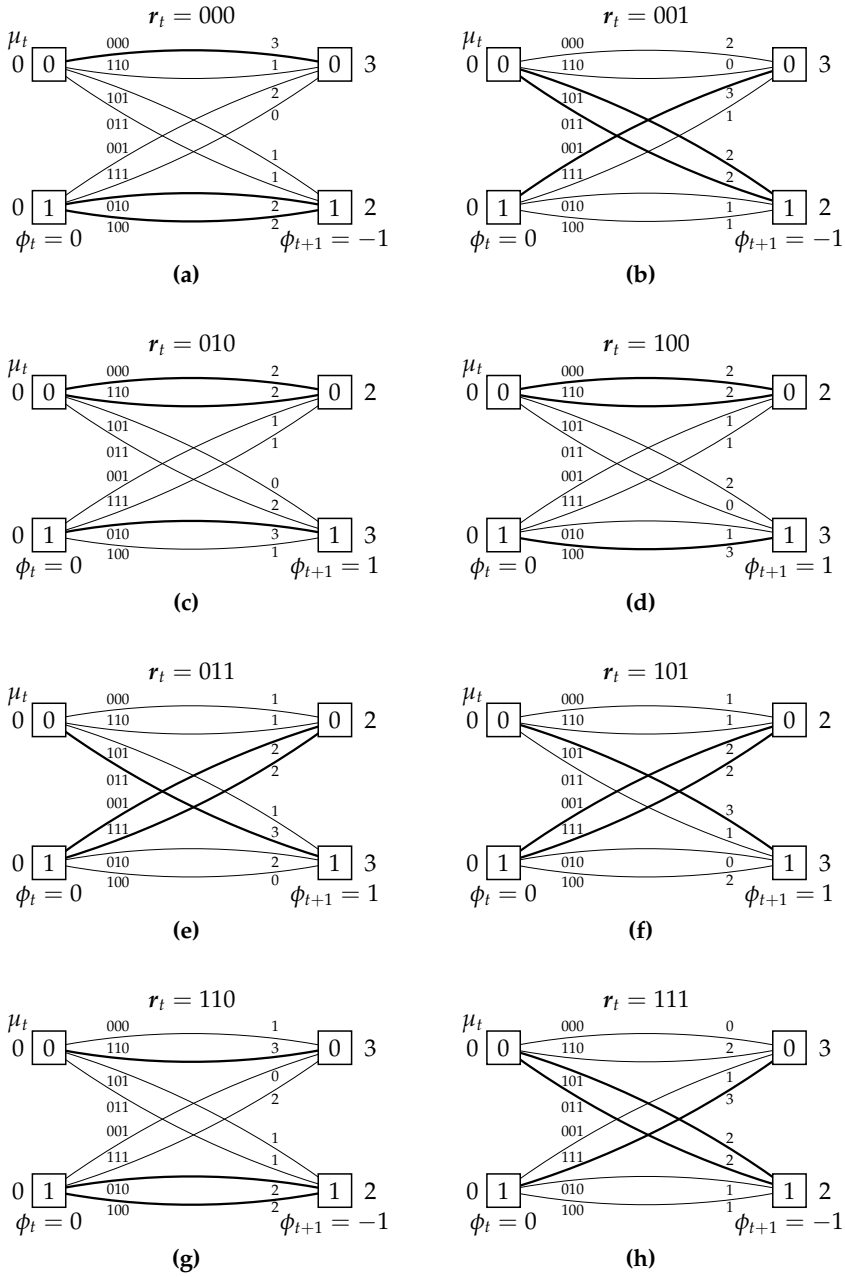


Figure 5.2: Eight (of a total of 40) trellis sections for the rate $R = 2/3$, 2-state encoder in Figure 5.1.

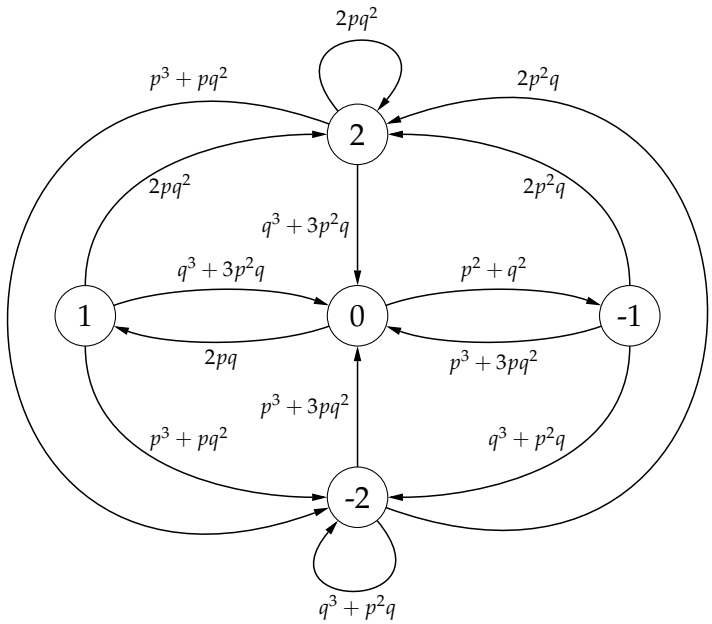


Figure 5.3: Illustration of the 5-state Markov chain formed by the sequences of normalized cumulative metrics ϕ_t.

Let $\boldsymbol{p}_t = (p_t^{(0)}\, p_t^{(1)} \ldots p_t^{(M-1)})$ denote the probabilities of the M different normalized cumulative metrics of Φ_t at time instant t, that is, $\phi_t \in \{\phi^{(0)}, \phi^{(1)}, \ldots, \phi^{(M-1)}\}$. Their corresponding stationary probability distribution is denoted by $\boldsymbol{p}_\infty = (p_\infty^{(0)}\, p_\infty^{(1)} \ldots p_\infty^{(M-1)})$ and follows as the solution of, for example, the first $M-1$ equations of

$$\boldsymbol{p}_\infty \Phi = \boldsymbol{p}_\infty \tag{5.13}$$

and

$$\sum_{i=0}^{M-1} p_\infty^{(i)} = 1 \tag{5.14}$$

For the 2-state convolutional encoder with systematic generator matrix (5.9) and realized in OCF, we obtain the stationary metric distribution

$$\boldsymbol{p}_\infty^{\mathrm{T}} = \frac{1}{1-p+10p^2-20p^3+20p^4-8p^5} \times \begin{pmatrix} 1+7p-28p^2+66p^3-100p^4+96p^5-56p^6+16p^7 \\ -3p+16p^2-46p^3+80p^4-88p^5+56p^6-16p^7 \\ -3p+10p^2-20p^3+20p^4-8p^5 \\ -6p^2+26p^3-60p^4+80p^5-56p^6-16p^7 \\ -2p^2-6p^3+40p^4-72p^5+56p^6-16p^7 \end{pmatrix}$$

5.2 COMPUTING THE VECTOR OF AVERAGE INFORMATION WEIGHTS

In order to compute the (exact) bit error probability according to (5.8), it is necessary to determine $\boldsymbol{w}_t(\zeta = 0)$. In the following, we will derive a recurrent matrix equation for the average information weights and illustrate how to obtain its components using as an example the rate $R = 2/3$ memory $m = 1$ minimal encoder determined by (5.9).

The vector $\boldsymbol{w}_t$ describes the dynamics of the information weights when we proceed along the trellis, and hence satisfies the recurrent matrix equation

$$\begin{cases} \boldsymbol{w}_{t+1} = \boldsymbol{w}_t A + \boldsymbol{b}_t B \\ \boldsymbol{b}_{t+1} = \boldsymbol{b}_t \Pi \end{cases} \tag{5.15}$$

where A and B are $M\,|\Sigma| \times M\,|\Sigma|$ nonnegative matrices, and Π is an $M\,|\Sigma| \times M\,|\Sigma|$ stochastic matrix, with $|\Sigma| = 2^m$. Both matrices A and B have a $|\Sigma| \times |\Sigma|$ block structure, containing the submatrices A_{ij} and B_{ij} of size $M \times M$, respectively, where the former satisfy

$$\sum_{i=0}^{|\Sigma|-1} A_{ij} = \Phi, \;\; j = 0, 1, \ldots, |\Sigma| - 1 \tag{5.16}$$

since we consider only encoders for which every encoder state is reachable with probability 1.

The matrix A represents the linear part of the affine transformation of the information weights while the matrix B describes their weight increments. That is, their submatrices A_{ij} and B_{ij} specify the updating of the average information weights if the transition from encoder state i to encoder state j exists; and are zero otherwise. Moreover, the vector $\boldsymbol{b}_t$ of length $M\,|\Sigma|$ is a concatenation of the $|\Sigma|$ stochastic vectors $\boldsymbol{p}_t$, and hence the matrix Π is determined by

$$\Pi = \begin{pmatrix} \Phi & 0 & \ldots & 0 \\ 0 & \Phi & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \Phi \end{pmatrix} \tag{5.17}$$

From the initial formula for the exact bit error probability (5.1), it follows that we are interested in the asymptotic values. Hence, we can choose the initial value of the vector of information weights at time instant 0 as

$$\boldsymbol{w}_0 = \boldsymbol{0} \tag{5.18}$$

Continuing the previous example, we will illustrate how the 10×10 matrices A and B can be obtained directly from all 40 trellis sections. For example,

the eight trellis sections shown in Figure 5.2 determine all transitions from $\phi_t = 0$ to either $\phi_{t+1} = -1$ or $\phi_{t+1} = 1$.

To be more precise, consider all transitions from $\zeta_t = 0$ and $\phi_t = 0$ to $\zeta_{t+1} = 0$ and $\phi_{t+1} = -1$, as shown in Figure 5.2(a), (b), (g), and (h). Only Figure 5.2(a) and (g) have transitions decided by the Viterbi algorithm, which are $\boldsymbol{v}_t = 000$ and $\boldsymbol{v}_t = 110$, respectively, and thus the entry $\zeta_t = 0$, $\phi_t = 0$, $\zeta_{t+1} = 0$, $\phi_{t+1} = -1$ within matrix A follows as

$$P(\boldsymbol{r}_t = 000) + P(\boldsymbol{r}_t = 110) = q^3 + p^2 q$$

and within matrix B as

$$\beta(000)\, P(\boldsymbol{r}_t = 000) + \beta(110)\, P(\boldsymbol{r}_t = 110) = 0 + 2p^2 q = 2p^2 q$$

where $\beta(\boldsymbol{v}_t)$ denotes the number of information 1s corresponding to $\boldsymbol{v}_t$.

Since we use coin-flipping to resolve ties, we obtain from the trellis sections in Figure 5.2(c) and (d) that the entry $\zeta_t = 0$, $\phi_t = 0$, $\zeta_{t+1} = 0$, $\phi_{t+1} = 1$ within matrix A is

$$\begin{aligned}
&\frac{1}{2}P(\boldsymbol{r}_t = 010) + \frac{1}{2}P(\boldsymbol{r}_t = 010) + \frac{1}{2}P(\boldsymbol{r}_t = 100) + \frac{1}{2}P(\boldsymbol{r}_t = 100) \\
&\quad = \frac{1}{2}pq^2 + \frac{1}{2}pq^2 + \frac{1}{2}pq^2 + \frac{1}{2}pq^2 = 2pq^2
\end{aligned}$$

while for matrix B we obtain

$$\begin{aligned}
&\frac{1}{2}\beta(000)\, P(\boldsymbol{r}_t = 010) + \frac{1}{2}\beta(110)\, P(\boldsymbol{r}_t = 010) \\
&+\frac{1}{2}\beta(000)\, P(\boldsymbol{r}_t = 100) + \frac{1}{2}\beta(110)\, P(\boldsymbol{r}_t = 100) \\
&\quad = \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 2pq^2 + \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 2pq^2 = 2pq^2
\end{aligned}$$

Similarly the entry $\zeta_t = 1$, $\phi_t = 0$, $\zeta_{t+1} = 0$, $\phi_{t+1} = -1$ follows from Figure 5.2(b) and (h) in matrix A as

$$pq^2 + p^3$$

and in matrix B as

$$0 + 2p^3 = 2p^3$$

Finally, according to Figure 5.2(e) and (f), the entry $\zeta_t = 1$, $\phi_t = 0$, $\zeta_{t+1} = 0$, $\phi_{t+1} = 1$ in matrix A is given by

$$\frac{1}{2}p^2 q + \frac{1}{2}p^2 q + \frac{1}{2}p^2 q + \frac{1}{2}p^2 q = 2p^2 q$$

and in matrix B by

$$\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 2p^2q + \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 2p^2q = 2p^2q$$

The trellis sections in Figure 5.2 additionally determine the entries for the transitions $\zeta_t = 0$, $\phi_t = 0$, $\zeta_{t+1} = 1$, $\phi_{t+1} = -1$ and $\zeta_t = 0$, $\phi_t = 0$, $\zeta_{t+1} = 1$, $\phi_{t+1} = 1$ as well as the transitions $\zeta_t = 1$, $\phi_t = 0$, $\zeta_{t+1} = 1$, $\phi_{t+1} = -1$ and $\zeta_t = 1$, $\phi_t = 0$, $\zeta_{t+1} = 1$, $\phi_{t+1} = 1$.

All other transitions with $\phi_t = 0$ are never decided by the Viterbi algorithm, and hence the remaining entries in the corresponding rows within the matrices A and B are zero. The eight trellis sections in Figure 5.2 yield 20 out of 100 entries in each of the two matrices A and B, while the 32 trellis sections not shown in Figure 5.2 yield the remaining 80 entries for each matrix. In summary, for the convolutional encoder shown in Figure 5.1, we obtain the matrices

$$A = \begin{array}{c} \\ \zeta_t = 0 \\ \zeta_t = 1 \end{array} \begin{array}{c} \begin{array}{cc} \zeta_{t+1} = 0 & \zeta_{t+1} = 1 \end{array} \\ \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \end{array} \tag{5.19}$$

where

$$A_{00} = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} q^3+p^2q & 0 & p^3+3pq^2 & 0 & 2p^2q \\ q^3+p^2q & 0 & \frac{1}{2}p^3+\frac{5}{2}pq^2 & 0 & p^2q \\ 0 & q^3+p^2q & 0 & 2pq^2 & 0 \\ 0 & 0 & \frac{1}{2}q^3+\frac{1}{2}p^2q & 0 & pq^2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.20}$$

$$A_{01} = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} p^2q+q^3 & 0 & p^3+3pq^2 & 0 & 2p^2q \\ \frac{1}{2}p^3+\frac{1}{2}pq^2 & 0 & p^2q & 0 & 0 \\ 0 & p^3+pq^2 & 0 & 2p^2q & 0 \\ \frac{1}{2}q^3+\frac{1}{2}p^2q & 0 & p^3+2pq^2 & 0 & 2p^2q \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.21}$$

$$A_{10} = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2}p^3+\frac{1}{2}pq^2 & 0 & p^2q \\ 0 & p^3+pq^2 & 0 & 2p^2q & 0 \\ p^3+pq^2 & 0 & \frac{1}{2}q^3+\frac{5}{2}p^2q & 0 & pq^2 \\ p^3+pq^2 & 0 & 3p^2q+q^3 & 0 & 2pq^2 \end{pmatrix} \end{array} \tag{5.22}$$

$$A_{11} = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}p^2q+\frac{1}{2}q^3 & 0 & pq^2 & 0 & 0 \\ 0 & p^2q+q^3 & 0 & 2pq^2 & 0 \\ \frac{1}{2}pq^2+\frac{1}{2}p^3 & 0 & 2p^2q+q^3 & 0 & 2pq^2 \\ p^3+pq^2 & 0 & 3p^2q+q^3 & 0 & 2pq^2 \end{pmatrix} \end{array} \tag{5.23}$$

and

$$B = \begin{array}{c} \\ \zeta_t = 0 \\ \zeta_t = 1 \end{array} \begin{array}{c} \begin{array}{cc} \zeta_{t+1} = 0 & \zeta_{t+1} = 1 \end{array} \\ \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \end{array} \tag{5.24}$$

where

$$B_{00} = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 2p^2q & 0 & p^3+2pq^2 & 0 & 2p^2q \\ 0 & 0 & p^2q & 0 & pq^2 \\ 0 & 2p^2q & 0 & 2pq^2 & 0 \\ 2p^2q & 0 & p^3+2pq^2 & 0 & p^2q \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.25}$$

$$B_{01} = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} p^2q+q^3 & 0 & p^3+3pq^2 & 0 & 2p^2q \\ \frac{1}{2}p^3+\frac{1}{2}pq^2 & 0 & p^2q & 0 & 0 \\ 0 & p^3+pq^2 & 0 & 2p^2q & 0 \\ \frac{1}{2}q^3+\frac{1}{2}p^2q & 0 & p^3+2pq^2 & 0 & 2p^2q \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.26}$$

$$B_{10} = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p^3 & 0 & p^2q \\ 0 & 2p^3 & 0 & 2p^2q & 0 \\ 2p^3 & 0 & 3p^2q & 0 & pq^2 \\ 2p^3 & 0 & 4p^2q & 0 & 2pq^2 \end{pmatrix} \end{array} \tag{5.27}$$

$$B_{11} = \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}p^2q+\frac{1}{2}q^3 & 0 & pq^2 & 0 & 0 \\ 0 & p^2q+q^3 & 0 & 2pq^2 & 0 \\ \frac{1}{2}pq^2+\frac{1}{2}p^3 & 0 & 2p^2q+q^3 & 0 & 2pq^2 \\ p^3+pq^2 & 0 & 3p^2q+q^3 & 0 & 2pq^2 \end{pmatrix} \end{array} \tag{5.28}$$

5.3 SOLVING THE RECURRENT EQUATION

Consider the second equation in (5.15). It follows from (5.8) that we are only interested in the asymptotic values, and hence letting t tend to infinity yields

$$\boldsymbol{b}_\infty = \boldsymbol{b}_\infty \Pi \tag{5.29}$$

where $\boldsymbol{b}_\infty$ can be chosen as

$$\boldsymbol{b}_\infty = (\boldsymbol{p}_\infty \, \boldsymbol{p}_\infty \cdots \boldsymbol{p}_\infty) \tag{5.30}$$

To obtain the last equality, we took into account that Π is a block-diagonal matrix whose diagonal elements are given by the transition probability matrix Φ which satisfies (5.13). Based on these observations, (5.15) is simplified to

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t A + \boldsymbol{b}_\infty B \tag{5.31}$$

By iterating this simplified recurrent matrix equation together with its initial value from (5.18), we obtain the vector of the information weights at time instant $t+1$ as

$$\boldsymbol{w}_{t+1} = \boldsymbol{b}_\infty B A^t + \boldsymbol{b}_\infty B A^{t-1} + \cdots + \boldsymbol{b}_\infty B \tag{5.32}$$

Before continuing, we would like to recall the following theorem on the convergence of the arithmetic mean of a sequence.

Theorem 5.1 (Cesàro mean [Har92]) Consider the sequences $\langle a_n \rangle$ and $\langle c_n \rangle$, where

$$c_n = \frac{1}{n} \sum_{i=1}^{n} a_i \tag{5.33}$$

is the arithmetic mean of the first n elements of $\langle a_n \rangle$. If the sequence $\langle a_n \rangle$ converges, then

$$\lim_{n\to\infty} c_n = \lim_{n\to\infty} a_n = a_\infty \tag{5.34}$$

Hence, taking the limit $t \to \infty$, it follows from (5.32) that

$$\lim_{t\to\infty} \frac{\boldsymbol{w}_t}{tb} = \lim_{t\to\infty} \frac{\boldsymbol{w}_{t+1}}{tb} = \lim_{t\to\infty} \frac{1}{tb} \sum_{j=0}^{t} \boldsymbol{b}_\infty B A^{t-j} = \frac{1}{b} \boldsymbol{b}_\infty B A^\infty \tag{5.35}$$

where A^∞ denotes the limit of the sequence $\langle A^t \rangle$ when t tends to infinity. Moreover, it follows from (5.16) and (5.17) that the vector

$$\boldsymbol{e}_{\mathrm{L}} = (\boldsymbol{p}_\infty \, \boldsymbol{p}_\infty \cdots \boldsymbol{p}_\infty) \tag{5.36}$$

satisfies

$$\boldsymbol{e}_{\mathrm{L}} A = \boldsymbol{e}_{\mathrm{L}} \tag{5.37}$$

and hence is a left eigenvector with eigenvalue $\lambda = 1$ for the matrix A. Similarly, let $\boldsymbol{e}_{\mathrm{R}}$ denote a corresponding right eigenvector, normalized such that $\boldsymbol{e}_{\mathrm{L}} \boldsymbol{e}_{\mathrm{R}}^{\mathrm{T}} = 1$.

Based on these preliminaries, we will prove in the following subsection that the eigenvalue $\lambda = 1$ is a maximal and simple eigenvalue for the matrix A, and hence the limit of the power series of A is given by

$$\lim_{m\to\infty} A^m = A^\infty = \boldsymbol{e}_{\mathrm{L}}^{\mathrm{T}} \boldsymbol{e}_{\mathrm{R}} \tag{5.38}$$

5.3.1 DETERMINING THE LIMIT OF A POWER SERIES

First, we will recall two basic Lemmas and one Corollary from matrix theory, which shall be useful later:

Corollary 5.2 (Corollary 8.1.30 in [HJ90]) Suppose that A is a nonnegative square matrix with a positive left eigenvector $\boldsymbol{e}_{\mathrm{L}} > 0$ and spectral radius $\rho(A)$. Then the corresponding eigenvalue is $\rho(A)$, that is, $|\lambda| < \rho(A)$ for any other eigenvalue λ.

Lemma 5.3 (Lemma 8.2.7, statement (i) in [HJ90]) Let A be a square matrix with eigenvalue $\lambda \neq 0$ and denote the corresponding left and right eigenvectors by $\boldsymbol{e}_{\mathrm{L}}$ and $\boldsymbol{e}_{\mathrm{R}}$, such that $\boldsymbol{e}_{\mathrm{L}} \boldsymbol{e}_{\mathrm{R}}^{\mathrm{T}} = 1$. If the eigenvalue λ has geometric multiplicity 1, it is the only eigenvalue with modulus $\rho(A)$, and satisfies $|\lambda| = \rho(A) > 0$, where $\rho(A)$ denotes the spectral radius of A, and

$$\lim_{m\to\infty} \left(\lambda^{-1} A\right)^m = e_{\mathrm{R}}^{\mathrm{T}} e_{\mathrm{L}} \tag{5.39}$$

Lemma 5.4 (Lemma 8.4.3 in [HJ90]) If A is a nonnegative square matrix and $A^k > 0$ for some $k \geq 1$, then $\rho(A)$ is an algebraically simple eigenvalue of A, where $\rho(A)$ denotes the spectral radius of A.

Depending on the ML decisions of the Viterbi decoder, there might be a state ζ with normalized cumulative metric $\boldsymbol{\phi}_t$, whose leaving transitions are never decided by the Viterbi algorithm. Clearly, such a state corresponds to an all-zero row in the matrix A, and hence does not contribute to the bit error probability and can be ignored. Additionally, all transitions arriving at such a state can be ignored, since those transitions can never be decided by the Viterbi algorithm during the following time instant. Hence, without loss of generality, all rows with only zero elements, as well as their corresponding columns can be removed. Denote the remaining matrix by $\widehat{A}$.

Note that due to those all-zero rows within the matrix A, the corresponding elements in the left eigenvector $\boldsymbol{e}_{\mathrm{L}}$ have to be zero. Hence, removing those zero elements, yields the reduced vector $\widehat{\boldsymbol{e}}_{\mathrm{L}}$ which satisfies

$$\widehat{\boldsymbol{e}}_{\mathrm{L}} \widehat{A} = \widehat{\boldsymbol{e}}_{\mathrm{L}} \tag{5.40}$$

and hence is a left eigenvector of the matrix $\widehat{A}$ with eigenvalue $\lambda = 1$.

For example, in case of the 2-state convolutional encoder with generator matrix (5.9), such states which are never decided by the Viterbi algorithm are given by $\zeta = 0$ with normalized cumulative metric $\phi = 2$ as well as by $\zeta = 1$ with normalized cumulative metric $\phi = -2$ (cf. submatrices (5.20) – (5.23)). That is, the corresponding matrix A can be written as

$$
A = \begin{array}{cc}
 & \begin{array}{cccccccccc} \multicolumn{5}{c}{\zeta_{t+1} = 0} & \multicolumn{5}{c}{\zeta_{t+1} = 1} \\ -2 & -1 & 0 & 1 & 2 & -2 & -1 & 0 & 1 & 2 \end{array} \\
\begin{array}{c} \\ \\ \zeta_t = 0 \\ \\ \\ \\ \\ \zeta_t = 1 \\ \\ \\ \end{array} &
\left(\begin{array}{cccccc|cccccc}
 & & & & * & * & & & & \\
 & & & & \vdots & \vdots & & & & \\
 & & & & * & * & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & * & * & & & & \\
 & & & & \vdots & \vdots & & & & \\
 & & & & * & * & & & &
\end{array}\right)
\end{array}
\begin{array}{c} -2 \\ -1 \\ 0 \\ 1 \\ 2 \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array}
\tag{5.41}
$$

where the all-zero rows and their corresponding columns with arbitrary entries are shaded in gray. After removing these rows and columns, we obtain the 8×8 nonnegative matrix $\widehat{A}$.

In particular, note that every state ζ within $\widehat{A}$ is reachable from any other state, and hence it follows that

$$\widehat{A} > 0 \quad \text{for some } \; k \geq 1 \tag{5.42}$$

Moreover, since the entries of the corresponding left eigenvector $\widehat{\boldsymbol{e}}_{\mathrm{L}}$ represent the probabilities of the stationary distribution of the Markov Chain where each state is reachable with probability 1, the eigenvector $\widehat{\boldsymbol{e}}_{\mathrm{L}}$ has to be strictly positive.

Thus, it follows from Corollary 5.2 and Lemma 5.4 that $\lambda = 1$ is a simple and maximal eigenvalue for the reduced matrix $\widehat{A}$ with left eigenvector $\widehat{\boldsymbol{e}}_{\mathrm{L}}$.

Finally, we will prove that the same result is valid for the eigenvalue $\lambda = 1$ of the matrix A with eigenvector $\boldsymbol{e}_{\mathrm{L}}$. Therefore, we consider the matrix $\widehat{A}'$, obtained by extending the matrix $\widehat{A}$ with an additional all-zero row and an additional column with arbitrarily chosen elements represented by $*$

$$\widehat{A}' = \begin{pmatrix} & & & * \\ & \widehat{A} & & * \\ & & & * \\ 0 & \dots & 0 & 0 \end{pmatrix} \tag{5.43}$$

Since the characteristic polynomials of $\widehat{A}$ and $\widehat{A}'$ satisfy

$$\det\left(\lambda I - \widehat{A}'\right) = \lambda \det\left(\lambda I - \widehat{A}\right) \tag{5.44}$$

it follows that $\widehat{A}'$ and $\widehat{A}$ share the same eigenvalues, except the additional eigenvalue $\lambda = 0$ for $\widehat{A}'$.

By induction we conclude that the eigenvalues of the matrices A and $\widehat{A}$ differ at most by (the multiplicity of) the eigenvalue $\lambda = 0$. In particular, it follows that the eigenvalue $\lambda = \rho(A) = 1$ for the matrix A is a simple and maximal eigenvalue with left eigenvector $\boldsymbol{e}_{\mathrm{L}}$.

Hence, the assumptions of Lemma 5.3 are satisfied and it follows that

$$A^{\infty} = \boldsymbol{e}_{\mathrm{R}}^{\mathrm{T}} \boldsymbol{e}_{\mathrm{L}} \tag{5.45}$$

5.3.2 DERIVING A CLOSED FORM EXPRESSION

Combining (5.35), (5.36), and (5.45), the limit of the recurrent matrix equation can be expressed as

$$\lim_{t\to\infty} \frac{\boldsymbol{w}_t}{tb} = \frac{1}{b} \boldsymbol{b}_{\infty} B \boldsymbol{e}_{\mathrm{R}}^{\mathrm{T}} \boldsymbol{e}_{\mathrm{L}} = \frac{1}{b} \boldsymbol{b}_{\infty} B \boldsymbol{e}_{\mathrm{R}}^{\mathrm{T}} \left(\boldsymbol{p}_{\infty}\, \boldsymbol{p}_{\infty} \cdots \boldsymbol{p}_{\infty} \right) \tag{5.46}$$

Following (5.8), we obtain the closed form expression for the exact bit error probability by summing up the first M components of the vector $(\boldsymbol{p}_{\infty} \boldsymbol{p}_{\infty} \cdots \boldsymbol{p}_{\infty})$ on the right side of (5.46), and hence

$$P_{\mathrm{b}} = \frac{1}{b} \boldsymbol{b}_{\infty} B \boldsymbol{e}_{\mathrm{R}}^{\mathrm{T}} \tag{5.47}$$

To summarize, the exact bit error probability P_{b} for Viterbi decoding of a rate $R = b/c$ minimal convolutional encoder, when communicating over a BSC, is calculated as follows:

(i) Construct the set of normalized cumulative (Viterbi) metrics and find the corresponding stationary probability distribution $\boldsymbol{p}_{\infty}$ of its metric state Markov chain.

(ii) Determine the matrices A and B as in Section 5.1 and compute the right eigenvector $\boldsymbol{e}_{\mathrm{R}}$ normalized such that $(\boldsymbol{p}_{\infty}\ \ \boldsymbol{p}_{\infty} \cdots \boldsymbol{p}_{\infty}) \boldsymbol{e}_{\mathrm{R}}^{\mathrm{T}} = 1$.

(iii) Calculate the exact bit error probability P_{b} using (5.47).

For the encoder shown in Figure 5.1 we obtain

$$\begin{aligned}
P_{\mathrm{b}} = {} & \Big(4p - 2p^2 + 67p^3 - 320p^4 + 818p^5 - 936p^6 - 884p^7 + 5592p^8 \\
& -11232p^9 + 13680p^{10} - 11008p^{11} + 5760p^{12} - 1792p^{13} + 256p^{14}\Big) / \\
& \Big(2 - 5p + 41p^2 - 128p^3 + 360p^4 - 892p^5 + 1600p^6 \\
& -1904p^7 + 1440p^8 - 640p^9 + 128p^{10}\Big) \\
= {} & 2p + 4p^2 + \frac{5}{2}p^3 - \frac{431}{4}p^4 - \frac{125}{8}p^5 + \frac{32541}{16}p^6 - \frac{70373}{32}p^7 \\
& - \frac{1675587}{64}p^8 + \frac{7590667}{128}p^9 + \frac{67672493}{256}p^{10} - \cdots
\end{aligned} \tag{5.48}$$

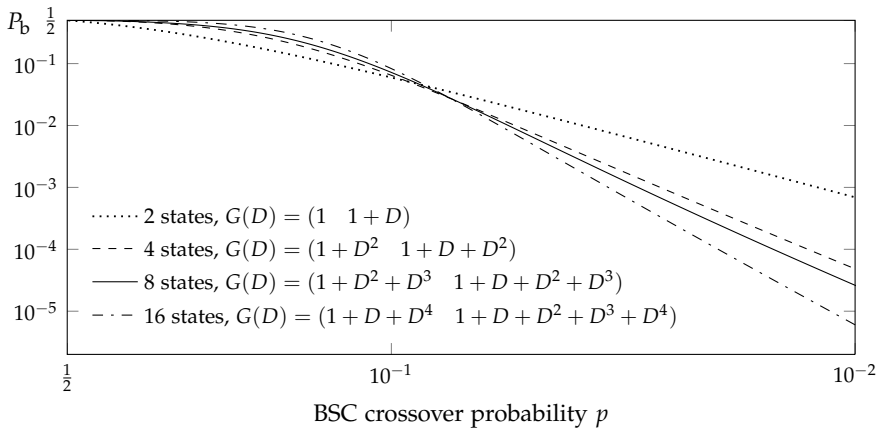


Figure 5.4: Exact bit error probability for the rate $R = 1/2$ minimal encoders of memory 1 (2-state), memory 2 (4-state), memory 3 (8-state), and memory 4 (16-state).

If the minimal generator matrix (5.9) would be instead realized in CCF, we obtain a nonminimal (4-state) encoder with $M = 12$ normalized cumulative metrics, whose exact bit error probability is slightly worse than that of its minimal realization in OCF (*cf.*, the note after (5.1)).

5.4 ADDITIONAL EXAMPLES

In the following section, the exact bit error probability shall be calculated for various convolutional codes, realized in either CCF or OCF, and being used to communicate over a BSC or a quantized AWGN channel.

As our first example, we consider four different rate $R = 1/2$ convolutional encoders with memory $m = 1$, 2, 3, and 4; that is, encoders with 2, 4, 8, and 16 states, realized in CCF. Their corresponding exact bit error probability is plotted in Figure 5.4.

■ **Example 5.1:**
Consider the rate $R = 1/2$, memory $m = 1$ (2-state) convolutional encoder with generator matrix $G(D) = (1 \quad 1+D)$. Its CCF realization yields 20 different trellis sections with the normalized cumulative metrics $\{-2, -1, 0, 1, 2\}$, while the stationary probability distribution of its metric state Markov chain is given by

$$\boldsymbol{p}_\infty^{\mathrm{T}} = \frac{1}{1+3p^2-2p^3} \begin{pmatrix} 1-4p+8p^2-7p^3+2p^4 \\ 2p-5p^2+5p^3-2p^4 \\ 2p-3p^2+3p^3 \\ 2p^2-3p^3+2p^4 \\ p^2+p^3-2p^4 \end{pmatrix} \tag{5.49}$$

Based on these 20 trellis sections, the 10×10 matrices A and B are constructed as the following block matrices

$$A = \begin{array}{c} \\ \zeta_t = 0 \\ \zeta_t = 1 \end{array} \begin{array}{c} \begin{array}{cc} \zeta_{t+1} = 0 & \zeta_{t+1} = 1 \end{array} \\ \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \end{array} \tag{5.50}$$

$$= \begin{array}{c} \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc|ccccc} -2 & -1 & 0 & 1 & 2 & -2 & -1 & 0 & 1 & 2 \end{array} \\ \left(\begin{array}{ccccc|ccccc} q^2 & 0 & 2pq & 0 & p^2 & q^2 & 0 & 2pq & 0 & p^2 \\ q^2 & 0 & 3pq/2 & 0 & p^2/2 & q^2/2 & 0 & 3pq/2 & 0 & p^2 \\ 0 & q^2 & 0 & pq & 0 & 0 & pq & 0 & p^2 & 0 \\ 0 & 0 & q^2/2 & 0 & pq/2 & pq/2 & 0 & p^2/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & pq/2 & 0 & p^2/2 & q^2/2 & 0 & pq/2 & 0 & 0 \\ 0 & pq & 0 & p^2 & 0 & 0 & q^2 & 0 & pq & 0 \\ pq & 0 & q^2/2 + p^2 & 0 & pq/2 & pq/2 & 0 & q^2 + p^2/2 & 0 & pq \\ pq & 0 & q^2 + p^2 & 0 & pq & pq & 0 & q^2 + p^2 & 0 & pq \end{array}\right) \end{array}$$

and

$$B = \begin{array}{c} \\ \zeta_t = 0 \\ \zeta_t = 1 \end{array} \begin{array}{c} \begin{array}{cc} \zeta_{t+1} = 0 & \zeta_{t+1} = 1 \end{array} \\ \begin{pmatrix} \mathbf{0}_{5,5} & A_{01} \\ \mathbf{0}_{5,5} & A_{11} \end{pmatrix} \end{array} \tag{5.51}$$

where $\mathbf{0}_{5,5}$ denotes the 5×5 all-zero matrix. Moreover, we obtain the corresponding normalized right eigenvector of A as

$$\boldsymbol{e}_{\mathrm{R}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \frac{pq}{2} & \frac{4pq}{2-p+4p^2-4p^3} & \frac{(2+7p-12p^2+13p^3-12p^4+4p^5)}{2(2-p+4p^2-4p^3)} & 1 \end{pmatrix}$$

Inserting (5.49), (5.51), and (5.52) into (5.47) yields the following expression for the exact bit error probability

$$\begin{aligned} P_{\mathrm{b}} &= \frac{14p^2 - 23p^3 + 16p^4 + 2p^5 - 16p^6 + 8p^7}{(1 + 3p^2 - 2p^3)(2 - p + 4p^2 - 4p^3)} \\ &= 7p^2 - 8p^3 - 31p^4 + 64p^5 + 86p^6 - \frac{635}{2}p^7 \\ &\quad - \frac{511}{4}p^8 + \frac{10165}{8}p^9 - \frac{4963}{16}p^{10} - \cdots \end{aligned} \tag{5.52}$$

which coincides with the exact bit error probability formula given in [BBL+95].

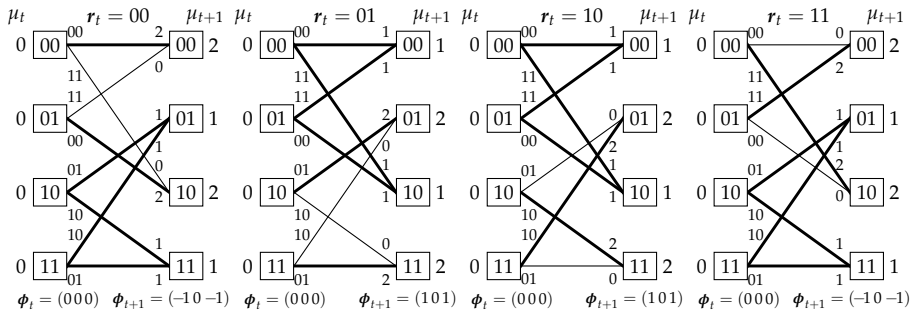


Figure 5.5: Four different trellis sections of the in total 124 for the $G(D) = (1 + D^2 \quad 1 + D + D^2)$ generator matrix.

■ **Example 5.2:**
For the rate $R = 1/2$, memory $m = 2$ (4-state) convolutional encoder with generator matrix $G(D) = (1 + D^2 \quad 1 + D + D^2)$ and realized in CCF, we obtain in total 124 different trellis sections, for example, the four trellis sections for $\phi_t = (000)$ plotted in Figure 5.5 whose corresponding normalized cumulative metrics at time instant $t + 1$ are $\phi_{t+1} = (-1\,0\ -1)$ and $\phi_{t+1} = (1\,0\,1)$.

Completing the set of trellis sections yields $M = 31$ different normalized cumulative metrics, and hence the 124×124 matrices A and B have the block structure

$$A = \begin{pmatrix} A_{00} & \mathbf{0}_{31,31} & A_{02} & \mathbf{0}_{31,31} \\ A_{10} & \mathbf{0}_{31,31} & A_{12} & \mathbf{0}_{31,31} \\ \mathbf{0}_{31,31} & A_{21} & \mathbf{0}_{31,31} & A_{23} \\ \mathbf{0}_{31,31} & A_{31} & \mathbf{0}_{31,31} & A_{33} \end{pmatrix} \tag{5.53}$$

and

$$B = \begin{pmatrix} \mathbf{0}_{31,31} & \mathbf{0}_{31,31} & A_{02} & \mathbf{0}_{31,31} \\ \mathbf{0}_{31,31} & \mathbf{0}_{31,31} & A_{12} & \mathbf{0}_{31,31} \\ \mathbf{0}_{31,31} & \mathbf{0}_{31,31} & \mathbf{0}_{31,31} & A_{23} \\ \mathbf{0}_{31,31} & \mathbf{0}_{31,31} & \mathbf{0}_{31,31} & A_{33} \end{pmatrix} \tag{5.54}$$

Following the method for calculating the exact bit error probability described in Section 5.3 we obtain

$$\begin{aligned} P_\mathrm{b} = {} & 44p^3 + \frac{3519}{8}p^4 - \frac{14351}{32}p^5 - \frac{1267079}{64}p^6 - \frac{31646405}{512}p^7 \\ & + \frac{978265739}{2048}p^8 + \frac{3931764263}{1024}p^9 - \frac{48978857681}{32768}p^{10} - \cdots \end{aligned} \tag{5.55}$$

which coincides with the previously obtained result given in [LTZ04].

■ **Example 5.3:**
For the rate $R = 1/2$, memory $m = 3$ (8-state) convolutional encoder with generator matrix $G(D) = (1 + D^2 + D^3 \quad 1 + D + D^2 + D^3)$ and realized in CCF, we have $M = 433$ normalized cumulative metrics yielding the matrices A and B of size $433 \cdot 2^3 \times 433 \cdot 2^3$.

Since the the complexity of the symbolic derivations increases greatly, we can only obtain a numerical solution of (5.47) as shown in Figure 5.4.

■ **Example 5.4:**
For the rate $R = 1/2$, memory $m = 4$ (16-state) convolutional encoder with generator matrix $G(D) = (1 + D^2 + D^3 + D^4 \quad 1 + D + D^4)$ and realized in CCF, we have as many as $M = 188687$ normalized cumulative metrics. Thus, the matrices A and B are of size $188687 \cdot 2^4 \times 188687 \cdot 2^4$.

The corresponding numerical solution of (5.47) is given in Figure 5.4.

The obvious next step would be to try a rate $R = 1/2$, memory $m = 5$ (32-state) convolutional encoder. We considered the generator matrix $G(D) = (1 + D + D^2 + D^3 + D^4 + D^5 \quad 1 + D^3 + D^5)$ realized in CCF, but we were only able to show that its number of normalized cumulative metrics M exceeds 4130000.

■ **Example 5.5:**
Next, consider the generator matrix

$$G_1(D) = \begin{pmatrix} 1 + D^2 & 1 + D + D^2 \end{pmatrix} \tag{5.56}$$

and its equivalent systematic generator matrices

$$G_2(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix} \tag{5.57}$$

and

$$G_3(D) = \begin{pmatrix} 1 & \frac{1+D+D^2}{1+D^2} \end{pmatrix} \tag{5.58}$$

When realized in CCF all three realizations have $M = 31$ normalized cumulative metrics. The exact bit error probability for $G_1(D)$ is given by (5.55). For $G_2(D)$ and $G_3(D)$ we obtain

$$\begin{aligned} P_\mathrm{b} = {} & \frac{163}{2}p^3 + \frac{365}{2}p^4 - \frac{24045}{8}p^5 - \frac{1557571}{128}p^6 + \frac{23008183}{512}p^7 \\ & + \frac{1191386637}{2048}p^8 + \frac{4249634709}{8192}p^9 + \frac{132555764497}{8192}p^{10} - \cdots \end{aligned} \tag{5.59}$$

and

$$\begin{aligned} P_\mathrm{b} = {} & \frac{141}{2}p^3 + \frac{1739}{8}p^4 - \frac{71899}{32}p^5 - \frac{1717003}{128}p^6 + \frac{2635041}{128}p^7 \\ & + \frac{540374847}{1024}p^8 + \frac{9896230051}{8192}p^9 - \frac{402578056909}{32768}p^{10} - \cdots \end{aligned} \tag{5.60}$$

respectively. The corresponding numerical results are shown in Figure 5.6.

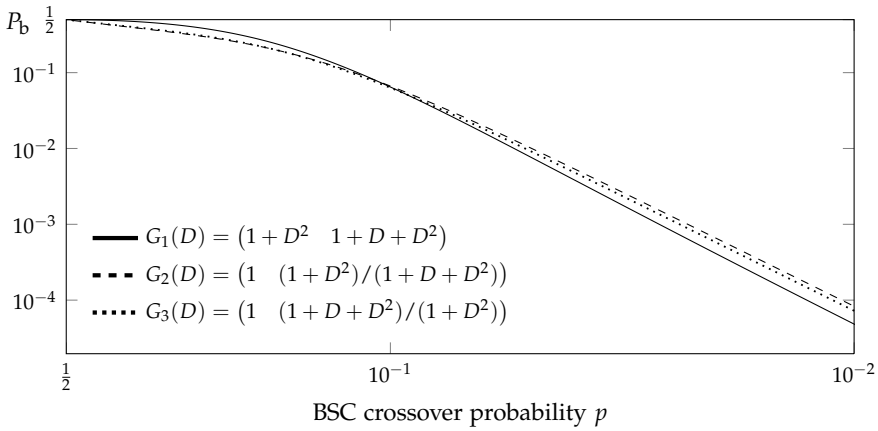


Figure 5.6: Exact bit error probability for rate $R = 1/2$, memory 2 minimal encoders.

■ **Example 5.6:**

The exact bit error probability for the rate $R = 2/3$ generator matrices with 4, 8, and 16 states, specified in Table 5.1 and realized in CCF, is plotted in Figure 5.7. For example, in case of the 4-state encoder we obtain the exact bit error probability

$$\begin{aligned} P_b = {} & \frac{67}{2}p^2 + \frac{17761}{48}p^3 - \frac{2147069}{648}p^4 - \frac{1055513863}{46656}p^5 + \frac{123829521991}{559872}p^6 \\ & + \frac{67343848419229}{60466176}p^7 - \frac{27081094434882419}{2176782336}p^8 \\ & - \frac{477727138796620247}{8707129344}p^9 + \frac{1944829319763332473469}{2821109907456}p^{10} + \cdots \end{aligned} \tag{5.61}$$

$G(D)$	#states	d_{free}	M
$\begin{pmatrix} D & 1+D & 1+D \\ 1 & D & 1+D \end{pmatrix}$	4	3	19
$\begin{pmatrix} 1+D & D & 1 \\ D^2 & 1 & 1+D+D^2 \end{pmatrix}$	8	4	347
$\begin{pmatrix} D+D^2 & 1 & 1+D^2 \\ 1 & D+D^2 & 1+D+D^2 \end{pmatrix}$	16	5	15867

Table 5.1: Rate $R = 2/3$ generator matrices.

If the BSC is replaced by a quantized AWGN channel, the calculation of the exact bit error probability follows the same method as described in Section 5.3, but its computational complexity increases dramatically as illustrated by the following example.

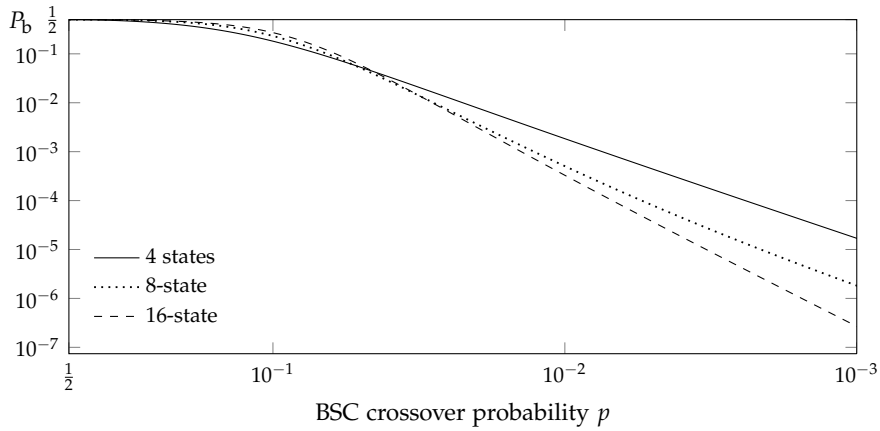


Figure 5.7: Exact bit error probability for the rate $R = 2/3$, overall constraint length $\nu = 2, 3$, and 4 (4-state, 8-state, and 16-state, respectively) encoders whose generator matrices are given in Table 5.1.

■ **Example 5.7:**

Consider the generator matrix $G(D) = (1 + D^2 \quad 1 + D + D^2)$ realized in CCF and used to communicate over a quantized AWGN channel. We use different quantization methods, namely, uniform quantization [HJ71] [OCC93] and Massey quantization [Mas74] [JZ99]; see Figure 5.8.

In both cases we optimized the cut-off rate R_0 to determine the quantization thresholds, but allowed nonuniform quantization intervals only for the latter case. When using uniform quantization with 7, 8, and 9 quantization levels, we obtained the same number of normalized cumulative metrics for all SNRs, namely $M = 1013$, $M = 2143$, and $M = 2281$, respectively. On the other hand,

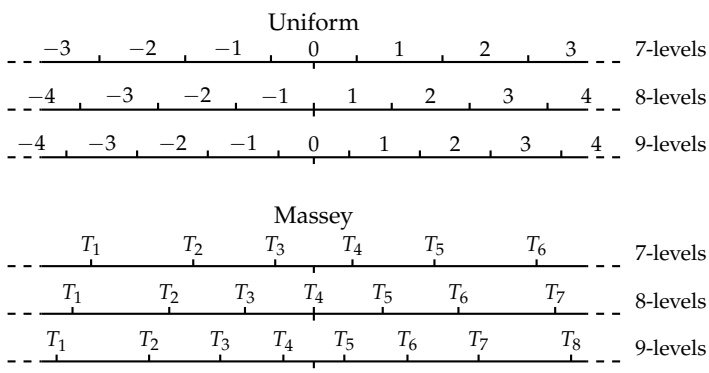


Figure 5.8: Examples of uniform and Massey quantizations for an AWGN channel with signal to noise ratio SNR = 0 dB.

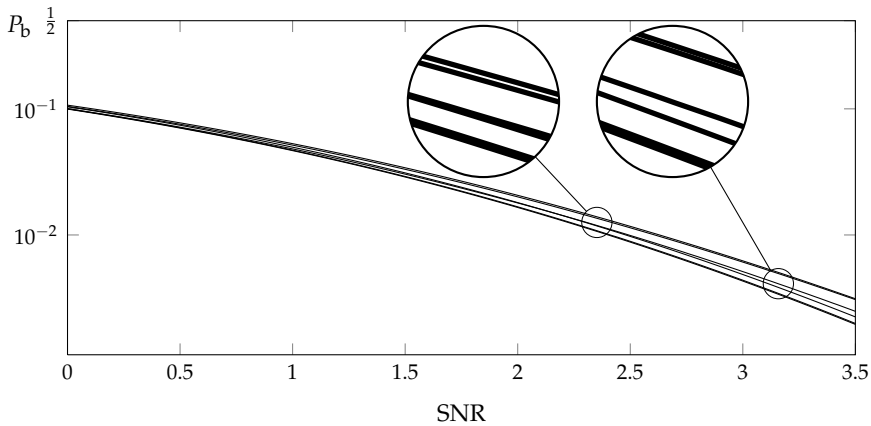


Figure 5.9: Exact bit error probability for the rate $R = 1/2$, memory 2 (4-state) encoder with $G(D) = (1 + D^2 \;\; 1 + D + D^2)$ used to communicate over an quantized AWGN channel with different quantization levels.

for Massey quantization, the number of normalized cumulative metrics varies with both the number of quantization levels and the chosen SNR. Moreover, these numbers are much higher. For example, considering the SNR interval between 0 dB and 3.5 dB with 8 quantization levels, we have $M = 16639$ normalized cumulative metrics for $E_b/N_0 \leq 2.43$ dB, while for $E_b/N_0 > 2.43$ dB we obtain $M = 17019$.

The exact bit error probability for this 4-state encoder is plotted for all different quantizations in Figure 5.9, ordered from worst (top) to best (bottom):

(i) Uniform quantization with 8 levels
(ii) Uniform quantization with 7 levels
(iii) Massey quantization with 7 levels
(iv) Uniform quantization with 9 levels
(v) Massey quantization with 8 levels
(vi) Massey quantization with 9 levels

All differences are very small, and hence it is hard to distinguish all curves. It is interesting to note that using 7 instead of 8 uniform quantization levels yields a better bit error probability. However, this is not surprising since the presence of a quantization bin around zero typically improves the quantization performance. Moreover, the number of normalized cumulative metrics for 7 quantization levels is only about one half of that for 8 quantization levels. Note that such a subtle comparison of channel output quantizers has only become possible due to the closed form expression for the exact bit error probability.

6

MacWilliams-type Identities for Convolutional Codes

Convolutional codes are often thought of as nonblock linear codes over a finite field. Sometimes, however, it is an advantage to regard convolutional codes as block codes over certain infinite fields; that is, as the $\mathbb{F}_2((D))$ row space of the generator matrix $G(D)$ or, in other words, as a rate $R = b/c$ block code over the infinite field of Laurent series encoded by $G(D)$. From this point of view it seems rather natural that convolutional codes would have similar properties as block codes and satisfy proper reformulations of theorems valid for block codes.

It is well-known, starting with the paper by Shearer and McEliece [SM77], that MacWilliams identity [MS77] does not hold for the Viterbi spectra of convolutional encoders. In [AG92], [GLS08], and [GLS09], MacWilliams-type identities were established, not for the Viterbi spectrum but for the so-called *weight adjacency matrix* (WAM) [McE98]. A MacWilliams-type identity with respect to WAMs for the encoders of an arbitrary convolutional code and its dual was formulated in [GLS08] and proved in [GLS09] by Gluesing-Luerssen and Schneider. Their work inspired Forney, and in [For09] and [For11] he proved their results in terms of the »constraint« code corresponding to each node of the trellis diagram and its dual. Moreover, he generalized them to the complete WAM as well as to group codes defined on graphs.

In Section 6.1, the concept of weakly equivalent convolutional generator matrices is introduced and it is shown that such matrices yield the same tail-biting block code spectrum. A similar comparison between the spectra of zero-tail terminated and truncated codes as well as the Viterbi spectra of their parent convolutional codes is presented in Section 6.2. The MacWilliams identity and various notations of duality for convolutional codes are revisited in Section 6.3. In particular, it is shown that MacWilliams identity holds for

properly truncated convolutional codes and their duals. Finally, it is proven in Section 6.4, that the spectra of truncated or tailbitten convolutional codes together with the corresponding spectra of their duals satisfy recursions of an order less than or equal to the rank r of the WAM of the minimal encoder of the convolutional code. In other words, it is enough to know $2r$ consecutive spectra of block codes obtained by truncating or tailbiting a convolutional code at lengths $c, 2c, \ldots, 2rc$, in order to find the infinite sequence of spectra of block codes which are terminations of the corresponding dual and vice-versa.

6.1 WEAKLY EQUIVALENT MATRICES

Consider the code trellis of a rate $R = b/c$ minimal convolutional encoder with memory m, terminated by either truncation, zero-tail termination or tailbiting. When using the Viterbi algorithm (cf. Section 2.3) to decode long sequences of received symbols, an erroneously decoded path will in general remerge with the correct path long before the end of the trellis. Hence, a typical error event consists of a detour from the correct path, corresponding to a *burst* of erroneously decoded information symbols, which starts and ends always with a code symbol error.

Communicating over a BSC with crossover probability p and using ML decoding, the burst error probability P_{B} of a convolutional code can be upper-bounded [Vit71] by

$$P_{\mathrm{B}} < T(W)\Big|_{W=2\sqrt{p(1-p)}} \tag{6.1}$$

where $T(W)$ is the path weight enumerator (2.13) of the convolutional encoder. Since the Viterbi spectrum is an encoder property, equivalent encoders might not necessarily yield the same upper bound on P_{B}.

■ **Example 6.1:**
The systematic generator matrix

$$G_1(D) = \begin{pmatrix} 1 & 0 & 1+D \\ 0 & 1 & D \end{pmatrix}$$

realized in OCF has the path weight enumerator

$$\begin{aligned} T_1^{(\mathrm{ocf})}(W) &= \frac{W^2 + W^3 - W^5}{1 - W - W^2} \\ &= W^2 + 4W^3 + 5W^4 + 8W^5 + 13W^6 + \cdots \end{aligned}$$

while its CCF realization yields the path weight enumerator

$$\begin{aligned} T_2^{(\mathrm{ccf})}(W) &= \frac{W^2 + 3W^3 - W^5}{1 - W - W^2 - 2W^3 + W^5} \\ &= W^2 + 4W^3 + 5W^4 + 10W^5 + 23W^6 + \cdots \end{aligned}$$

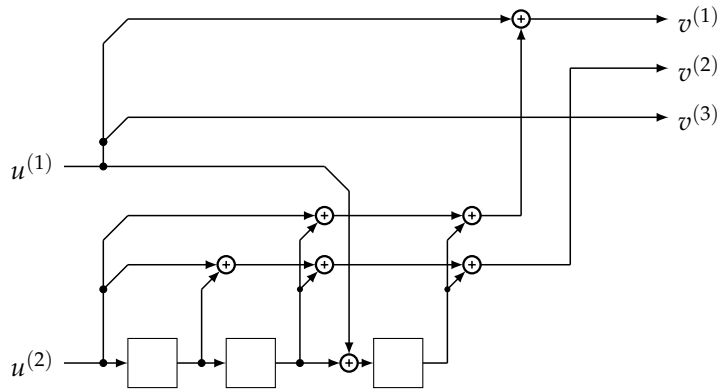


Figure 6.1: A minimal encoder for the minimal generator matrix $G_2(D)$.

Clearly, its OCF realization yields the better path weight enumerator, and hence, a better upper bound on the burst error probability, which is however a code property. This is not surprising since its OCF realization is a minimal encoder for the convolutional code determined by $G_1(D)$. Moreover, consider all paths within a code trellis of Hamming weights up to $2d_{\text{free}} - 1$, that is, paths consisting of one complete (merging with the all-zero path) detour plus an incomplete detour starting from the zero state. The Hamming weights of such paths are a code property, and hence equivalent generator matrices have identical path weight enumerators at least up to Hamming weight $2d_{\text{free}} - 1$.

■ **Example 6.2:**
The minimal realization of the minimal generator matrix [JZ99]

$$G_2(D) = \begin{pmatrix} 1+D & D & 1 \\ 1+D^2+D^3 & 1+D+D^2+D^3 & 0 \end{pmatrix}$$

is neither in CCF nor OCF, while its realization shown in Figure 6.1 is a minimal encoder. Somewhat surprisingly the path weight enumerators are the same for all three realizations, namely,

$$\begin{aligned} T_2^{(\text{min})}(W) &= T_2^{(\text{ccf})}(W) = T_2^{(\text{ocf})}(W) \\ &= W^4 + 5W^5 + 24W^6 + 71W^7 + 238W^8 + \cdots \end{aligned}$$

■ **Example 6.3:**
Consider the following two generator matrices

$$\begin{aligned} G_3(D) &= \begin{pmatrix} 1 & 1+D \end{pmatrix} \\ G_4(D) &= \begin{pmatrix} D^2 & 1+D \end{pmatrix} \end{aligned}$$

and their path weight enumerators

$$\begin{aligned} T_3^{(\mathrm{ccf})}(W) &= \frac{W^3}{1-W} \\ &= W^3 + W^4 + W^5 + W^6 + \cdots \\ T_4^{(\mathrm{ccf})}(W) &= \frac{W^3}{1-W-W^3} \\ &= W^3 + W^4 + W^5 + 2W^6 + \cdots \end{aligned}$$

Since the two path weight enumerators are different, they yield different Viterbi bounds on the burst error probability P_{B}.

Moreover, if we use sequential decoding, the computational performance using $G_4(D)$ will be much worse since its distance profile is not optimum while it is for $G_3(D)$. On the other hand, if we consider the convolutional code $\mathcal{C}_4$, encoded by $G_4(D)$, and shift the first code symbol in each tuple two steps to the left, we obviously obtain the first convolutional code. Hence, it can be expected that these convolutional codes share some common properties.

As mentioned in Chapter 2, two convolutional codes $\mathcal{C}$ and $\mathcal{C}'$ are *equivalent* if the code sequences $\boldsymbol{v} \in \mathcal{C}$ are finite-memory permutations of the code sequences $\boldsymbol{v}' \in \mathcal{C}'$. Hence, we introduce the following concept:

Definition 6.1
Two generator matrices $G(D)$ and $G'(D)$ are *weakly equivalent* (WE) if they encode equivalent convolutional codes.

Consider a *generalized permutation matrix* $\Pi(D)$, whose entries are either zero or monomials D^i, where i is a (possibly) different integer for each nonzero entry, and let $G(D)$ be a generator matrix of a rate $R = b/c$ convolutional code $\mathcal{C}$. By multiplying $G(D)$ from the right with a generalized permutation matrix $\Pi(D)$ of size $c \times c$ we obtain a generator matrix $G'(D)$ of an equivalent convolutional code $\mathcal{C}'$.

Thus, it follows from Definition 6.1 that all generator matrices $G'(D)$ of the form

$$G'(D) = T(D)G(D)\Pi(D)$$

are WE to $G(D)$, where $T(D)$ is a nonsingular $b \times b$ matrix and $\Pi(D)$ is a $c \times c$ generalized permutation matrix.

■ **Example 6.3 (Cont'd):**
For the previously used generator matrices $G_3(D)$ and $G_4(D)$ we obtain that

$$G_4(D) = G_3(D) \begin{pmatrix} D^2 & 0 \\ 0 & 1 \end{pmatrix}$$

and hence these generator matrices are WE to each other.

■ **Example 6.4:**
The following matrices

$$G_5(D) = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1+D & 1+D & 0 & 1 \\ 0 & D & 1 & 1+D & D \end{pmatrix}$$

$$G_6(D) = \begin{pmatrix} 1+D & 1+D & 0 & 1 & 0 \\ D & 1 & 1+D & D & 0 \\ D & D & D & 0 & 1 \end{pmatrix}$$

$$G_7(D) = \begin{pmatrix} 1 & 1+D & D & 0 & 1 \\ D & D & 0 & 1 & 1 \\ D^2+D & 0 & D & 0 & 1+D \end{pmatrix}$$

are considered »equivalent« in [TLF06] since they can be obtained from each other by cyclic shifting of their trellis module. In our terminology these matrices are WE because

$$G_6(D) = T(D)G_5(D)\Pi(D)$$

and

$$G_7(D) = T(D)G_6(D)\Pi(D)$$

where

$$T(D) = \begin{pmatrix} 0 & D^{-1} & 0 \\ 0 & 0 & D^{-1} \\ 1 & 0 & 0 \end{pmatrix}$$

and

$$\Pi(D) = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ D & 0 & 0 & 0 & 0 \\ 0 & D & 0 & 0 & 0 \\ 0 & 0 & D & 0 & 0 \\ 0 & 0 & 0 & D & 0 \end{pmatrix}$$

If we tailbite the convolutional codes encoded by the WE generator matrices $G_5(D)$, $G_6(D)$, and $G_7(D)$, we obtain the same (block code) spectrum for all three codes, even though they have different overall constraint lengths. For example, using the tailbiting length $M = 1000$ yields the block code spectrum

$$B^{(\mathrm{tb})}_{M=1000}(W) = 1 + 1000W^4 + 12000W^5 + 32000W^6 + \cdots$$

for all three tailbitten convolutional codes.

6.2 BLOCK SPECTRA FOR ZERO-TAIL TERMINATED AND TRUNCATED CONVOLUTIONAL CODES

Consider a rate $R = b/c$ convolutional code with memory m, zero-tail terminated to length $M + m$ c-tuples. Computing the (block code) spectrum for the two zero-tail terminated convolutional codes with WE parent generator matrices $G_3(D)$ and $G_4(D)$ with termination length $M = 1000$ yields the identical spectra

$$B^{(\mathrm{zt})}_{3,M=1000}(W) = B^{(\mathrm{zt})}_{4,M=1000}(W) = \\ 1 + 1000W^3 + 999W^4 + 998W^5 + 499498W^6 + \cdots$$

Note that the block length for $\mathcal{C}_4^{(\mathrm{zt})}$ has to be chosen to be one c-tuple longer than that for $\mathcal{C}_3^{(\mathrm{zt})}$ since $m_4 = m_3 + 1 = 2$.

Next, consider the rate $R = 1/2$ convolutional codes with free distance $d_{\mathrm{free}} = 6$ and memory $m = 3$, determined by its generator matrix

$$G_8(D) = \begin{pmatrix} 1 + D + D^2 + D^3 & 1 + D^2 + D^3 \end{pmatrix}$$

Its normalized (by M) zero-tail terminated and truncated spectra are shown in Table 6.1.

Clearly, the zero-tail terminated spectrum is a good approximation up to $2d_{\mathrm{free}} - 1$ of the Viterbi spectrum (2.12) for the corresponding minimal encoder. Since the length of the detour of weight d_{free} is $m + 2$, not $m + 1$ c-tuples, we do not get integers when computing the normalized spectrum for the zero-tail terminated convolutional code. Had its length been $m + 1$ c-tuples, it would yield perfect agreement at least for d_{free}.

However, zero-tail termination causes a slightly decreased code rate (1.75). Truncating a convolutional code after M c-tuples does not introduce such a rate loss, but yields a different normalized truncated spectrum as can be seen in Table 6.1. Only the spectral components for weights $d_{\mathrm{free}} = 6$ and $d_{\mathrm{free}} + 1 = 7$ of the truncated normalized spectrum (since $c = 2$ in this case) are a good approximation of the corresponding components in the Viterbi spectrum. For higher weights, a significant number of code sequences with weights $d_{\mathrm{free}} + c$, $d_{\mathrm{free}} + c + 1$, ..., that will not merge with the all-zero state contribute to the normalized truncated spectrum.

6.3 MACWILLIAMS-TYPE IDENTITIES

A natural approach to study duality and MacWilliams-type identities for convolutional codes is based on obtaining sequences of block codes from a parent convolutional code and applying the MacWilliams identity to these block

Weight	Viterbi spectrum	Normalized ZT spectrum	Normalized TR spectrum	
		$M = 1000$	$M = 200$	$M = 1000$
0	0	0.001	0.005	0.001
1	0	0	0	0
2	0	0	0.005	0.001
3	0	0	0.015	0.003
4	0	0	0.035	0.007
5	0	0	0.080	0.016
6	1	0.999	1.150	1.030
7	3	2.995	3.305	3.061
8	5	4.981	6.635	6.127
9	11	10.940	18.185	17.237
10	25	24.829	48.015	46.403
11	55	54.541	120.910	118.582
12	121	614.319	390.185	788.037
13	267	3227.073	1272.990	3675.400

Table 6.1: Normalized zero-tail terminated and truncated spectra for the generator matrix $G_8(D)$.

codes. Hence, we will start by recalling MacWilliams identity for block codes together with two duality definitions for convolutional codes:

Theorem 6.1 (MacWilliams identity for block codes [MS77]) Let $\mathcal{B}$ be a binary block code of rate $R = K/N$ and let $\mathcal{B}^{\perp}$ be its dual of rate $R = (N - K)/N$. Then their spectra satisfy

$$\mathcal{S}_{\mathcal{B}^{\perp}}(x, y) = \frac{1}{2^k}\mathcal{S}_{\mathcal{B}}(x + y, x - y) \tag{6.2}$$

where the spectrum of the block code $\mathcal{B}$ is given by

$$\mathcal{S}_{\mathcal{B}}(x, y) = \sum_{v \in \mathcal{B}} x^{n - w_{\mathrm{H}}(v)} y^{w_{\mathrm{H}}(v)}$$

and $w_{\mathrm{H}}(v)$ denotes the Hamming weight of the codeword v.

Definition 6.2
A *dual code* $\mathcal{C}^{\perp}$ to a rate $R = b/c$ convolutional code $\mathcal{C}$ is the set of all c-tuples of code sequences $\boldsymbol{v}^{\perp}$ such that the inner product

$$\left(\boldsymbol{v}, \boldsymbol{v}^{\perp}\right) = \boldsymbol{v}\left(\boldsymbol{v}^{\perp})\right)^{\mathrm{T}} = 0 \tag{6.3}$$

that is, $\boldsymbol{v}$ and $\boldsymbol{v}^{\perp}$ are orthogonal for all code sequences $\boldsymbol{v} \in \mathcal{C}$.

The dual code $\mathcal{C}^{\perp}$ of a rate $R = b/c$ convolutional code is a rate $R = (c-b)/c$ convolutional code encoded by the semi-infinite generator matrix $\boldsymbol{G}^{\perp}$ which satisfies

$$\boldsymbol{G}\left(\boldsymbol{G}^{\perp}\right)^{\mathrm{T}} = \boldsymbol{0} \tag{6.4}$$

Moreover, it is a vector space of dimension $c - b$ over $\mathbb{F}_2((D))$. In [GLS09], the dual code $\mathcal{C}^{\perp}$ is called *sequence space dual*.

Definition 6.3
The *convolutional dual code* $\mathcal{C}_{\perp}$ to a convolutional code $\mathcal{C}$, which is encoded by the rate $R = b/c$ generator matrix $G(D)$, is the set of all code sequences encoded by any rate $R = (c-b)/c$ generator matrix $G_{\perp}(D)$ such that

$$G(D)G_{\perp}^{\mathrm{T}}(D) = \boldsymbol{0} \tag{6.5}$$

In [GLS09], the convolutional dual code $\mathcal{C}_{\perp}$ is called *module-theoretic dual*.

In other words, Definition 6.2 is related to the orthogonality of the vectors $(G_0, G_1, \ldots, G_m)$ and $(G_0^{\perp}, G_1^{\perp}, \ldots, G_{m^{\perp}}^{\perp})$ while Definition 6.3 is based on the orthogonality of the polynomials $G(D)$ and $G_{\perp}(D)$. In particular, note that for two arbitrary polynomials

$$a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \tag{6.6}$$

and

$$b(x) = b_0 + b_1 x + \cdots + b_{n-1}x^{n-1} \tag{6.7}$$

it follows from the equality

$$a(x)b(x) = 0 \tag{6.8}$$

that

$$\left(\boldsymbol{a}, \overleftarrow{\boldsymbol{b}}\right) = 0 \tag{6.9}$$

but in general $(\boldsymbol{a}, \boldsymbol{b}) \neq 0$, where $\boldsymbol{a} = (a_0\, a_1 \ldots a_{n-1})$ and $\overleftarrow{\boldsymbol{b}} = (b_{n-1}\, b_{n-2} \ldots b_0)$.

Since we deal with a particular case of the above statement, it can be easily shown that the polynomial generator matrix $G^{\perp}(D)$ of the dual code $\mathcal{C}^{\perp}$ is

the *reversal* with respect to the polynomial generator matrix $G_\perp(D)$ of the convolutional dual code $\mathcal{C}_\perp$, that is,

$$G^\perp(D) = G_\perp(D^{-1})D^{m_\perp} = \overleftarrow{G}_\perp(D)$$

where

$$G^\perp(D) = G_0^\perp + G_1^\perp D + \cdots + G_{m^\perp}^\perp D^{m^\perp}$$

and

$$\overleftarrow{G}_\perp(D) = G_{m_\perp,\perp} + G_{m_\perp-1,\perp} D + \cdots + G_{0,\perp} D^{m_\perp}$$

from which $G_j^\perp = G_{m_\perp-j,\perp}$, $j = 0, 1, \ldots, m^\perp$ and $m_\perp = m^\perp$ follow. In general, the dual code and the corresponding convolutional dual code are different.

Based on these definitions, MacWilliams identities for convolutional codes can be interpreted in different ways.

We start by considering the codewords of a terminated convolutional code $\mathcal{C}_M$ and analyze which termination procedure does not violate (6.3). Clearly, both zero-tail termination as well as tailbiting satisfy (6.3). However, duals of zero-tail terminated convolutional codes are not zero-tail terminated convolutional dual codes since terminations to length $M + m$ and $M + m^\perp$ of $\boldsymbol{G}$ and $\boldsymbol{G}^\perp$, respectively, yield generator matrices of the two block codes of rates $Mb/(M + m)c$ and $M(c - b)/(M + m^\perp)c$ which are not duals of each other (since, in general, $m \neq m^\perp$).

On the other hand, truncating $\boldsymbol{G}$ and $\boldsymbol{G}^\perp$ to length M c-tuples yields the two generator matrices $\boldsymbol{G}_M^{(\mathrm{tr})}$ and $\boldsymbol{G}_M^{(\mathrm{tr})\perp}$ of size $Mb \times Mc$ and $M(c - b) \times Mc$, respectively, but

$$\boldsymbol{G}_M^{(\mathrm{tr})} \left(\boldsymbol{G}_M^{(\mathrm{tr})\perp}\right)^{\mathrm{T}} \neq \boldsymbol{0}$$

That is, the truncated versions of $\mathcal{C}$ and $\mathcal{C}^\perp$ are not orthogonal since the last $t - m + 1$ rows of $\boldsymbol{G}_M^{(\mathrm{tr})}$ as well as the last $t - m^\perp + 1$ rows of $\boldsymbol{G}_M^{(\mathrm{tr})\perp}$ are not complete; they do not contain all submatrices G_i, $i = 0, 1, \ldots, m$, and not all submatrices $G_i^\perp$, $i = 0, 1, \ldots, m^\perp$, respectively. The products of these incomplete rows are equal to incomplete matrix convolutions.

Considering however the generator matrix of the dual code $\mathcal{C}^\perp$ reverse-truncated by M and given by

$$\overleftarrow{\boldsymbol{G}}_M^{(\mathrm{tr})\perp} = \begin{pmatrix} G_{m^\perp}^\perp & & & & & & \\ G_{m^\perp-1}^\perp & G_{m^\perp}^\perp & & & & & \\ \vdots & \vdots & \ddots & & & & \\ G_0^\perp & G_1^\perp & \cdots & G_{m^\perp}^\perp & & & \\ & G_0^\perp & \cdots & G_{m^\perp-1}^\perp & G_{m^\perp}^\perp & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & G_0^\perp & \cdots & G_{m^\perp-1}^\perp & G_{m^\perp}^\perp \end{pmatrix} \tag{6.10}$$

we obtain that

$$\boldsymbol{G}_M^{(\mathrm{tr})} \left(\overleftarrow{\boldsymbol{G}}_M^{(\mathrm{tr})\perp} \right)^{\mathrm{T}} = \boldsymbol{0}$$

Note that $\overleftarrow{\boldsymbol{G}}_M^{(\mathrm{tr})\perp}$ is *not* a generator matrix of a truncated convolutional code, but writing both its rows and columns in reversed order, yields the truncated reversal of $\boldsymbol{G}_M^{(\mathrm{tr})\perp}$ as

$$\begin{pmatrix} G_{m^\perp}^\perp & \cdots & G_1^\perp & G_0^\perp & & & \\ & G_{m^\perp}^\perp & \cdots & G_1^\perp & G_0^\perp & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & G_{m^\perp}^\perp & \cdots & G_1^\perp & G_0^\perp \\ & & & & G_{m^\perp}^\perp & \cdots & G_1^\perp \\ & & & & & \ddots & \vdots \\ & & & & & & G_{m^\perp}^\perp \end{pmatrix}$$

which is a generator matrix of a truncated convolutional code.

Since terminated (truncated and tailbitten) convolutional codes and their duals are block codes, their spectra satisfy MacWilliams identity (cf. Theorem 6.1).

The spectra of the corresponding zero-tail terminated, truncated, and tailbitten convolutional codes can be computed via the $2^\nu \times 2^\nu$ WAM $A(W) \in \mathbb{Z}[[W]]^{2^\nu \times 2^\nu}$ of the encoder of the parent convolutional code (see, for example, [McE98]), whose entries are generating functions of the formal variable W. Its (i,j)th entry is a sum of monomials $\sum_w W^w$ whose degrees w are determined by the Hamming weights w of all parallel branches directly connecting the states i and j in its state-transition diagram (cf. Section 2.1). In particular, such an entry is a monomial in case of only one connecting branch and zero if there is no connecting branch.

Since the (i,j)th entry of $A(W)^M$ is a generating function of the Hamming weights of paths of length M branches going from state i to state j, the spectra of the corresponding zero-tail terminated, truncated, and tailbitten convolutional codes are

$$B_M^{(\mathrm{zt})}(W) = \boldsymbol{z} A(W)^M \boldsymbol{z}^{\mathrm{T}} \tag{6.11}$$

$$B_M^{(\mathrm{tr})}(W) = \boldsymbol{z} A(W)^M \boldsymbol{1}^{\mathrm{T}} \tag{6.12}$$

$$B_M^{(\mathrm{tb})}(W) = \mathrm{Tr}\left(A(W)^M \right) \tag{6.13}$$

where $\boldsymbol{z}$=(1 0 ... 0) and $\boldsymbol{1}$=(1 1 ... 1) are row vectors of length 2^ν, and $B_M^{()}(W)$ is a polynomial in W with integer coefficients, that is, $B_M^{()}(W) \in \mathbb{Z}[W]$.

Theorem 6.2 Let $\mathcal{C}_M$ be a binary convolutional code of rate $R = b/c$, truncated after M c-tuples and let $\mathcal{C}_M^{\perp}$ be its dual code. Then

$$\sum_{i=0}^{Mc} A_i^{\perp} x^{Mc-i} y^i = \frac{1}{2^{Mb}} \sum_{i=0}^{Mc} A_i \left(x+y\right)^{Mc-i} \left(x-y\right)^i \tag{6.14}$$

with

$$\boldsymbol{z} A(W)^M \mathbf{1}^{\mathrm{T}} = \sum_{i=0}^{Mc} A_i W^i \tag{6.15}$$

and

$$\mathbf{1} A^{\perp}(W)^M \boldsymbol{z}^{\mathrm{T}} = \sum_{i=0}^{Mc} A_i^{\perp} W^i \tag{6.16}$$

where $A(W)$ and $A^{\perp}(W)$ are WAMs obtained from the state-transition diagrams for the minimal encoders of $\mathcal{C}$ and $\mathcal{C}^{\perp}$, respectively. In particular, note that (6.14) corresponds directly to (6.2) with $N = Mc$ and $K = Mb$. In [For11] it is showed that the same approach holds for tailbiting codes.

The following example illustrates the considered notions of duality for the convolutional code analyzed in [SM77] where the absence of the MacWilliams identity for the Viterbi spectra was stated.

■ **Example 6.5:**
Shearer and McEliece [SM77] considered the generator matrix

$$G(D) = \begin{pmatrix} 1 & D & 1+D \end{pmatrix}$$

and its convolutional dual [JZ99]

$$G_{\perp}(D) = \begin{pmatrix} 1 & 1 & 1 \\ D & 1 & 0 \end{pmatrix}$$

where

$$G(D)\Big(G_{\perp}(D)\Big)^{\mathrm{T}} = \mathbf{0}$$

and showed that MacWilliams identity does not hold for the Viterbi spectra of their minimal encoders.

We consider the same generator matrix $G(D)$ together with the generator matrix of its dual, given in minimal-basic form

$$G^{\perp}(D) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1+D & 1 \end{pmatrix}$$

It is easy to verify that $G(D)\left(G^{\perp}(D)\right)^{\mathrm{T}} \neq \mathbf{0}$, but for their corresponding code sequences it holds that

$$\boldsymbol{v}\left(\boldsymbol{v}^{\perp}\right)^{\mathrm{T}} = 0$$

The generator matrix $G(D)$ has the semi-infinite binary generator matrix

$$\boldsymbol{G} = \begin{pmatrix} 101 & 011 & & \\ & 101 & 011 & \\ & & \ddots & \ddots \end{pmatrix}$$

As a simple example, we consider the corresponding terminated convolutional code with truncation length $M = 2$, whose generator matrix is

$$G_{M=2} = \begin{pmatrix} 101 & 011 \\ 000 & 101 \end{pmatrix}$$

On the other hand, the reversal of its dual code follows as

$$\overleftarrow{G}^{\perp}(D) = \begin{pmatrix} D & D & D \\ 0 & 1+D & D \end{pmatrix}$$

or in minimal-basic form as

$$\overleftarrow{G}^{\perp}_{\mathrm{mb}}(D) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1+D & D \end{pmatrix}.$$

For truncation length $M = 2$ we have

$$\overleftarrow{G}^{\perp}_{M=2} = \begin{pmatrix} 111 & 000 \\ 010 & 011 \\ \\ 000 & 111 \\ 000 & 010 \end{pmatrix}$$

or, equivalently, the generator matrix for the dual of the truncated convolutional code $\mathcal{C}_{M=2}$ is

$$G^{\perp}_{M=2} = \begin{pmatrix} 000 & 111 \\ 011 & 010 \\ \\ 111 & 000 \\ 010 & 000 \end{pmatrix}$$

It is easy to verify that

$$G_{M=2}\left(G^{\perp}_{M=2}\right)^{\mathrm{T}} = \begin{pmatrix} 101 & 011 \\ 000 & 101 \end{pmatrix} \begin{pmatrix} 00 & 10 \\ 01 & 11 \\ 01 & 10 \\ \\ 10 & 00 \\ 11 & 00 \\ 10 & 00 \end{pmatrix} = \mathbf{0}$$

The corresponding WAMs for $G(D)$ and $G^{\perp}(D)$, realized in CCF, are

$$A(W) = \begin{pmatrix} 1 & W^2 \\ W^2 & W^2 \end{pmatrix} \tag{6.17}$$

and

$$A^{\perp}(W) = \begin{pmatrix} 1+W^3 & W+W^2 \\ W+W^2 & W+W^2 \end{pmatrix} \tag{6.18}$$

Hence, the spectra for $\mathcal{C}_{M=2}$ and its dual $\mathcal{C}^{\perp}_{M=2}$ follow as

$$\begin{aligned} \boldsymbol{z}A(W)^2\mathbf{1}^{\mathrm{T}} &= 1+W^2+2W^4 \\ \mathbf{1}A^{\perp}(W)^2\boldsymbol{z}^{\mathrm{T}} &= 1+W+3W^2+6W^3+3W^4+W^5+W^6 \end{aligned} \tag{6.19}$$

where $\boldsymbol{z} = (1,0)$ and $\mathbf{1} = (1,1)$. Clearly, MacWilliams identity holds for those block code spectra.

In [GLS09] and [For11] another interpretation of MacWilliams-type identities for convolutional codes is considered. In particular, MacWilliams-type identities with respect to the WAMs of the minimal encoders of the dual code $\mathcal{C}^{\perp}$ and convolutional dual code $\mathcal{C}_{\perp}$ are proven. For completeness, we include the results of [GLS09] and [For11] for binary convolutional codes, where it is shown that

$$A^{\perp}(W) = 2^{-b}\,(1+W)^c\,HA\left(\frac{1-W}{1+W}\right)H^{\mathrm{T}}$$

and

$$A_{\perp}(W) = 2^{-b}\,(1+W)^c\,HA^{\mathrm{T}}\left(\frac{1-W}{1+W}\right)H^{\mathrm{T}}$$

where

$$H = \left\{(-1)^{(\boldsymbol{u}_i,\boldsymbol{u}_j)}\right\},\ i,j = 0,1,\ldots,2^{\nu}-1$$

is the $2^{\nu}\times 2^{\nu}$ Hadamard transform matrix, ν is the overall constraint length of the convolutional code $\mathcal{C}$, and $\boldsymbol{u}_i$ denotes a binary row vector of length ν with $\boldsymbol{u}_i \neq \boldsymbol{u}_j$ for $i \neq j$.

6.4 INFINITE SEQUENCES OF SPECTRA

In practical applications, convolutional codes are always used together with some kind of termination procedure. Hence, it is important to know the spectra of the corresponding terminations (block codes) of different lengths. Certainly, these spectra can be computed for both the parent convolutional code and its dual via their encoder WAMs with complexity of order 2^{ν}, where ν is the overall constraint length of the minimal encoder. However, the sparsity of WAMs, that is, the number of nonzero terms in each row of a WAM can be different for convolutional codes and their duals.

In this section a recursion for spectra of sequences of truncated as well as tailbitten dual codes shall be derived, using only the WAM of the encoder of the corresponding parent convolutional code and applying a MacWilliams-type identity. In particular, the order of this recursion shall be proven to be less than or equal to the rank of the WAM of the minimal encoder of the parent convolutional code.

Let the sequence of (block code) spectra for both terminations be given by

$$B_M(W) = B_0 + B_1 W + \cdots + B_{Mc} W^{Mc} \quad M = 0, 1, 2, \ldots$$

where

$$B_i = \begin{cases} A_i, & \text{for truncated codes (cf. (6.15))} \\ T_i, & \text{for tailbiting codes} \end{cases}$$

and

$$\mathrm{Tr}\left(A(W)^M\right) = \sum_{i=0}^{Mc} T_i W^i$$

Note, that the spectral components B_k can be obtained from the spectral components $B_i^{\perp}$, $i = 0, 1, \ldots, Mc$, of the dual code $\mathcal{C}_M^{\perp}$ as

$$B_k = \frac{1}{2^{Mc}} \sum_{i=0}^{Mc} B_i^{\perp} P_k(i) \quad k = 0,\, 1, \ldots,\, Mc \tag{6.20}$$

where $P_k(i)$ is a Krawtchouk polynomial [MS77].

Theorem 6.3 Let $\mathcal{C}$ be a rate $R = b/c$ convolutional code whose minimal encoder WAM $A(W)$ has rank r and let $\mathcal{C}_M$ be a truncated or tailbiting block code of $\mathcal{C}$ with block length M c-tuples. Then there exists an integer $l \leq r$ such that the (block code) spectra of $\mathcal{C}_M$ satisfy

$$B_M(W) = \sum_{i=1}^{l} a_i(W) B_{M-i}(W) \quad M = l,\, l+1, \ldots$$

where $a_i(W)$, $i = 1, 2, \ldots, r$, are the coefficients of the characteristic equation for $A(W)$.

Proof. Any matrix over a commutative ring satisfies its Hamilton-Cayley (characteristic) equation [Bro93, Ch. 7, p.62]. Since $A(W)$ has size $2^\nu \times 2^\nu$ it satisfies

$$\det\left(A(W) - \lambda I\right) = \lambda^{2^\nu} - \sum_{i=1}^{2^\nu} a_i(W) \lambda^{2^\nu - i} = 0 \tag{6.21}$$

where λ is a formal variable. Thus, it follows that

$$A(W)^{2^\nu} = \sum_{i=1}^{2^\nu} a_i(W) A(W)^{2^\nu - i} \tag{6.22}$$

Multiplying both sides of (6.22) by $A(W)^k$, $k = 0, 1, 2, ...$, yields the following recurrent equation

$$A(W)^M = \sum_{i=1}^{2^\nu} a_i(W) A(W)^{M-i} \quad M = 2^\nu, 2^\nu + 1, \ldots \tag{6.23}$$

Assuming that $A(W)$ has rank r, all of its minors of order higher than r are zero, while there exists at least one nonzero minor of order r. It is straightforward to show that the coefficient $a_i(W)$ of the characteristic equation (6.21) is completely determined by the $\binom{2^\nu}{i}$ principal minors of order i. Thus, we can conclude that all $a_i(W)$ for $i = r+1, r+2, \ldots, 2^\nu$ are zero, and hence (6.23) can be reduced to

$$A(W)^M = \sum_{i=1}^{r} a_i(W) A(W)^{M-i} \quad M = r, r+1, \ldots \tag{6.24}$$

Multiplying both sides with $\boldsymbol{z}$ and $\mathbf{1}^{\mathrm{T}}$ from the left and right, respectively, yields that the (block) spectrum $B_M^{(\mathrm{tr})}(W) = \boldsymbol{z} A(W)^M \mathbf{1}^{\mathrm{T}}$ of the a truncated convolutional code satisfies the main statement of Theorem 6.3.

Denote by $\boldsymbol{e}_k$ a row vector of length 2^ν with a single 1 as its kth entry and zeros elsewhere. Multiplying (6.24) by $\boldsymbol{e}_k$ and $\boldsymbol{e}_k^{\mathrm{T}}$ from the left and right, respectively, we obtain that the statement of Theorem 6.3 is valid for $\boldsymbol{e}_k A(W)^M \boldsymbol{e}_k^{\mathrm{T}}$.

Taking into account that

$$\sum_{k=0}^{2^\nu - 1} \boldsymbol{e}_k A(W)^M \boldsymbol{e}_k^{\mathrm{T}} = \mathrm{Tr}\left(A(W)^M\right)$$

yields

$$\mathrm{Tr}\left(A(W)^M\right) = \sum_{i=1}^{r} a_i(W) \, \mathrm{Tr}\left(A(W)^{M-i}\right) \quad M = r, r+1, \ldots \tag{6.25}$$

and thus the main statement of Theorem 6.3 is also valid for the spectra of tailbitten convolutional codes.

Finally, note that the order of (6.24) can be less than r, for example, if the only nonzero minors of order r within $A(W)$ are nonprinciple minors. ■

It follows from Theorem 6.3 that for the (block code) spectra of truncated as well as tailbitten convolutional codes, both codes, $\mathcal{C}_M$ and $\mathcal{C}_M^\perp$, satisfy the recursion of the same order l but with different coefficients, namely the coefficients of the Hamilton-Cayley equations for $A(W)$ and $A^\perp(W)$, respectively. Since the coefficients of a recurrent equation over $\mathbb{Z}[W]$ of order l can be found from $2l$ output values by solving a system of l linear equations, $2l$ spectra of

i	$a_i(W)$	$a_i^{\perp}(W)$
1	$1+W^2$	$1+W+W^2+W^3$
2	W^4-W^2	$2W^3-W^5-W$
3	W^2-W^6	$W+W^2-W^3-W^4-W^5-W^6+W^7+W^8$
4	$3W^6-2W^4-W^2$	$-W-3W^2-W^3+4W^4+2W^5-2W^6+2W^7+$ $4W^8-W^9-3W^{10}-W^{11}$
5	$W^{12}+W^{10}-3W^8-W^6+2W^4$	$3W^2-W^3-5W^4-W^5-2W^6+6W^7+6W^8-2W^9$ $-W^{10}+5W^{11}-W^{12}+3W^{13}$
6	$-W^{14}-W^{12}+2W^{10}$ $+2W^8-W^6-W^4$	$-2W^2+2W^4+6W^6-6W^8-6W^{10}$ $+6W^{12}+2W^{14}-2W^{16}$
7	0	0
8	$W^{16}-W^{14}-2W^{12}+2W^{10}$ $+W^8-W^6$	$-W^3-2W^4+6W^5+8W^6-8W^8-8W^9-$ $8W^{10}+6W^{11}+20W^{12}+6W^{13}-8W^{14}$ $-8W^{15}-8W^{16}+8W^{18}+3W^{19}-2W^{20}-W^{21}$
9	$-W^{18}+3W^{14}-3W^{10}+W^6$	$W^3+3W^4-8W^6-9W^7-3W^8+8W^9+24W^{10}$ $+18W^{11}-10W^{12}-24W^{13}-24W^{14}-10W^{15}+18W^{16}$ $+24W^{17}+8W^{18}-3W^{19}-9W^{20}-8W^{21}+3W^{23}+W^{24}$

Table 6.2: Coefficients of the recursions in Example 6.6.

terminated codes are sufficient to obtain the recursion for the sequence of the spectra of their duals and vice versa. Note that in general such a system of linear equations can be solved with reduced complexity by applying the Berlekamp-Massey algorithm [Bla85][1].
The following example illustrates the statement of Theorem 6.3.

■ **Example 6.6:**
Consider a rate $R = 1/3$ convolutional code determined by the generator matrix

$$G(D) = \begin{pmatrix} 1+D+D^2+D^3+D^4 & 1+D+D^4 & 1+D^3 \end{pmatrix}$$

[1]Although the coefficients $a_i(W)$ belong to a polynomial ring, they can be found by computations over the field of rational functions. Alternatively, the inversion-free modification of the Berlekamp-Massey algorithm [RST91] can be used.

The WAM of its minimal encoder realized in CCF is

$$A(W) = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & W & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & W & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & W & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & W & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & W & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & W & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & W^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2
\end{pmatrix}$$

Since the eight rows i and $i+1$, $i = 5, 7, 9, 11$, of $A(W)$ are pairwise linearly dependent, the matrix $A(W)$ does not have full rank $2^\nu = 2^4 = 16$, but rank $r = 12$. Moreover, it can be shown that the spectra of the sequences of truncated convolutional codes satisfy the system of linear equations

$$B_M^{(\mathrm{tr})}(W) = \sum_{i=1}^{l} a_i(W) B_{M-i}^{(\mathrm{tr})}(W) \quad t = l,\, l+1, \ldots,\, 2l-1 \tag{6.26}$$

of order $l = 9$, where the spectrum $B_M^{(\mathrm{tr})}(W)$ is computed according to (6.12). By solving (6.26) we obtain the coefficients $a_i(W)$, $i = 1, 2, \ldots, 9$, and hence according to Theorem 6.3, for any finite truncation length M, the spectra of the truncated sequence of the convolutional codes satisfy the equation of order $l = 9$ with coefficients $a_i(W)$, $i = 1, 2, \ldots, 9$, presented in the second column of Table 6.2.

By applying MacWilliams identity to the sequence of the spectra $B_0^{(\mathrm{tr})}$, $B_1^{(\mathrm{tr})}$, $\ldots$, $B_{2l-1}^{(\mathrm{tr})}$ we find the sequence of the spectra $B_0^{(\mathrm{tr})\perp}, B_1^{(\mathrm{tr})\perp}, \ldots, B_{2l-1}^{(\mathrm{tr})\perp}$ of the corresponding reverse truncations of the dual code. Inserting them into (6.26) with coefficients $a_i^\perp(W)$, $i = 1, 2, \ldots, 9$, yields the coefficients $a_i^\perp(W)$, $i = 1, 2, \ldots, 9$. Thus, the dual spectra satisfy the equation of order $l = 9$ with coefficients $a_i^\perp(W)$, $i = 1, 2, \ldots, 9$, presented in the third column of the same table. Note that the WAM of the minimal encoder of its corresponding dual code, which is not shown here, contains four nonzero entries in each row since it has rate $R = 2/3$

7

Optimum and Near-Optimum Convolutional Codes

The error-detecting and error-correcting capabilities of convolutional codes are mainly determined by their free distance d_{free} (cf. Section 1.3.2), and, in case of equal free distances, by their Viterbi spectra. This motivates the search for convolutional codes with an optimum free distance and an optimum Viterbi spectrum, so-called OFD codes, for a given rate R and overall constraint length ν.

However, when searching for optimum free distance (OFD) convolutional codes with increased overall constraint length ν, the computational complexity becomes prohibitively large. On the other hand, the set of convolutional codes with an optimum column distance profile, so-called ODP codes, contains only a fraction of possible generator matrices with nevertheless near-optimum free distance. Hence, searching »only« among ODP codes, it is possible to obtain convolutional codes with both large overall constraint lengths ν as well as near-optimum free distance d_{free}.

In Section 7.1, different distance properties of convolutional generator matrices, like the column and row distance, as well as the Smith form decomposition are discussed. Section 7.2 is devoted to finding OFD encoding matrices with various rates between $1/2$ and $4/5$ and memory up to 26, extending previously published code tables. In particular, by searching among the ensemble of convolutional parity-check matrices and apply the BEAST (cf. Section 2.4) to their syndrome trellis (cf. Section 2.1), it becomes possible to determine high-rate OFD convolutional codes with moderate search complexity. This chapter is concluded by a similar search for systematic and nonsystematic ODP convolutional encoders in Section 7.3, yielding previously unknown ODP encoder matrices with rate $1/2$ and $1/3$ as well as overall constraint length up to 40.

7.1 DISTANCE PROPERTIES

7.1.1 COLUMN DISTANCE

Consider a convolutional code $\mathcal{C}$ with memory m and generator matrix $G(D)$, whose causal information sequence

$$\boldsymbol{u}(D) = \boldsymbol{u}_0 + \boldsymbol{u}_1 D + \boldsymbol{u}_2 D^2 + \cdots$$

is encoded to form the causal code sequence

$$\boldsymbol{v}(D) = \boldsymbol{v}_0 + \boldsymbol{v}_1 D + \boldsymbol{v}_2 D^2 + \cdots$$

according to $\boldsymbol{v}(D) = \boldsymbol{u}(D)G(D)$ as given in (1.57).

The jth order *column distance* d_j^{c} of the generator matrix $G(D)$ was introduced in 1969 by Costello [Cos69] and is defined as the minimum Hamming distance d_{H} between any two code sequences $\boldsymbol{v}_{[0,j]}$ of length $(j+1)$ c-tuples, resulting from two causal information sequences $\boldsymbol{u}_{[0,j]}$ with different $\boldsymbol{u}_0$.

Due to the linearity of convolutional codes, the jth order column distance d_j^{c} coincides with the minimum Hamming weight of any path $\boldsymbol{v}_{[0,j]}$ resulting from the causal information sequence $\boldsymbol{u}_{[0,j]}$ with $\boldsymbol{u}_0 \neq \mathbf{0}$, that is,

$$d_j^{\mathrm{c}} = \min_{\boldsymbol{u}_0 \neq \mathbf{0}} \left\{ w_{\mathrm{H}}(\boldsymbol{v}_{[0,j]}) \right\} \tag{7.1}$$

where w_{H} denotes the Hamming weight.

Moreover, in case of a polynomial generator matrix $G(D)$, let G_j^{c} denote the truncation of the semi-infinite matrix G (1.49) after $j+1$ columns; that is,

$$G_j^{\mathrm{c}} = \begin{pmatrix} G_0 & G_1 & G_2 & \dots & G_j \\ & G_0 & G_1 & & G_{j-1} \\ & & G_0 & & G_{j-2} \\ & & & \ddots & \vdots \\ & & & & G_0 \end{pmatrix} \tag{7.2}$$

where $G_i = \mathbf{0}$ when $i > m$. Then the sequence $\boldsymbol{v}_{[0,j]}$ can be expressed by

$$\boldsymbol{v}_{[0,j]} = \boldsymbol{u}_{[0,j]} G_j^{\mathrm{c}} \tag{7.3}$$

and the jth order column distance in (7.1) is simplified to

$$d_j^{\mathrm{c}} = \min_{\boldsymbol{u}_0 \neq \mathbf{0}} \left\{ w_{\mathrm{H}}(\boldsymbol{u}_{[0,j]} G_j^{\mathrm{c}}) \right\} \tag{7.4}$$

Note that the column distances of a generator matrix satisfy the following conditions [JZ99]:

(i) $d_j^{\mathrm{c}} \leq d_{j+1}^{\mathrm{c}}, \quad j = 0, 1, \ldots$

(ii) The sequence $d_0^{\mathrm{c}}, d_1^{\mathrm{c}}, d_2^{\mathrm{c}}, \ldots$ is bounded from above.

(iii) d_j^{c} becomes stationary as j increases.

Since the free distance d_{free} is equal to the upper limit of the sequence of column distances, it follows that

$$\lim_{j \to \infty} d_j^{\mathrm{c}} = d_\infty^{\mathrm{c}} = d_{\mathrm{free}} \tag{7.5}$$

(OPTIMUM) DISTANCE PROFILE

Consider a generator matrix $G(D)$ of memory m and let the $(m+1)$-tuple

$$\boldsymbol{d}^{\mathrm{p}} = (d_0^{\mathrm{c}}, d_1^{\mathrm{c}}, \ldots, d_m^{\mathrm{c}}) \tag{7.6}$$

denote the *distance profile of the generator matrix* $G(D)$, where d_j^{c}, $0 \leq j \leq m$ is its jth order column distance.

Note that the distance profile is a generator matrix property. However, since the jth order column distance is invariant over equivalent generator matrices and the memory m is identical for equivalent minimal-basic encoding matrices (cf. Subsection 1.3.2), the *distance profile of a convolutional code* $\mathcal{C}$ with minimal-basic encoding matrix $G_{\mathrm{mb}}(D)$ of memory m is defined as the $(m+1)$-tuple

$$\boldsymbol{d}^{\mathrm{p}} = (d_0^{\mathrm{c}}, d_1^{\mathrm{c}}, \ldots, d_m^{\mathrm{c}}) \tag{7.7}$$

where d_j^{c} denotes the jth order column distance of $G_{\mathrm{mb}}(D)$.

Such a distance profile $\boldsymbol{d}^{\mathrm{p}}$ is said to be *superior* to another distance profile $\widetilde{\boldsymbol{d}}^{\mathrm{p}}$ of same rate R and memory m if there exists an integer ℓ such that

$$d_j^{\mathrm{c}} \begin{cases} = \widetilde{d}_j^{\mathrm{c}} & j = 0, 1, \ldots, \ell - 1 \\ > \widetilde{d}_j^{\mathrm{c}} & j = l \end{cases} \tag{7.8}$$

Moreover, a convolutional code $\mathcal{C}$ is said to have an *optimum distance profile* (to be an ODP code) if there exists no other convolutional code of the same rate and memory with a better distance profile.

Finally, let $G'(D)$ be an encoding matrix of memory m' and denote by $G(D) = G'(D)\big|_m$ the truncation of all numerator and denominator polynomials to degree $m \leq m'$. Then it follows that the encoding matrix $G(D)$ of memory m and the encoding matrix $G'' = T(D)G(D)\big|_m$ where $T(D)$ is a $b \times b$ nonsingular matrix, are equivalent over the first memory length m. In particular, they share the same distance profile $\boldsymbol{d}^{\mathrm{p}}$.

7.1.2 ROW DISTANCE

Denote as the *zero state driving information sequence* for a rate $R = b/c$ generator matrix of memory m and realized in CCF, the sequence of information tuples that causes all memory elements to be successively filled with zeros. Clearly, the length of such a sequence is at most m information b-tuples, while it might be shorter if the first memory elements contain already zeros. To simplify notations, we shall denote the zero driving sequence by $\boldsymbol{u}^{\mathrm{zs}}_{(t,t+m]}$, regardless of its actual length.

Then the jth order *row distance* d^{r}_j [Cos69] for the generator matrix $G(D)$ of memory m and realized in CCF is defined as the minimum Hamming weight of the code sequences $\boldsymbol{v}_{[0,j+m]}$ resulting from the causal information sequences

$$\boldsymbol{u}_{[0,j+m]} = \boldsymbol{u}_{[0,j]}\boldsymbol{u}^{\mathrm{zs}}_{(j,j+m)} \tag{7.9}$$

with $\boldsymbol{u}_{[0,j]} \neq \mathbf{0}$.

Similarly to (7.2), denote in case of a polynomial generator matrix $G(D)$, the truncation of the semi-infinite matrix G (1.49) after $j+1$ rows

$$G^{\mathrm{r}}_j = \begin{pmatrix} G_0 & G_1 & \dots & G_m & & & \\ & G_0 & G_1 & \dots & G_m & & \\ & & G_0 & G_1 & \dots & G_m & \\ & & & \ddots & & & \ddots \\ & & & & G_0 & G_1 & G_m \end{pmatrix} \tag{7.10}$$

Hence its jth order row distance d^{r}_j can be expressed as

$$d^{\mathrm{r}}_j = \min_{\boldsymbol{u}_{[0,j]} \neq \mathbf{0}} \left\{ w_{\mathrm{H}} \left(\boldsymbol{u}_{[0,j]} G^{\mathrm{r}}_j \right) \right\} \tag{7.11}$$

Note that truncating the semi-infinite matrix G in such a way corresponds to zero-tail termination (cf. Subsection 1.3.2).

Similarly to the column distance d^{c}_j, the row distance d^{r}_j is a generator matrix property and satisfies the following conditions [JZ99]:

(i) $d^{\mathrm{r}}_{j+1} \leq d^{\mathrm{r}}_j, \quad j = 0, 1, \dots$

(ii) $d^{\mathrm{r}}_j > 0, \quad j = 0, 1, \dots$

(iii) d^{r}_j becomes stationary as j increases.

Since the column distance d^{c}_i denotes the minimum Hamming weight of a path segments of length $i+1$ which diverges from the all-zero state at time instant $t = 0$, it is obvious that

$$d^{\mathrm{c}}_i \leq d^{\mathrm{r}}_j \ \ \forall i, j \tag{7.12}$$

and, in particular

$$d^{\mathrm{c}}_0 \leq d^{\mathrm{c}}_1 \leq \dots \leq d^{\mathrm{c}}_\infty = d_{\mathrm{free}} \leq d^{\mathrm{r}}_\infty \leq \dots d^{\mathrm{r}}_1 \leq d^{\mathrm{r}}_0 \tag{7.13}$$

where d^{r}_∞ denotes the lower limit of sequence of row distances, which is equal to the free distance d_{free} in case of noncatastrophic encoders.

7.1.3 SMITH FORM DECOMPOSITION

Let $G(D)$ be a $b \times c$ polynomial generator matrix with rank r. Using the Smith form decomposition, such a matrix can be written as

$$G(D) = A(D)\Gamma(D)B(D) \tag{7.14}$$

where $A(D)$ and $B(D)$ are polynomial matrices of size $b \times b$ and $c \times c$, respectively, and have unit determinant. The matrix $\Gamma(D)$ is called the *Smith form* of $G(D)$ and is a diagonal polynomial matrix of size $b \times c$, given by

$$\Gamma(D) = \begin{pmatrix} \gamma_1(D) & & & & & & & \\ & \gamma_2(D) & & & & & & \\ & & \ddots & & & & & \\ & & & \gamma_r(D) & & & & \\ & & & & 0 & & & \\ & & & & & \ddots & & \\ & & & & & & 0 & \dots & 0 \end{pmatrix} \tag{7.15}$$

with nonzero elements $\gamma_i(D)$, $i = 1, 2, \dots, r$, on its diagonal positions. These elements are called the *invariant factors* of $G(D)$ and are uniquely determined polynomials satisfying

$$\gamma_i(D) \mid \gamma_{i+1}(D), \;\; i = 1, 2, \dots, r \tag{7.16}$$

The pre- and post-multipliers, $A(D)$ and $B(D)$, respectively, are defined as the product of the following two types of elementary matrices. The first matrix type interchanges either two rows or two columns. For example, to interchange the ith and the jth row (or column), the corresponding pre- (or post-) multiplier, is given by

$$P_{ij} = \begin{array}{c} \left(\begin{array}{ccccccccccc} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & 0 & & & & 1 & & & \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & 1 & & & & 0 & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{array}\right) \end{array} \begin{array}{l} \\ \\ \\ \text{row } i \\ \\ \\ \\ \text{row } j \\ \\ \\ \\ \end{array}$$

The second matrix type adds all elements in one row (or column) to the corresponding elements in another row (or column), multiplied by a fixed polynomial $p(D)$. For example, to add all the elements of the ith row (or column)

to the corresponding elements of the jth row (or column), multiplied by a constant polynomial factor $p(D)$, the corresponding multiplier is

$$R_{ij}\left(p(D)\right) = \begin{array}{c} \left(\begin{array}{ccccccc} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & \cdots & p(D) & \cdots & \\ & & & \ddots & \vdots & & \\ & & & & 1 & \cdots & \\ & & & & & \ddots & \\ & & & & & & 1 \end{array}\right) \end{array} \begin{array}{l} \\ \\ \text{row } i \\ \\ \text{row } j \\ \\ \\ \end{array}$$

Premultiplication by any of these elementary matrices results in the associated transformation being performed on the rows, whereas postmultiplication yields the same for columns. In particular, note that these matrices satisfy $P_{ij}^{-1} = P_{ij}$ and $R_{ij}^{-1}(p(D)) = R_{ij}(-p(D))$ and have unit determinant, that is, $\det(P_{ij}) = \det(R_{ij}(p(D))) = 1$.

Since both elementary matrices are polynomial square matrices with unit determinant and have a polynomial inverse, the same holds for their products, that is, the pre- and post-multiplier $A(D)$ and $B(D)$ [JZ99]. Moreover, using its Smith form decomposition, it can be shown that a rate $R = b/c$ polynomial encoding matrix is noncatastrophic if and only if its largest invariant factor satisfies $\gamma_b(D) = D^s$ for some nonnegative integer $s \geq 0$.

7.2 OPTIMUM FREE DISTANCE CONVOLUTIONAL CODES

When searching for OFD convolutional codes for a given rate R and complexity ν (or memory m) an *exhaustive code search* for all possible encoding polynomials (and their different combinations) has to be performed. Even though the BEAST (cf. Section 2.4) is a very fast and efficient algorithm to determine the free distance for a given encoding matrix, it becomes rather time-consuming and quickly infeasible to examine all possible combination.

For example, when searching for a rate $R = 1/2$, overall constraint length $\nu = 24$ (that is, memory $m = 24$) OFD encoding matrix $G(D)$ with

$$G(D) = G_0 + G_1 D + \cdots G_m D^m \tag{7.17}$$

where $G_0 = (11)$ and $G_m = (10)$, (01), or (11), the total number of possible encoding matrices is $3 \times 2^{2m-2} \approx 2.11 \cdot 10^{14}$. However, using different rejection rules based on the previously discussed distance properties, the ensemble of encoders can be limited to a small fraction, which nevertheless contains all OFD encoding matrices. The efficiency of these rejection rules shall be numerically illustrated on the previously mentioned example.

As the free distance d_{free} is invariant among the set of equivalent encoding matrix $G(D) = (g_0(D) \quad g_1(D))$, an obvious first step is to eliminate all equivalent encoding matrices $G(D)' = (g_1(D) \quad g_0(D))$. In other words, for a given encoding polynomial $g_0(D)$ we shall only consider such combinations with another polynomial $g_1(D)$, that satisfy $g_0(D) < g_1(D)$ (in obvious binary notation). Thereby, the number of remaining encoding matrices can be reduced to $3 \times \left(2^{2m-3} - 2^{m-2}\right) = 1.05 \cdot 10^{14}$.

To further reduce this number, all catastrophic encoding matrices (cf. Subsection 1.3.2) shall be rejected. As mentioned in Subsection 7.1.3 an encoding matrix of a rate $R = b/c$ convolutional code is noncatastrophic, if and only if, its largest invariant factor $\gamma_b(D) = D^s$ with some nonnegative integer $s \geq 0$. Moreover, since we are only interested in encoding matrices with a realizable right inverse, this rejection rule can be further strengthened to $\gamma_b(D) = 1$.

However, applying the Smith form decomposition to an encoding matrix in order to obtain its invariant factors is rather complex. On the other hand, for rate $R = 1/c$ encoding matrices, the only invariant factor $\gamma_1(D)$ can be easily determined as the greatest common divisor of its encoding polynomials

$$\gamma_1(D) = \gamma_b(D) = \gcd\left\{g_0(D),\ g_1(D), \ldots g_{c-1}(D)\right\} \tag{7.18}$$

Finally, the row distances (cf. Subsection 7.1.2) can be used as a last rejection rule to upper-bound the free distance d_{free}. Hoping to find an encoding matrix with a certain target free distance $d^{\text{t}}_{\text{free}}$, all encoding matrices with row distances $d^{\text{r}}_j < d^{\text{t}}_{\text{free}}$, $j = 0, 1, \ldots, n$, are rejected, where n is an arbitrarily chosen positive integer and typically ≈ 20 in our implementation.

Hence, applying the row distance rejection rule with target free distance $d^{\text{t}}_{\text{free}} = 27$, the number of encoding matrices is further decreased to approximately $6.3 \cdot 10^6$. Finally their free distance and Viterbi spectrum is determined by the BEAST (cf. Section 2.4), yielding an OFD encoding matrix. In Table 7.1, the results of similar searches for rate $R = 1/2$ OFD encoding matrices with memory between 12 and 26 are given, where free distances satisfying the corresponding Griesmer bound [JZ99] are highlighted by $*$.

These OFD encoding matrices coincide to a large extent with the previously published results in [BHJK04, Table II] and [JZ99, Table 8.3]. However, the previously published OFD encoding matrices for memory $m = 23$ and $m = 24$ presented in [BHJK04] are not optimal and have been further improved. For memory $m = 23$, an encoding matrix with better Viterbi spectrum was found, reducing the number of code sequences with Hamming weight $d_{\text{free}} = 26$ from 51 to 45, while for memory $m = 24$ the free distance was increased from 26 to 27. Additionally, the previously unknown OFD encoding matrices for memory $m = 25$ and $m = 26$ have been determined.

Note that an encoding matrix of a rate $R = b/c$ convolutional code contains in general bc generator polynomials, while its corresponding parity-check ma-

m	$g_0(D)$	$g_1(D)$	d_{free}	spectrum
12	53734	72304	$*16$	14, 38, 35, 108, 342, 724 [a]
13	63676	45272	16	1, 17, 38, 69, 158, 414
14	75063	56711	$*18$	26, 0, 165, 0, 845, 0 [a]
15	533514	653444	19	30, 67, 54, 167, 632, 1402 [a, b]
16	626656	463642	$*20$	43, 0, 265, 0, 1341, 0 [a, b]
17	611675	550363	20	4, 24, 76, 150, 354, 826 [a, b]
18	4551474	6354344	22	65, 0, 349, 0, 1903, 0 [a, b]
19	7504432	4625676	22	5, 52, 116, 163, 456, 1135 [b]
20	6717423	5056615	$*24$	145, 0, 225, 0, 3473, 0 [b]
21	63646524	57112134	24	17, 95, 136, 138, 679, 2149 [b]
22	64353362	41471446	25	47, 88, 137, 313, 912, 2172 [b]
23	75420671	45452137	26	45, 0, 364, 0, 1968, 0
24	766446634	540125704	27	50, 135, 118, 294, 1481, 3299
25	662537146	505722162	28	71, 196, 112, 339, 2053, 4548
26	637044367	450762321	28	9, 66, 152, 307, 757, 1823

(a) previously listed in [JZ99]. *(b)* previously listed in [BHJK04].

Table 7.1: Viterbi spectra $n_{d_{\text{free}}+1}$, $i = 0, 1, \ldots, 5$, for nonsystematic rate $R = 1/2$ OFD encoding matrices $G(D) = (g_0(D) \quad g_1(D))$ and memory $m = 12, 13, \ldots, 26$.

trix consists of $(c-b)c = c^2 - bc$ parity-check polynomials. Hence, when searching for OFD encoding matrices with rate $R = (c-1)/c$ it is more convenient to search over the complete ensemble of corresponding parity-check matrices $H(D)$ since

$$(c-1)c > c^2 - (c-1)c \tag{7.19}$$

Consider for example parity-check matrix of such a rate $R = (c-1)/c$ convolutional code $\mathcal{C}$ given by

$$H(D) = \begin{pmatrix} h_0(D) & h_1(D) & \ldots & h_{c-1}(D) \end{pmatrix} \tag{7.20}$$

Since catastrophic error propagation is an encoder property, not a code property, the corresponding rejection rule for catastrophic encoding matrices can not be applied. However, if and only if

$$\gcd\left\{h_0(D), h_1(D), \ldots, h_{c-1}(D)\right\} = 1 \tag{7.21}$$

the corresponding parity-check matrix is *minimal*, that is, there exists no parity-check with smaller overall constraint length for the same convolutional code.

ν	$h_0(D)$	$h_1(D)$	$h_2(D)$	d_{free}	spectrum
1	6	6	4	2	1, 2, 4, 10, 20, 40 [a]
2	7	6	5	3	1, 4, 14, 40, 116, 339 [b]
3	74	64	54	4	1, 5, 24, 71, 238, 862 [c]
4	62	56	52	5	2, 13, 45, 143, 534, 2014 [a]
5	61	55	53	6	6, 27, 70, 285, 1103, 4063 [d]
6	634	514	504	7	17, 53, 133, 569, 2327, 8624 [e]
7	772	662	576	8	41, 0, 528, 0, 7497, 0 [d]
8	631	555	477	8	6, 42, 153, 510, 1853, 7338 [c]
9	7264	6214	4504	9	17, 81, 228, 933, 3469, 13203 [c]
10	7642	6406	4232	10	69, 0, 925, 0, 13189, 0 [c]
11	7741	6667	5715	10	10, 80, 260, 864, 3336, 13131
12	70754	62364	42074	11	32, 144, 477, 1769, 6718, 25717
13	72166	60302	52536	12	116, 0, 1768, 0, 24984, 0
14	71341	64657	40773	12	22, 134, 464, 1702, 6477, 24767

(a) previously listed in [AMB04]. *(b)* previously listed in [CCG79].
(c) previously listed in [CHL97]. *(d)* previously listed in [BK97].
(e) previously listed in [Paa74].

Table 7.2: Viterbi spectra $n_{d_{\text{free}}+1}$, $i = 0, 1, \ldots, 5$, for nonsystematic rate $R = 2/3$ OFD convolutional encoders specified by their parity-check matrices with overall constraint length $\nu = 1, 2, \ldots, 14$.

Moreover, using the relationship between (1.23) and (1.27), it is possible to obtain c different, not necessarily minimal, generator matrices $G^{(i)}$, $i = 0, 1, \ldots, c-1$, for the given convolutional code $\mathcal{C}$, namely,

$$G^{(i)}(D) = \begin{pmatrix} h_i(D) & & & & & & h_0(D) \\ & \ddots & & & & & \vdots \\ & & h_i(D) & & & & h_{i-1}(D) \\ & & & h_i(D) & & & h_{i+1}(D) \\ & & & & \ddots & & \vdots \\ & & & & & h_i(D) & h_{c-1}(D) \end{pmatrix} \tag{7.22}$$

Clearly, the row distances $d_j^{\mathrm{r}(i)}$, $j = 0, 1, \ldots, n$, for these (nonminimal) generator matrices upper-bound their free distance d_{free}. Hence, hoping to find a convolutional code $\mathcal{C}$ with a certain target free distance $d_{\text{free}}^{\mathrm{t}}$, all matrices are rejected if any row distance among those c (nonminimal) generator matrices

ν	$h_0(D)$	$h_1(D)$	$h_2(D)$	$h_3(D)$	d_{free}	spectrum
1	6	6	4	4	2	2, 8, 17, 40, 96, 224 [a]
2	7	6	5	2	3	6, 23, 80, 284, 1027, 3724 [b]
3	74	64	54	44	4	5, 36, 152, 708, 3424, 16312 [c]
4	72	62	56	46	4	1, 16, 84, 376, 1912, 9728 [a]
5	77	65	61	47	5	7, 45, 223, 1066, 5612, 29012 [d]
6	604	564	554	434	6	27, 118, 529, 2978, 15201, 79518 [e]
7	702	632	556	422	6	5, 65, 292, 1442, 7618, 39734 [e]
8	767	743	551	461	7	25, 184, 714, 4081, 20038, 110599
9	7464	6774	5114	4104	8	131, 0, 3574, 0, 97035, 0
10	7276	6252	5642	4406	8	25, 202, 919, 4552, 24327, 128857

(a) previously listed in [AMB04]. *(b)* previously listed in [CCG79].
(c) previously listed in [Paa74]. *(d)* previously listed in [TLUF06].
(e) previously listed in [CHL97].

Table 7.3: Viterbi spectra $n_{d_{\text{free}}+1}$, $i = 0, 1, \ldots, 5$, for nonsystematic rate $R = 3/4$ OFD convolutional encoders specified by their parity-check matrices with overall constraint length $\nu = 1, 2, \ldots, 10$.

is less than the target free distance $d^{\text{t}}_{\text{free}}$, that is,

$$\min_{0 \le i < c} \left\{ d_j^{\text{r}(i)} \right\} < d^{\text{t}}_{\text{free}} \quad j = 0, 1, \ldots, n \tag{7.23}$$

Finally, the BEAST is applied to the syndrome trellis specified by the parity-check matrix $H(D)$ to determine the free distance and Viterbi spectrum for all remaining parity-check matrices.

Note that for rate $R = b/c$ convolutional codes, the computational complexity of determining the row distances increases exponentially with b. On the other hand, applying the BEAST to syndrome trellis of high rate convolutional codes with moderate overall constraint length ν, efficiently yields their free distance. Hence, as a complexity tradeoff, we typically use only the first two row distances as a rejection rule, before determining the actual free distance as well as the Viterbi spectrum with the BEAST.

The parity-check matrices for rate $R = 2/3$, $R = 3/4$, and $R = 4/5$ OFD codes are given in Tables 7.2 – 7.4. Even though most of these parity-check polynomials (or their corresponding generator matrices) have been listed in previous publications by [Paa74], [CCG79], [BK97], [CHL97], [AMB04], and [TLUF06], their OFD property was mostly unknown.

ν	$h_0(D)$	$h_1(D)$	$h_2(D)$	$h_3(D)$	$h_4(D)$	d_{free}	spectrum
1	6	6	6	4	4	2	4, 12, 39, 148, 492, 1632 [a]
2	7	7	6	5	2	2	1, 9, 47, 229, 1095, 5265 [a]
3	74	70	64	54	44	3	1, 21, 139, 776, 4583, 27380 [a]
4	72	62	66	56	46	4	7, 56, 376, 2236, 14385, 92304 [a]
5	71	66	57	45	41	4	1, 36, 220, 1349, 8976, 58757
6	774	704	624	554	514	5	11, 100, 620, 4024, 26557, 177078
7	772	762	612	506	426	6	70, 245, 2504, 11894, 104486
8	717	667	571	535	441	6	14, 174, 1080, 6936, 46364, 309835
9	7754	6514	6304	5524	4474	6	1, 80, 576, 3374, 22207, 151637

(a) previously listed in [AMB04].

Table 7.4: Viterbi spectra $n_{d_{\text{free}}+1}$, $i = 0, 1, \ldots, 5$, for nonsystematic rate $R = 4/5$ OFD convolutional encoders specified by their parity-check matrices with overall constraint length $\nu = 1, 2, \ldots, 9$.

7.3 OPTIMUM DISTANCE PROFILE CONVOLUTIONAL CODES

For convolutional codes with increased overall constraint length ν the complexity of an exhaustive code search becomes prohibitively large. However, limiting ourselves to the subclass of ODP codes (cf. Subsection 7.1.1), it is possible to determine convolutional encoding matrices with large overall constraint length ν and near-optimum free distance.

Denote by $\mathcal{G}_{\text{sys},m}$ the set of all rate $R = 1/2$ systematic convolutional encoders $G_{\text{sys}}(D) = (1 \quad g(D))$ of memory m having an optimum column distance profile $\boldsymbol{d}^{\text{p}}$ (7.6). While all such encoding matrices share the same *minimum distance* $d_{\min} = d^{\text{r}}_m$, they differ in the number of (truncated) codewords $\boldsymbol{v}_{[0,m]}$ with Hamming weight $d_{\min}$.

In Table 7.5, the number of such encoding matrices with memory $m = 25, 26, \ldots, 40$, is given together with their minimum distance $d_{\min}$. Additionally, one (among many) systematic, rate $R = 1/2$ ODP encoding matrix with the fewest number of (truncated) codewords $\boldsymbol{v}_{[0,m]}$ of Hamming weight $d_{\min}$ is specified.

As previously mentioned in Subsection 7.1.1, multiplying a size $b \times c$ encoding matrix $G(D)$ of memory m with a $b \times b$ nonsingular matrix $T(D)$ and truncating all polynomials at degree m does not change its distance profile $\boldsymbol{d}^{\text{p}}$. Hence, multiplying a systematic, rate $R = 1/2$, encoding matrix

m	$\lvert\mathcal{G}_{\text{sys},m}\rvert$	$d_{\min}$	$\#\boldsymbol{v}_{[0,m]}$ of weight $d_{\min}$	$g(D)$
25	48	11	5	671145432
26	96	11	1	671145431
27	36	12	27	6711454574
28	72	12	8	6711454306
29	144	12	2	6711454306
30	12	13	43	67114545754
31	24	13	15	67114545754
32	48	13	4	67114545755
33	96	13	1	671145457554
34	12	14	34	671145457556
35	24	14	14	67114545447
36	48	14	5	6711454544704
37	96	14	2	6711454306444
38	16	15	31	6711454575564
39	32	15	12	6711454306444
40	64	15	3	67114545755712

Table 7.5: Number of systematic, ODP, rate $R = 1/2$ encoding matrices with memory $m = 25, 26, \ldots, 40$, together with their minimum distance $d_{\min}$ and one (among many) encoding matrix $G(D) = (1 \quad g(D))$ with the fewest number of (truncated) codewords $\boldsymbol{v}_{[0,m]}$ of Hamming weight $d_{\min}$.

$G_{\text{sys},m}(D) \in \mathcal{G}_{\text{sys},m}$ of memory m with a randomly chosen polynomial $t(D)$ of degree at most m and truncating their product at degree m, yields the nonsystematic, rate $R = 1/2$ ODP encoding matrix

$$G(D) = t(D) \begin{pmatrix} 1 & g(D) \end{pmatrix} \Big|_m = \begin{pmatrix} t(D) & t(D)g(D) \Big|_m \end{pmatrix} \tag{7.24}$$

Since the polynomial $t(D)$ can be freely chosen for a given memory m, there exist 2^m different nonsystematic encoding matrices $G_{t(D)}(D)$ based on a single systematic encoding matrix $G_{\text{sys}}(D)$ with the same rate and memory.

After applying the previously described rejection rules to the set of nonsystematic ODP encoding matrices, the BEAST is used to determine their free distance and Viterbi spectrum. For each memory between 25 and 40, the best

m	$g_0(D)$	$g_1(D)$	d_{free}	spectrum
25	746411326	544134532	27	14, 58, 120, 264, 569, 1406
26	525626523	645055711	28	24, 56, 131, 273, 736, 1723 [a]
27	7270510714	5002176664	28	1, 28, 66, 138, 366, 789
28	7605117332	5743521516	30	54, 0, 356, 0, 2148, 0 [a]
29	7306324763	5136046755	30	5, 47, 97, 211, 514, 1171 [a]
30	60425367524	45542642234	32	143, 0, 240, 0, 3870, 0
31	51703207732	66455246536	32	14, 65, 136, 336, 753, 1860
32	41273467427	70160662325	33	28, 61, 167, 372, 898, 2168
33	407346436304	711526703754	34	44, 0, 338, 0, 2081, 0
34	410174456276	702647441572	34	5, 35, 84, 229, 532, 1320
35	627327244767	463171036121	36	111, 0, 553, 0, 3309, 0
36	7664063056054	5707165143064	36	12, 53, 146, 360, 783, 1917
37	7267577012232	5011131253046	37	18, 73, 163, 381, 884, 2232
38	6660216760717	4131271202755	38	30, 83, 225, 524, 1152, 2761
39	42576550101264	66340614757214	38	2, 38, 97, 219, 575, 1324
40	26204724041271	37146123573117	40	78, 0, 532, 0, 6040, 0

(a) previously listed in [BHJK04].

Table 7.6: Viterbi spectra $n_{d_{\text{free}}+1}$, $i = 0, 1, \ldots, 5$, for nonsystematic rate $R = 1/2$ ODP convolutional encoders with memory $m = 25, 26, \ldots, 40$.

ODP encoding matrix is given in Table 7.6. Note that the rate $R = 1/2$ encoding matrices for memory $m = 26$, 28, and 29 coincide with the previously published results in [BHJK04]. For memories $m = 25$, 37, 31, 33, 34, 36, and 38 we found nonsystematic ODP encoding matrices with improved spectral components, while for memories $m = 30$, 32, 35, 37, and 40 the corresponding free distance have been improved.

Generalizing the previously described approaches to rate $R = 1/c$ convolutional codes, yields the systematic and nonsystematic, rate $R = 1/3$ ODP encoding matrices as presented in Tables 7.7 and 7.8, respectively, extending previously published results in [JZ99] to memory $m = 38$ and $m = 26$ for systematic and nonsystematic encoding matrices, respectively.

m	$g_1(D)$	$g_2(D)$	d_{free}	spectrum
29	6766735721	5312071307	33	3, 2, 5, 12, 16, 31 [a]
30	73251313564	51445320354	34	1, 6, 6, 5, 14, 31 [a]
31	71261062646	65376166062	36	9, 0, 22, 0, 52, 0
32	73251267417	51445036206	34	1, 1, 3, 3, 11, 18
33	732512674174	514450362064	36	1, 2, 5, 6, 8, 14
34	732512674172	514450362066	38	4, 0, 14, 0, 36, 0
35	732512674172	514450362067	39	4, 12, 3, 4, 43, 58
36	7325126741734	514450362066	40	12, 0, 25, 0, 78, 0
37	732512674173	5144503620676	40	8, 0, 15, 0, 55, 0
38	7127344503677	6530553473343	40	1, 0, 7, 0, 12, 0

(a) previously listed in [JZ99].

Table 7.7: Viterbi spectra $n_{d_{\text{free}}+1}$, $i = 0, 1, \ldots, 5$, for systematic rate $R = 1/3$ ODP convolutional encoders with memory $m = 29, 30, \ldots, 38$.

m	$g_0(D)$	$g_1(D)$	$g_2(D)$	d_{free}	spectrum
18	4551064	6247274	7730474	34	28, 0, 44, 0, 182, 0 [a]
19	5531236	6151572	7731342	35	8, 18, 29, 32, 54, 78 [a]
20	5361071	6561265	7237543	36	3, 14, 21, 25, 48, 92
21	54564334	60721124	76366644	38	11, 0, 55, 0, 129, 0
22	42441722	66766532	72142746	39	13, 14, 24, 36, 50, 83
23	47446663	66164457	77054535	40	2, 9, 17, 36, 53, 86
24	105264111	157347063	176625137	42	10, 14, 36, 37, 64, 116
25	331576613	211707125	365742567	44	41, 0, 45, 0, 206, 0
26	515357647	644225425	740663661	44	1, 10, 19, 35, 42, 78

(a) previously listed in [JZ99].

Table 7.8: Viterbi spectra $n_{d_{\text{free}}+1}$, $i = 0, 1, \ldots, 5$, for nonsystematic rate $R = 1/3$ ODP convolutional encoders with memory $m = 18, 19, \ldots, 36$.

Acronyms

APP *A Posteriori* Probability

AWGN Additive White Gaussian Noise

BD Blue-ray Disk

BEAST Bidirectional Efficient Algorithm for Searching code Trees

BER Bit decoding Error Rate

BP Belief Propagation

BPSK Binary Phase-Shift Keying

BSC Binary Symmetric Channel

BSS Binary Symmetric Source

CCF Controller Canonical Form

CD Compact Disk

CWAM Complete Weight Adjacency Matrix

DMC Discrete Memoryless Channel

DSL Digital Subscriber Line

DVD Digital Versatile Disk

FIR Finite Impulse Response

GSM Global System for Mobile Communications

HDD Hard Disk Drive

IEEE Institute of Electrical and Electronics Engineers

IIR Infinite Impulse Response

LDPC Low-Density Parity-Check

LFSR Linear Feedback Shift Register

LTE 3GPP Long Term Evolution

LTI Linear Time Invariant

MAP Maximum *A Posteriori*

MB Minimal-Basic

MD Minimum (Hamming) Distance

ML Maximum-Likelihood

OCF Observer Canonical Form

ODP Optimum Distance Profile

OFD Optimum Free Distance

PDF Probability Density Function

PSD Power Spectral Density

QC Quasi-Cyclic

RC Repetition Code

REF Row Echelon Form

RREF Reduced Row Echelon Form

SNR Signal-to-Noise Ratio

SPC Single Parity-Check Code

SSD Solid State Flash Drive

STS Steiner Triple System

TB Tailbiting

TB Truncation

UMTS Universal Mobile Telecommunications System

VG Varshamov-Gilbert

WAM Weight Adjacency Matrix

WE Weakly Equivalent

WiMAX Worldwide Interoperability for Microwave Access

ZT Zero-Tail

References

[AG92] K. A. S. Abdel-Ghaffar, »On unit constraint-length convolutional codes,« *IEEE Transactions on Information Theory*, vol. IT-38, no. 1, pp. 200 – 206, January 1992.

[AMB04] A. Amat, G. Montorsi, and S. Benedetto, »Design and decoding of optimal high-rate convolutional codes,« *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 867 – 881, May 2004.

[BB93] Y. Berger and Y. Be'ery, »Bounds on the trellis size of linear block codes,« *IEEE Transactions on Information Theory*, vol. 39, no. 1, pp. 203 – 209, January 1993.

[BBL+95] M. R. Best, M. V. Burnashev, Y. Levy, A. Rabinovich, P. C. Fishburn, A. R. Calderbank, and D. J. Costello, Jr., »On a technique to calculate the exact performance of a convolutional code,« *IEEE Transactions on Information Theory*, vol. 41, no. 2, pp. 441 – 447, March 1995.

[BC90] M. V. Burnashev and D. L. Cohn, »Symbol error probability for convolutional codes,« *Problems of Information Transmission*, vol. 26, no. 4, pp. 289 – 298, 1990.

[BCJR74] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, »Optimal decoding of linear codes for minimizing symbol error rate,« *IEEE Transactions on Information Theory*, vol. IT-20, no. 2, pp. 284 – 287, March 1974.

[BHJ+] I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, »Degree matrices for QC LDPC codes,« Online: `http://www.eit.lth.se/goto/QC_LDPC_Codes`.

[BHJ+10] I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, »New low-density parity-check codes with large girth based on hypergraphs,« in *Proc. IEEE International Symposium on Information Theory (ISIT'10)*, Austin, USA, June 13 – 18, 2010, pp. 819 – 823.

[BHJ+11] I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, »Some voltage graph-based LDPC tailbiting codes with large girth,« in *Proc. IEEE International Symposium on Information Theory (ISIT'11)*, St. Petersburg, Russia, July 31 – August 5, 2011, pp. 732 – 736.

[BHJ+12] I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, »Searching for voltage graph-based LDPC tailbiting codes with large girth,« *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2265 – 2279, April 2012.

[BHJK01] I. E. Bocharova, M. Handlery, R. Johannesson, and B. D. Kudryashov, »A BEAST for prowling in trees,« in *Proc. 39th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, USA, October 3 – 5, 2001, pp. 52 – 61.

[BHJK04] I. E. Bocharova, M. Handlery, R. Johannesson, and B. D. Kudryashov, »A BEAST for prowling in trees,« *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1295 – 1302, June 2004.

[BHJK05] I. E. Bocharova, M. Handlery, R. Johannesson, and B. D. Kudryashov, »BEAST decoding of block codes obtained via convolutional codes,« *IEEE Transactions on Information Theory*, vol. 51, no. 5, pp. 1880 – 1891, May 2005.

[BHJK10] I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »On weight enumerators and MacWilliams identity for convolutional codes,« in *Proc. Information Theory and Applications Workshop (ITA'10)'*, San Diego, USA, January 31 – February 5, 2010.

[BHJK11a] I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »Woven convolutional graph codes with large free distances,« *Problems of Information Transmission*, vol. 47, no. 1, pp. 3 – 18, 2011.

[BHJK11b] I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »Another look at the exact bit error probability for Viterbi decoding of convolutional codes,« in *International Mathematical Conference '50 Years of IPPI'*, Moscow, Russia, July 25 – 29, 2011.

[BHJK11c] I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »Double-Hamming based QC LDPC codes with large minimum distance,« in *Proc. IEEE International Symposium on Information Theory (ISIT'11)*, St. Petersburg, Russia, July 31 – August 5, 2011, pp. 923 – 927.

[BHJK11d] I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »On the exact bit error probability for Viterbi decoding of convolutional codes,« in *Proc. Information Theory and Applications Workshop (ITA'11)*, San Diego, USA, February 6 – 11, 2011.

[BHJK12a] I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »A note on duality and MacWilliams-type identities for convolutional codes,« *Problems of Information Transmission*, vol. 48, no. 1, 2012.

[BHJK12b] I. E. Bocharova, F. Hug, R. Johannesson, and B. D. Kudryashov, »A closed form expression for the exact bit error probability for Viterbi decoding of convolutional codes,« accepted for publication in *IEEE Transactions on Information Theory*, vol. 58, 2012.

[BJKL04] I. E. Bocharova, R. Johannesson, B. D. Kudryashov, and M. Lončar, »BEAST decoding of block codes,« *European Transactions on Telecommunications*, vol. 15, no. 4, pp. 297 – 305, July 2004.

[BK97] I. E. Bocharova and B. D. Kudryashov, »Rational rate punctured convolutional codes for soft-decision Viterbi decoding,« *IEEE Transactions on Information Theory*, vol. 43, no. 4, pp. 1305 – 1313, July 1997.

[BKJZ07] I. E. Bocharova, B. D. Kudryashov, R. Johannesson, and V. V. Zyablov, »Asymptotically good woven codes with fixed constituent convolutional codes,« in *Proc. IEEE International Symposium on Information Theory (ISIT'07)*, Nice, France, June 24 – 29, 2007, pp. 2326 – 2330.

[BKJZ10] I. E. Bocharova, B. D. Kudryashov, R. Johannesson, and V. V. Zyablov, »Woven graph codes: Asymptotic performances and examples,« *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 121 – 129, January 2010.

[BKS09] I. E. Bocharova, B. D. Kudryashov, and R. V. Satyukov, »Graph-based convolutional and block LDPC codes,« *Problems of Information Transmission*, vol. 45, no. 4, pp. 357 – 377, 2009.

[BKSS09] I. E. Bocharova, B. D. Kudryashov, R. V. Satyukov, and S. Stiglmayr, »Short quasi-cyclic LDPC codes from convolutional codes,« in *Proc. IEEE International Symposium on Information Theory (ISIT'09)*, Seoul, South-Korea, June 28 – July 3, 2009, pp. 551 – 555.

[Bla85] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Boston, USA: Addison-Wesley Publishing Co., 1985.

[Bos98] M. Bossert, *Kanalcodierung*, 2nd ed. Stuttgart, Germany: Teubner, 1998.

[Bro93] W. C. Brown, *Matrices over commutative rings*. New York, USA: Marcel Dekker, 1993.

[CCG79] J. Cain, G. Clark, and J. Geist, »Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding (corresp.),« *IEEE Transactions on Information Theory*, vol. IT-25, no. 1, pp. 97 – 100, January 1979.

[CHL97] J.-J. Chang, D.-J. Hwang, and M.-C. Lin, »Some extended results on the search for good convolutional codes,« *IEEE Transactions on Information Theory*, vol. 43, no. 5, pp. 1682 – 1697, September 1997.

[Cos69] D. J. Costello, Jr., »A construction technique for random-error-correcting convolutional codes,« *IEEE Transactions on Information Theory*, vol. IT-15, no. 5, pp. 631 – 636, May 1969.

[DLZ+09] L. Dolecek, P. Lee, Z. Zhang, V. Anantharam, B. Nikolic, and M. Wainwright, »Predicting error floors of structured LDPC codes: Deterministic bounds and estimates,« *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 239 – 246, August 2009.

[EG10] M. Esmaeili and M. Gholami, »Structured quasi-cyclic LDPC codes with girth 18 and column-weight $J \geq 3$,« *International Journal of Electronics and Communications (AEU)*, vol. 64, no. 3, pp. 202 – 217, March 2010.

[Eli55] P. Elias, »Coding for noisy channels,« in *Proc. IRE Convention Record*, vol. 4, 1955, pp. 37 – 46.

[Eur08] *Digital Video Broadcasting (DVB)*, European Telecommunications Standards Institute ETSI EN 302 755, Rev. 1.1.1, July 2008.

[Eur09] *Digital Video Broadcasting (DVB)*, European Telecommunications Standards Institute ETSI EN 302 307, Rev. 1.2.1, August 2009.

[For67] G. D. Forney, Jr., »Review of random tree codes,« *NASA Ames Research Center, Moffett Field, California, Contract NAS2-3637, NASA CR 73176, Final Report, Appendix A*, December 1967.

[For88] G. D. Forney, Jr., »Coset codes II: Binary lattices and related codes,« *IEEE Transactions on Information Theory*, vol. IT-34, no. 5, pp. 1152 – 1187, September 1988.

[For09] G. D. Forney, Jr., »MacWilliams identities for codes on graphs,« in *Proc. IEEE Information Theory Workshop (ITW'09)*, Taormina, Italy, October 11 – 16, 2009, pp. 120 – 124.

[For11] G. D. Forney, Jr., »Codes on Graphs: Duality and MacWilliams identities,« *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1382 – 1397, March 2011.

[Fos04] M. P. C. Fossorier, »Quasi-cyclic low-density parity-check codes from circulant permutation matrices,« *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1788 – 1793, August 2004.

[Gal62] R. G. Gallager, »Low-density parity-check codes,« *IRE Transactions on Information Theory*, vol. IT-8, pp. 21 – 28, January 1962.

[Gal63] R. G. Gallager, »Low-density parity-check codes,« Ph.D. dissertation, MIT Press, Cambridge, USA, 1963.

[GLS08] H. Gluesing-Luerssen and G. Schneider, »On the MacWilliams identity for convolutional codes,« *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1636 – 1550, April 2008.

[GLS09] H. Gluesing-Luerssen and G. Schneider, »A MacWilliams identity for convolutional codes: The general case,« *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 2920 – 2930, July 2009.

[Har92] G. H. Hardy, *Divergent Series*, 2nd ed. Providence, USA: American Mathematical Society, November 1992.

[HBJK10] F. Hug, I. E. Bocharova, R. Johannesson, and B. D. Kudryashov, »A rate R=5/20 hypergraph-based woven convolutional code with free distance 120,« *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1618 – 2623, April 2010.

[HJ71] J. A. Heller and I. M. Jacobs, »Viterbi decoding for satellite and space communication,« *IEEE Transactions on Communications*, vol. COM-19, no. 5, pp. 835 – 848, October 1971.

[HJ90] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, England: Cambridge University Press, February 1990.

[HJZ02] S. Höst, R. Johannesson, and V. V. Zyablov, »Woven convolutional codes I: Encoder properties,« *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 149 – 161, January 2002.

[IEE05] *Air Interface for Fixed and Mobile Broadband Wireless Access Systems*, IEEE P802.16e/D12 Draft, October 2005.

[JBHH11] D. Johnsson, F. Bjärkeson, M. Hell, and F. Hug, »Searching for new convolutional codes using the cell broadband engine architecture,« *IEEE Communications Letters*, vol. 15, no. 5, pp. 560 – 562, May 2011.

[JW01a] S. J. Johnson and S. R. Weller, »Construction of low-density parity-check codes from Kirkman triple systems,« in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'01)*, vol. 2, San Antonio, USA, November 25 – 29, 2001, pp. 970 – 974.

[JW01b] S. J. Johnson and S. R. Weller, »Regular low-density parity-check codes from combinatorial designs,« in *Proc. IEEE Information Theory Workshop (ITW'01)*, Cairns, Australia, September 2 – 7, 2001, pp. 90 – 92.

[JZ99] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. Piscataway, USA: IEEE Press, 1999.

[Kir47] T. P. Kirman, »On a problem in combinatorics,« *Cambridge and Dublin mathematical Journal*, vol. 2, pp. 191 – 204, 1847.

[KLF01] Y. Kou, S. Lin, and M. P. C. Fossorier, »Low-density parity-check codes based on finite geometries: A rediscovery and new results,« *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711 – 2736, November 2001.

[KNCS07] S. Kim, J.-S. No, H. Chung, and D.-J. Shin, »Quasi-cyclic low-density parity-check codes with girth larger than 12,« *IEEE Transactions on Information Theory*, vol. 53, no. 8, pp. 2885 – 2891, August 2007.

[KS95] F. R. Kschischang and V. Sorokine, »On the trellis structure of block codes,« *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 1924 – 1937, November 1995.

[KW08] C. A. Kelley and J. L. Walker, »LDPC codes from voltage graphs,« in *Proc. IEEE International Symposium on Information Theory (ISIT'08)*, Toronto, Canada, July 6 – 11, 2008, pp. 792 – 796.

[LC04] S. Lin and D. J. Costello, Jr., *Error Control Coding*, 2nd ed. Upper Saddle River, USA: Prentice Hall, 2004.

[LTZ04] M. Lentmaier, D. V. Truhachev, and K. S. Zigangirov, »Analytic expressions for the bit error probabilities of rate-1/2 memory 2 convolutional encoders,« *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1303 – 1311, June 2004.

[Mas74] J. L. Massey, »Coding and modulation in digital communications,« in *Proc. International Zurich Seminar on Digital Communications*, Zurich, Switzerland, March 1974, pp. E2(1) – E2(4).

[Mas78] J. L. Massey, »Foundations and methods of channel coding,« in *Proc. International Conference on Information Theory and Systems*, vol. 65, Berlin, Germany, September 1978.

[Mas84] J. L. Massey, »The how and why of channel coding,« in *Proc. International Zurich Seminar on Digital Communications*, Zurich, Switzerland, March 1984, pp. 67 – 73.

[McE96] R. J. McEliece, »On the BCJR trellis for linear block codes,« *IEEE Transactions on Information Theory*, vol. 42, no. 4, pp. 1072 – 1092, July 1996.

[McE98] R. J. McEliece, »How to compute weight enumerators for convolutional codes,« in *Communications and Coding (P. G. Farrell 60th Birthday Celebration)*, M. Darnell and B. Honary, Eds. Hoboken, USA: Wiley-Blackwell, 1998, pp. 121 – 141.

[MD99] D. J. C. MacKay and M. C. Davey, »Evaluation of Gallager codes for short block length and high rate applications,« in *Codes, Systems and Graphical Models*. Berlin, Germany: Springer-Verlag, 1999, pp. 113 – 130.

[MKL06] O. Milenkovic, N. Kashyap, and D. Leyba, »Shortened array codes of large girth,« *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3707 – 3722, August 2006.

[Mor70] T. N. Morrissey, Jr., »Analysis of decoders for convolutional codes by stochastic sequential machine methods,« *IEEE Transactions on Information Theory*, vol. IT-16, no. 4, pp. 460 – 469, July 1970.

[MS77] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, Netherlands: North Holland, 1977.

[Mud88] D. J. Muder, »Minimal trellises for block codes,« *IEEE Transactions on Information Theory*, vol. IT-34, no. 5, pp. 1049 – 1053, September 1988.

[OCC93] I. M. Onyszchuk, K.-M. Cheung, and O. Collins, »Quantization loss in convolutional decoding,« *IEEE Transactions on Communications*, vol. 41, no. 2, pp. 261 – 265, February 1993.

[O'S06] M. E. O'Sullivan, »Algebraic construction of sparse matrices with large girth,« *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 718 – 727, February 2006.

[Paa74] E. Paaske, »Short binary convolutional codes with maximal free distance for rates 2/3 and 3/4 (corresp.),« *IEEE Transactions on Information Theory*, vol. IT-20, no. 5, pp. 683 – 689, January 1974.

[Pea88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 2nd ed. San Francisco, USA: Morgan Kaufmann, 1988.

[PS08] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. New York, USA: McGraw Hill, 2008.

[RST91] I. Reed, M. Shih, and T. Truong, »VLSI design of inverse-free Berlekamp-Massey algorithm,« *Computers and Digital Techniques, IEE Proceedings-E*, vol. 138, no. 5, pp. 295 – 298, September 1991.

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

[SFRU01] C. Sae-Young, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, »On the design of low-density parity-check codes within 0.0045 db of the shannon limit,« *IEEE Communications Letters*, vol. 5, no. 2, pp. 58 – 60, February 2001.

[Sha48] C. E. Shannon, »A mathematical theory of communication,« *Bell System Technical Journal*, vol. 27, pp. 379 – 423 (Part I), 623 – 656 (Part II), 1948.

[SM77] J. B. Shearer and R. J. McEliece, »There is no MacWilliams identity for convolutional codes,« *IEEE Transactions on Information Theory*, vol. IT-23, no. 6, pp. 775 – 776, November 1977.

[SV11] R. Smarandache and P. O. Vontobel, »Quasi-cyclic LDPC codes: Influence of proto- and Tanner-graph structure on minimum Hamming distance upper bounds,« *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 585 – 607, February 2011.

[SZ94] V. Sidorenko and V. Zyablov, »Decoding of convolutional codes using a syndrome trellis,« *IEEE Transactions on Information Theory*, vol. 40, no. 5, pp. 1663 – 1666, September 1994.

[TAD04] J. Thorpe, K. Andrews, and S. Dolinar, »Methodologies for designing LDPC codes using protographs and circulants,« in *Proc. IEEE International Symposium on Information Theory (ISIT'04)*, Chicago, USA, June 27 – July 2, 2004, p. 238.

[Tan81] R. M. Tanner, »A recursive approach to low-complexity codes,« *IEEE Transactions on Information Theory*, vol. IT-27, no. 5, pp. 533 – 546, September 1981.

[TLF06] H.-H. Tang, M.-C. Lin, and B. F. U. Filho, »Minimal trellis modules and equivalent convolutional codes,« *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3738 – 3746, April 2006.

[TLUF06] H.-H. Tang, M.-C. Lin, and B. Uchoa-Filho, »Minimal trellis modules and equivalent convolutional codes,« *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3738 – 3746, August 2006.

[TSF01] R. M. Tanner, D. Sridhara, and T. Fuja, »A class of group-structured LDPC codes,« in *Proc. 6th International Symposium on Communication Theory and Applications*, Ambleside, England, July 15 – 20, 2001, pp. 365 – 370.

[TSS+04] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, Jr., »LDPC block and convolutional codes based on circulant matrices,« *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 2966 – 2984, December 2004.

[Van74] L. Van de Meeberg, »A tightened upper bound on the error probability of binary convolutional codes with Viterbi decoding,« *IEEE Transactions on Information Theory*, vol. IT-20, no. 3, pp. 389 – 391, May 1974.

[Var98] A. Vardy, »Trellis Structure of Codes,« in *Handbook of Coding Theory*, R. A. Brualdi, W. C. Huffman, and V. Pless, Eds. Amsterdam, The Netherlands: Elsevier Science & Technology, November 1998, pp. 1989 – 2018.

[Vit67] A. J. Viterbi, »Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,« *IEEE Transactions on Information Theory*, vol. IT-13, no. 2, pp. 260 – 269, April 1967.

[Vit71] A. J. Viterbi, »Convolutional codes and their performance in communication systems,« *IEEE Transactions on Communications*, vol. COM-19, no. 5, pp. 751 – 772, October 1971.

[WJ65] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*. Hoboken, USA: Wiley, 1965.

[Wol78] J. Wolf, »Efficient maximum likelihood decoding of linear block codes using a trellis,« *IEEE Transactions on Information Theory*, vol. IT-24, no. 1, pp. 76 – 80, January 1978.

[WYD08] Y. Wang, J. S. Yedidia, and S. C. Draper, »Construction of high-girth QC-LDPC codes,« in *Proc. 5th International Symposium on Turbo Codes and Related Topics*, Lausanne, Switzerland, September 1 – 5, 2008, pp. 180 – 185.

[Wym07] H. Wymeersch, *Iterative Receiver Design*. Cambridge, England: Cambridge University Press, 2007.

[ZW10] G. Zhang and X. Wang, »Girth-12 quasi-cyclic LDPC codes with consecutive lengths,« *arXiv: 1001.3916v1*, January 2010.