On Woven Convolutional Codes

Höst, Stefan

1999

On Woven Convolutional Codes

Stefan Höst

Lund University

Ph. D. Thesis, September 24, 1999

Stefan Höst
Department of Information Technology
Lund University
P.O. Box 118
S-221 00 Lund, Sweden
e-mail: stefan.host@it.lth.se
http://www.it.lth.se/

Printed in Sweden
KFS AB, 1999

ISRN: LUTEDX/TEIT-99/1013-SE
ISBN: 91-7167-016-5

Contents

Preface

This doctoral thesis is the result of the five years I spent as a Ph. D. student at the Department of Information Technologies at Lund University. In total the thesis can be seen as a chronological presentation of my research work. There are of course parts of the early work that have been added later on, but the main ideas follow this order.

Shortly after I started my studies in December 1994 Professor Viktor Zyablov visited us for a two months period. That was the beginning of a fruitful collaboration between him, me, and my supervisor Professor Rolf Johannesson. Since then he has visited us for about two months every year.

During the years the material in this thesis has been presented at conferences and workshops. Here I would like to mention some of the places where it occurs in the proceedings:

- The 7th joint Swedish-Russian Workshop on Information Theory, St.-Petersburg, Russia, 1995.
- The 5th International Workshop on Algebraic and Combinatorial Coding Theory, Sozopol, Bulgaria, 1996.
- The 1st IEEE Workshop on Convolutional Codes, Essen, Germany, 1996.
- The 4th International Symposium on Communication Theory and Applications[1], Lake District, UK, 1997.
- 1997 IEEE International Symposium on Information Theory, Ulm, Germany, 1997.

- 1998 IEEE International Symposium on Information Theory, Cambridge, Mass., USA, 1998.
- The 6th International Workshop on Algebraic and Combinatorial Coding Theory[1], Pskov, Russia, 1998.
- The 2nd IEEE Workshop on Convolutional Codes, Kamp Lintfort, Germany, 1998.

Parts have also been published, accepted for publication, or submitted for possible publication in transactions and books:

- The chapter Cascaded Convolutional Codes, in *Communication and Coding*, published in honor of Paddy G. Farell on the occasion of his 60th birthday, 1998.
- The paper Nonequivalent Convolutional Codes Obtained from Equivalent Constituent Encoders, in *Problemy Peredachi Informatsii*, 1998.
- The paper Active Distances for Convolutional Codes, in *IEEE Transactions on Information Theory*, 1999.
- The paper On the Error Exponent for Woven Convolutional Codes With Outer Warp, in *IEEE Transactions on Information Theory*, 1999.
- The paper Asymptotic Distance Properties of Binary Woven Convolutional Codes, to appear in *Problemy Peredachi Informatsii*, 1999.
- The paper Woven Convolutional Codes: Encoder Properties, submitted to *IEEE Transactions on Information Theory*, 1999.

Acknowledgments

A Ph. D. thesis is hardly ever, and should not be, the work of a single student sitting in his room with the door closed. The work presented here has been made possible by several people. There are especially two persons that have meant a lot for my understanding of the problems concerning concatenation of convolutional codes, and that I would like to thank.

First, I would like to thank my supervisor Rolf Johannesson for introducing me to the area of convolutional codes and concatenation of convolutional codes. I also like to thank him for the many discussions on the fundamental secrets of convolutional coding. Second, as mentioned before, I would like to thank Viktor Zyablov for a rewarding cooperation and many discussions on

[1] Not presented by the author of this thesis.

the ideas of concatenated codes. I have developed the greatest respect for their knowledge and intuition in this area.

I also would like to thank Volodia Sidorenko with whom I worked when he visited Lund in both 1995 and 1997.

I have benefited a lot from the discussions on woven convolutional codes with Ralph Jordan.

Then, I would like to thank the staff of the Department of Information Technology for making it such an inspiring environment to work in. To mention some, I would like to thank Joakim Persson for help and inspiration during the first years of my studies. I also would like to thank Emma Wittenmark for many discussions on convolutional codes and the everyday life at the department. A special thank also to Jan Åberg who has always taken the time for a chat about life in general or the development of a thesis. I am also indebted to Thomas Johansson and Michael Lentmaier for comments and discussions on the decoding part of my work.

The final year of my studies I spent at the Department of Access Technologies and Signal Processing at Ericsson Radio Systems AB in Stockholm. It meant a lot to me and I immediately felt welcomed. Especially I am grateful to Johan Nyström, Jan Färjh, and Mikael Höök who made it possible for me to stay there.

I am also grateful to my family who always supported and helped me throughout my studies. Finally, I would like to thank Camilla, for always encouraging me in my work and for listening patiently to my, in many cases cryptic, explanations of some new ideas I have got; I love you.

Stefan Höst

1

Introduction

Today we use digital communication systems without taking much notice about it, e.g., when we listen to music on a CD, make a call using a cellular phone, or send an e-mail. Soon we will watch television programs that are broadcasted digitally. Unfortunately, contrary to the impression in the media, digital signals are affected by noise as much as analog signals. We still have disturbances in the transmission and we still have errors in the received signals. It can be due to scratches on the surface of the CD or background noise for a satellite link, just to mention some sources of noise.

This motivates the use of error correcting codes. In a digital system it is quite easy, compared to analog systems, to add redundant data to the transmitted information that makes it possible to detect and correct some of the occurred errors. There will always be patterns of errors that we cannot correct and the codes can be more or less effective in its work, which often result in more or less complex encoders and decoders. One way to construct good codes is to combine several simpler ones, i.e., to concatenate codes. In this thesis a new type of concatenated convolutional codes, called woven convolutional codes, in which the constituent codes are woven together will be described.

In the first section of this chapter we will consider a simple, but general, block diagram of a communication system and in Section 1.2 the ideas of error correcting codes are introduced. Finally, in Section 1.3 the outline of the thesis is given.

1.1. A Communication System

In 1948 Claude E. Shannon published his paper *A Mathematical Theory of Communication* [45]. This was the beginning of the information theory. Shannon showed that without loss of optimality a communication system can be considered to be digital and that we can divide the transmission into two parts, source coding and channel coding. A block diagram of such model is shown in Figure 1.1. The source in the figure produces the information that will be sent, e.g., speech, pictures, music, computer programs etc.

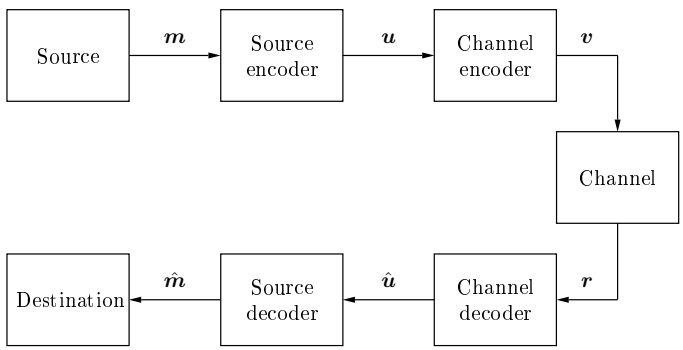

Figure 1.1: The block diagram of a communication system.

Almost without exceptions, the messages from the source contains redundancy. For example, we can in most cases still read a text were every fourth letter is missing or even replaced. The source encoder removes the redundancy in such a way that the messages can be recreated by the receiver. There are specific source coding algorithms for different kinds of data. For example, in computers text files are often compressed with a variant of the Lempel-Ziv algorithm, e.g., zip, while pictures are often stored with the jpeg standard. It can be devastating if the compressed data are corrupted by errors. To cope with this, the channel encoder introduces redundancy. This is added in a controlled way that follows some simple[1] rules that a computer can handle. Then some of the errors that occur on the channel can be detected and even corrected. In this thesis we will only be interested in channel coding and, therefore, assume perfect source encoding, which means that we have equally likely symbols in the information sequence.

[1] Simple compared to for example natural languages.

The channel of the communication system is a statistical model of the influence the noise has on the transmitted data. The simplest, and also most well-known, channel is the binary symmetric channel. Each bit from the transmitter side has a probability, p, of being received erroneously. Thus, if a '0' is transmitted we will receive a '1' with probability p and a '0' with probability $1-p$. Similarly, if a '1' is transmitted we receive a '0' with probability p and a '1' with probability $1-p$. This can be illustrated graphically as in Figure 1.2.

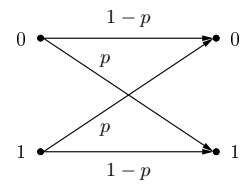

Figure 1.2: The binary symmetric channel.

Another, somewhat controversial, result in [45] was the interpretation of the channel capacity, C. It was stated that it is possible to achieve arbitrary low error probability as long as the rate—information bits per code bits—is less than the channel capacity. For a binary symmetric channel the capacity is

$$C = 1 - h(p), \tag{1.1}$$

where[2]

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

is the binary entropy function.

A straight forward way to improve the channel of Figure 1.2 is to use a channel with more than two outputs. In Figure 1.3 a discrete memoryless channel with binary inputs and 8-ary outputs is shown. The output symbols are ordered with increasing certainty of the transmitted bit. If the symbol 0_4 is received we can be almost certain that is was a '0' transmitted, while if we receive 0_1 it is much more uncertain. In this way we obtain more information about the noise on the channel and the receiver can make more reliable estimates of the transmitted symbols. Often, such channels are assumed to be symmetric, i.e., $P(0_4|0) = P(1_4|1)$, $P(0_3|0) = P(1_3|1)$, $\ldots$, $P(1_4|0) = P(0_4|1)$.

The channels described in Figures 1.2 and 1.3 are both memoryless, i.e., the transition probabilities are independent from time to time. This is a

[2] Here and hereafter $\log(\cdot)$ denotes the logarithm of base two.

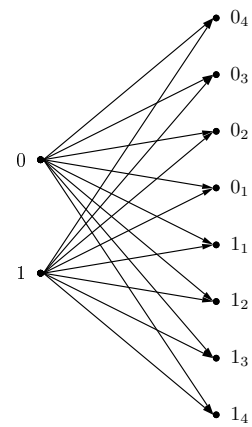

Figure 1.3: A binary input, 8-ary output discrete memoryless channel.

good model when describing for example satellite communication. However, if we like to model the channel for a mobile communication system like the transmissions between a cellular phone and a base station it becomes more complicated. If there is a house in between it is not likely to disappear within the next milliseconds. We get a model with memory and it depends on factors like the surrounding areas and the speed of the object.

1.2. Channel Coding

The simplest error correcting code is the repetition code. Instead of transmitting one binary symbol once we transmit it three times. If one of the transmissions is erroneously received we can detect this and still be able to say what was transmitted. In this coding scheme we transmit three code symbols for each information symbol, and we say that the rate of the code is $R = 1/3$.

If blocks of several symbols are used it is possible to construct more efficient codes. Consider the case when an information word consists of four bits, $\boldsymbol{u} = (u_1 u_2 u_3 u_4)$. Given the generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{1.3}$$

the codewords are constructed as $\boldsymbol{v} = \boldsymbol{u}G$. Thus, the codeword is the binary 7-tuple $\boldsymbol{v} = (u_1u_2u_3u_4v_5v_6v_7)$, where the parity symbols are calculated as

$$\begin{cases} v_5 = u_1 + u_3 + u_4 \\ v_6 = u_1 + u_2 + u_4 \\ v_7 = u_1 + u_2 + u_3 \end{cases} \tag{1.4}$$

and the additions are carried out modulo 2. The complete list of information words and codewords is given in Table 1.1. The rate is $R = 4/7$ which is significantly higher than for the repetition code. Since any two codewords differ in at least three positions we can still correct all single errors.

$\boldsymbol{u}$	$\boldsymbol{v}$	$\boldsymbol{u}$	$\boldsymbol{v}$
0000	0000000	1000	1000111
0001	0001110	1001	1001001
0010	0010101	1010	1010010
0011	0011011	1011	1011100
0100	0100011	1100	1100100
0101	0101101	1101	1101010
0110	0110110	1110	1110001
0111	0111000	1111	1111111

Table 1.1: The encoding scheme generated by (1.3).

One way to construct good codes is to combine several simpler encoders such that they work together. Consider a simple parity check code where for each four bit information word a parity bit is attached such that the five bit codeword has an even number of ones. For example, the information word $\boldsymbol{u} = (0110)$ is encoded into the codeword $\boldsymbol{v} = (01100)$ and $\boldsymbol{u} = (1110)$ into $\boldsymbol{v} = (11101)$. This construction can only detect one error but not correct it. To construct a concatenated encoder from this we let the information word be a block of 4×4 bits. Attach first a parity symbol to each row, and then to each column, e.g., the information block

$$\boldsymbol{u} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \tag{1.5}$$

is encoded into

$$\boldsymbol{v} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & \end{pmatrix}. \tag{1.6}$$

The construction has the encoding rate $R = 16/24 = 2/3$. If one symbol is erroneously received we can correct it since we detect both the row and the column. Also, some patterns of two errors can be corrected (if they are located in different rows and columns).

We will conclude this section with some historic landmarks that are important for the contents of this thesis. In [45] Shannon showed that there exist error correcting codes and how much we can gain by them. The first constructed code was the Hamming codes [21] that were published in 1950[3]. In this thesis we will consider convolutional codes. They were first introduced by Elias in 1955 [10]. The idea to concatenated codes was presented by Forney in his Ph.D. thesis [11]. Forney also invented the concept of the trellis [12,14] in 1967 and began the structural analysis of convolutional codes in 1970 [13]. Concatenated convolutional codes are often suitable to be decoded by an iterative scheme. Iterative decoding is a topic that dates back in the history of coding theory, e.g., [17], but until recently we did not have the computer power to exploit it. For concatenated convolutional codes it was first performed for Turbo codes in 1993 by Berrou, Glavieux, and Thitimajshima in [6] (see also [5]).

1.3. Outline of the Thesis

This thesis is divided into three major parts, viz., the active distances, cascaded convolutional codes, and woven convolutional codes. The part about active distance builds upon [29]. Most of the material about cascaded convolutional codes can be found in [23,30,32]. Woven convolutional codes were introduced in [25] and further developed in [27]. These studies will continue here and in [26].

In Chapter 2 convolutional codes is introduced together with its generator matrices and encoders. The basic properties of their structure and distances are defined. The concept of the trellis, which is used several times throughout the thesis, is defined, and a short review on time-varying convolutional codes is given.

[3] Hamming actually invented the codes earlier and they were mentioned in [45].

In, for example, [36,50], the extended distances [46] were used to analyze concatenation of convolutional codes with unit memory. Chapter 3 is devoted to the active distances which is a generalization of the extended distances to convolutional codes with arbitrary memories. It is shown that important properties of convolutional codes can be obtained from the active distances. Also lower bounds on the active distances are derived.

To start the analyzes of concatenated convolutional codes the simplest construction is considered in Chapter 4, viz., a cascade of two convolutional encoders. First, structural properties for cascaded convolutional codes are derived for two cases: with and without matched rates. In Sections 4.4 and 4.5 specific examples are studied and analyzed. Finally, lower bounds on the active distances are derived for fixed time-invariant and for time-varying cascaded convolutional codes.

In Chapter 5 woven convolutional codes are introduced. In its general form, the twill, it is a cascade of two sets of parallel convolutional encoders. Also two degenerated but important special cases are considered, viz., woven convolutional codes with outer warp and with inner warp. The generator matrices together with some important structural properties of the building blocks, the warps, are derived. Then, the generator matrices for the woven construction follows together with their structural properties. The free distance and the active distances are bounded for fixed constituent codes. Woven convolutional codes can, as mentioned, be seen as a generalization of cascaded convolutional codes. It can also be looked upon as concatenated convolutional codes inspired by generalized concatenated codes [7].

Chapter 6 is devoted to decoding of woven convolutional codes. The BCJR algorithm is first rederived. It is used to decode the constituent codes. Then an iterative decoding scheme for woven convolutional codes is described and simulation results are given.

The structure of woven convolutional codes invites to theoretical analyzes. Chapter 7 gives a survey of bounds on time-varying woven convolutional codes. First a short review of the error exponents for block and convolutional codes is given. This leads to the error exponents for woven convolutional codes with outer and inner warp. Finally, lower bounds on the active distances for woven convolutional codes with outer and inner warp are given. This chapter has the structure of an overview and the bounds are given without proofs.

Finally, some concluding remarks and a short discussion on future investigations are given in Chapter 8.

2

Convolutional Codes

To grasp the contents of this thesis, some knowledge about binary convolutional codes is required. The basic theory used in the thesis is described in this chapter. It is important to distinguish between the convolutional code, the convolutional encoder, and the convolutional generator matrix. In Section 2.1, the convolutional code is defined together with the generator matrices and the encoders. In Section 2.2, important structural properties of the generator matrices and encoders are explained and, in Section 2.3, some well-known distance measures for convolutional codes are considered. In Section 2.4, the trellis structure is introduced.

Sections 2.1 to 2.4 deals with time-invariant convolutional codes. In some cases it is necessary to consider time-varying convolutional codes. An introduction to this topic is given in Section 2.5.

This chapter only gives a brief introduction to the theory of convolutional codes. For a more thorough survey refer to, e.g., [35, 38, 44]. The notations in this thesis follow [35].

2.1. Binary Convolutional Codes

This section is divided into two parts. First convolutional codes with polynomial generator matrices are introduced in an intuitive way. Then, the formal definitions follow. The aim is to first get some understanding of the concepts, so it will be easier to understand the definitions.

In general, a rate $R = b/c$, $b \leqslant c$, convolutional encoder input (*information sequence*) is a sequence of binary b-tuples,

$$\boldsymbol{u} = \ldots, \boldsymbol{u}_{-1}, \boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \tag{2.1}$$

where $\boldsymbol{u}_i = (u_i^{(1)} \ldots u_i^{(b)})$. The output (*code sequence*) is a sequence of binary c-tuples,

$$\boldsymbol{v} = \ldots, \boldsymbol{v}_{-1}, \boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \tag{2.2}$$

where $\boldsymbol{v}_i = (v_i^{(1)} \ldots v_i^{(c)})$. The sequences must start at a finite (positive or negative) time and may or may not end. The relation between the information sequences and the code sequences is determined by the equation

$$\boldsymbol{v} = \boldsymbol{u}\boldsymbol{G}, \tag{2.3}$$

where

$$\boldsymbol{G} = \begin{pmatrix} G_0 & G_1 & \ldots & G_m & & & \\ & G_0 & G_1 & \ldots & G_m & & \\ & & G_0 & G_1 & \ldots & G_m & \\ & & & \ddots & \ddots & & \ddots \end{pmatrix} \tag{2.4}$$

is the semi-infinite *generator matrix*, and where the sub-matrices G_i, $0 \leqslant i \leqslant m$, are binary $b \times c$ matrices. The arithmetic in (2.3) is carried out over the binary field, $\mathbb{F}_2$, and the parts left blank in the generator matrix $\boldsymbol{G}$ are assumed to be filled in with zeros.

The right hand side of (2.3) defines a discrete-time convolution between $\boldsymbol{u}$ and $\boldsymbol{g} = (G_0\ G_1 \ldots G_m)$, hence, the name *convolutional codes*. As in many other situations where convolutions appear it is convenient to express the sequences in some sort of transform. In information theory and coding theory it is common to use the delay operator D, the $\mathcal{D}$-transform. The information and code sequences becomes

$$\boldsymbol{u}(D) = \cdots + \boldsymbol{u}_{-1}D^{-1} + \boldsymbol{u}_0 + \boldsymbol{u}_1 D + \boldsymbol{u}_2 D^2 + \cdots \tag{2.5}$$

and

$$\boldsymbol{v}(D) = \cdots + \boldsymbol{v}_{-1}D^{-1} + \boldsymbol{v}_0 + \boldsymbol{v}_1 D + \boldsymbol{v}_2 D^2 + \cdots. \tag{2.6}$$

They are related through the equation

$$\boldsymbol{v}(D) = \boldsymbol{u}(D)G(D), \tag{2.7}$$

where

$$G(D) = G_0 + G_1 D + \cdots + G_m D^m \tag{2.8}$$

is the generator matrix.

The set of polynomial generator matrices is a special case of the rational generator matrices. Hence, instead of having finite impulse response in the encoder, as for the polynomial case, we can allow periodically repeating infinite impulse responses. To make the formal definitions for this case it is easier to start in the $\mathcal{D}$-domain. Let $\mathbb{F}_2((D))$ denote the *field of binary Laurent series*. The element

$$x(D) = \sum_{i=r}^{\infty} x_i D^i \in \mathbb{F}_2((D)),\ r \in \mathbb{Z}, \tag{2.9}$$

contains at most finitely many negative powers of D. Similarly, let $\mathbb{F}_2[D]$ denote the *ring of binary polynomials*. A polynomial

$$x(D) = \sum_{i=0}^{k} x_i D^i \in \mathbb{F}_2[D],\ k \in \mathbb{Z}^+, \tag{2.10}$$

contains no negative powers of D and only finitely many positive. Given a pair of polynomials $x(D), y(D) \in \mathbb{F}_2[D]$, where $y(D) \neq 0$, we can obtain the element $x(D)/y(D) \in \mathbb{F}_2((D))$ by long division. All non-zero ratios $x(D)/y(D)$ are invertible, so they form the *field of binary rational functions*, $\mathbb{F}_2(D)$, which is a sub-field of $\mathbb{F}_2((D))$.

We are now ready for the following definitions [34].

Definition 2.1: A rate $R = b/c$ (binary) *convolutional transducer* over the field of rational functions $\mathbb{F}_2(D)$ is a linear mapping

$$\begin{aligned} \tau : \mathbb{F}_2^b((D)) &\to \mathbb{F}_2^c((D)) \\ \boldsymbol{u}(D) &\to \boldsymbol{v}(D), \end{aligned} \tag{2.11}$$

which can be represented as

$$\boldsymbol{v}(D) = \boldsymbol{u}(D)G(D), \tag{2.12}$$

where $G(D)$ is a $b \times c$ *transfer function matrix* of rank b with entries in $\mathbb{F}_2(D)$ and $\boldsymbol{v}(D)$ is called the *code sequence* corresponding to the *information sequence* $\boldsymbol{u}(D)$. □

Definition 2.2: A rate $R = b/c$ *convolutional code* $\mathcal{C}$ over $\mathbb{F}_2$ is the image set of a rate $R = b/c$ convolutional transducer. □

We will only consider realizable (causal) transfer function matrices, which we call generator matrices.

Definition 2.3: A transfer function matrix of a convolutional code is called a *generator matrix* if it is realizable (causal). □

It follows from the definitions that a rate $R = b/c$ convolutional code $\mathcal{C}$ with the $b \times c$ generator matrix $G(D)$ is the row space of $G(D)$ over $\mathbb{F}((D))$. Hence, it is the set of all code sequences generated by the convolutional generator matrix, $G(D)$.

We have now defined the convolutional code and its generator matrix and continue with the encoder.

Definition 2.4: A rate $R = b/c$ *convolutional encoder* of a convolutional code with rate $R = b/c$ generator matrix $G(D)$ over $\mathbb{F}_2(D)$ is a realization by linear sequential circuits of $G(D)$. □

Example 2.1: Consider the (polynomial) generator matrix

$$G(D) = \begin{pmatrix} 1 & D & 1+D \\ D^2 & 1 & 1+D+D^2 \end{pmatrix}. \tag{2.13}$$

An encoder for this generator matrix can be built as in Figure 2.1. □

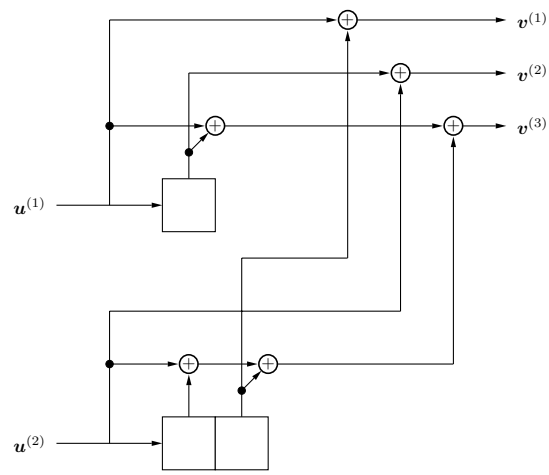

Figure 2.1: An encoder for the generator matrix in (2.13).

2.2. Structural Properties of Convolutional Codes

To be able to say more about the convolutional code and its encoders we need to take a closer look at some properties of the generator matrices.

We say that two generator matrices are *equivalent* [41] if they generate the same code $\mathcal{C}$. Thus, we have the following definition of equivalent generator matrices.

Definition 2.5: Two convolutional generator matrices $G(D)$ and $G'(D)$ are *equivalent* if they encode the same code. Two convolutional encoders are equivalent if their generator matrices are equivalent. □

In several parts of this thesis we shall exploit the well-known result that two generator matrices, $G(D)$ and $G'(D)$, are equivalent if and only if there exists a rational invertible matrix $T(D)$ such that

$$G'(D) = T(D)G(D). \tag{2.14}$$

An explanation[1] of this property is as follows. Let the two generator matrices $G(D)$ and $G'(D)$ be related via (2.14). The code sequence $\boldsymbol{v}(D) = \boldsymbol{u}(D)G(D)$

[1] It is not a proof of (2.14). It is more an interpretation of the concept of equivalence.

can be generated by $G'(D)$ as

$$\begin{aligned} \boldsymbol{v}(D) &= \boldsymbol{u}(D)G(D) = \boldsymbol{u}(D)T^{-1}(D)T(D)G(D) \\ &= \boldsymbol{u}'(D)G'(D), \end{aligned} \tag{2.15}$$

where $\boldsymbol{u}'(D) = \boldsymbol{u}(D)T^{-1}(D)$. All code sequences that can be generated by one generator matrix can also be generated by the other, but not necessarily by the same information sequence.

Definition 2.6: A convolutional generator matrix is *systematic* if the information sequence appear unchanged in the corresponding code sequence. □

As shown by the next example, every generator matrix has an equivalent systematic generator matrix.

Example 2.2: Let $G(D)$ be the same generator matrix as in Example 2.1 and choose $T(D)$ as the inverse of the first non-singular sub-matrix of $G(D)$,

$$T(D) = \begin{pmatrix} 1 & D \\ D^2 & 1 \end{pmatrix}^{-1} = \frac{1}{1+D^3}\begin{pmatrix} 1 & D \\ D^2 & 1 \end{pmatrix}. \tag{2.16}$$

An equivalent systematic generator matrix for $G(D)$ is

$$G_{\text{sys}}(D) = T(D)G(D) = \begin{pmatrix} 1 & 0 & \frac{1+D^2+D^3}{1+D^3} \\ 0 & 1 & \frac{1+D+D^3}{1+D^3} \end{pmatrix}. \tag{2.17}$$

□

In Figure 2.2 and Figure 2.3 two different encoders for the systematic generator matrix in (2.17) are shown. The encoder in Figure 2.2 is realized on *controller canonical form* and the encoder on Figure 2.3 is realized on *observer canonical form*. The controller canonical form has one shift register for each input (row of $G(D)$) while the observer canonical form has one shift register for each output (column of $G(D)$).

A *catastrophic* [41] generator matrix is a generator matrix that encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols. This gives that a finite number of channel errors may result in infinitely many errors in the receiver. Naturally, all codes have both catastrophic and non-catastrophic generator matrices.

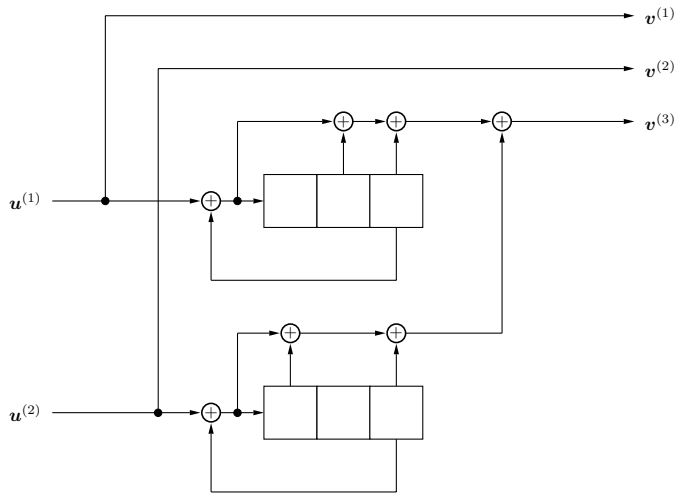

Figure 2.2: A realization on controller canonical form of the systematic generator matrix in (2.17).

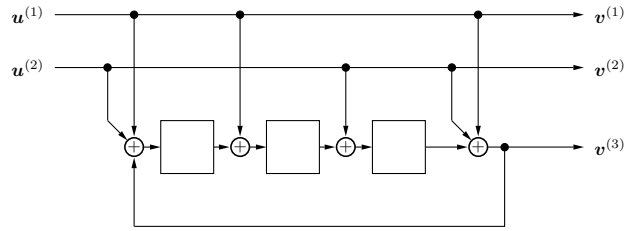

Figure 2.3: A realization on observer canonical form of the systematic generator matrix in (2.17).

The *physical state* of an encoder is the contents of the delay elements in the realization. The set of *abstract states* of a generator matrix is the set of possible continuations of code sequences when the encoder starts in an arbitrary physical state and is fed with the all-zero sequence. The set of physical states is an encoder property, it depends on the realization of the encoder, while the set of abstract states is a property of the generator matrix and does not depend on the realization. The number of abstract states is a measurement of the complexity of the generator matrix. Therefore, it is desirable to find a generator matrix with as few abstract states as possible.

Definition 2.7: A generator matrix, $G(D)$, is *minimal* [13] if it has a minimum number of abstract states among all equivalent generator matrices. □

An encoder realized from a minimal generator matrix with a minimum number of delay elements is called a *minimal encoder*. Such minimal encoders have a minimum number of delay elements over all encoders that generate the code. It is easy to show that all systematic generator matrices are minimal [35]. Thus, $G_{\text{sys}}(D)$ in Example 2.2 is a minimal generator matrix and the encoder in Figure 2.3 is a minimal encoder, while the encoder in Figure 2.2 is *not* minimal.

Let

$$G(D) = \begin{pmatrix} g_{11}(D) & \cdots & g_{1c}(D) \\ \vdots & & \vdots \\ g_{b1}(D) & \cdots & g_{bc}(D) \end{pmatrix} \tag{2.18}$$

be a generator matrix. The ith row of $G(D)$ can be written as

$$\begin{aligned} \boldsymbol{g}_i(D) &= \begin{pmatrix} g_{i1}(D) & \cdots & g_{ic}(D) \end{pmatrix} \\ &= \begin{pmatrix} \frac{f_{i1}(D)}{q(D)} & \cdots & \frac{f_{ic}(D)}{q(D)} \end{pmatrix} \\ &= \frac{1}{q(D)} \begin{pmatrix} f_{i1}(D) & \cdots & f_{ic}(D) \end{pmatrix}, \end{aligned} \tag{2.19}$$

where $f_{i1}(D), \ldots, f_{ic}(D)$, and $q(D)$ are polynomials and

$$\gcd\left(f_{i1}(D), \ldots, f_{ic}(D), q(D)\right) = 1. \tag{2.20}$$

The *constraint length* of the ith row is defined as

$$\nu_i \triangleq \max\left\{\deg f_{i1}(D), \ldots, \deg f_{ic}(D), \deg q(D)\right\}. \tag{2.21}$$

Define the *overall constraint length* as the sum of all constraint lengths,

$$\nu \triangleq \sum_{i=1}^{b} \nu_i, \tag{2.22}$$

and the *memory* and *minimum constraint length* as

$$m \triangleq \max_{1 \leqslant i \leqslant b} \{\nu_i\} \qquad \text{and} \qquad \nu_{\min} \triangleq \min_{1 \leqslant i \leqslant b} \{\nu_i\}, \tag{2.23}$$

respectively.

A realization on controller canonical form of a generator matrix uses ν delay elements. Hence, one way to minimize the complexity of the encoder is to minimize the overall constraint length of the generator matrix.

Definition 2.8: A generator matrix is *canonical* [16] if it has minimum overall constraint length over all equivalent generator matrices. □

It is possible to show that a canonical generator matrix is minimal and that a realization on controller canonical form of a canonical generator matrix is a minimal encoder.

The constraint lengths ν_i, $1 \leqslant i \leqslant b$, are invariant, up to rearrangements, over the set of canonical generator matrices. Therefore, when we talk about the constraint lengths, the overall constraint length, or the memory of a convolutional code, they are understood to be derived for a canonical generator matrix for the code.

For the class of polynomial generator matrices the constraint length of the ith row is, according to (2.21), defined as

$$\nu_i \triangleq \max_{1 \leqslant j \leqslant c} \{\deg g_{ij}(D)\}. \tag{2.24}$$

The definitions for the overall constraint length, memory, and minimal constraint length are identical to (2.22) and (2.23).

Other important properties of the class of polynomial generator matrices are *basic* [13] generator matrices and *minimal-basic* [35] generator matrices.

Definition 2.9: A generator matrix is *basic* if it is polynomial and has a polynomial right inverse. □

The generator matrix $G(D)$ from Example 2.1 and Example 2.2 is basic since it is polynomial and has a polynomial right inverse, viz.,

$$G^{-1}(D) = \begin{pmatrix} 1 + D^3 + D^4 & D + D^2 \\ D^2 + D^4 & 1 + D^2 \\ D^4 & D^2 \end{pmatrix}. \tag{2.25}$$

Definition 2.10: A basic generator matrix is *minimal-basic* if it has minimal overall constraint length over all equivalent basic generator matrices. □

A minimal-basic generator matrix is a polynomial and canonical generator matrix, and vice versa. Therefore, a minimal-basic generator matrix is minimal and a realization on controller canonical form is a minimal encoder.

Remark: Although every minimal-basic generator matrix is basic and minimal, minimal-basic is not the same as basic and minimal. It is easy to construct an example [35] where the generator matrix is both basic and minimal but not minimal-basic.

2.3. Distance Properties of Convolutional Codes

The Hamming weight of a sequence, $w_{\mathrm{H}}(\boldsymbol{x})$, is the number of positions in which a sequence $\boldsymbol{x}$ differs from zero. Similarly, the Hamming distance between two sequences, $d_{\mathrm{H}}(\boldsymbol{x}_1, \boldsymbol{x}_2)$, is the number of positions in which they differ from each other,

$$d_{\mathrm{H}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = w_{\mathrm{H}}(\boldsymbol{x}_1 - \boldsymbol{x}_2). \tag{2.26}$$

The most important distance property of convolutional codes is the *free distance* [8].

Definition 2.11: The *free distance*, d_{free}, of a convolutional code, $\mathcal{C}$, is the minimum Hamming distance between two code sequences,

$$d_{\mathrm{free}} \triangleq \min_{\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{C}} \{d_{\mathrm{H}}(\boldsymbol{v}_1, \boldsymbol{v}_2)\}. \tag{2.27}$$

□

Since convolutional codes are linear, all non-zero code sequences can be compared with the all-zero sequence to get the same result,

$$d_{\mathrm{free}} = \min_{\boldsymbol{v} \in \mathcal{C} \setminus \boldsymbol{0}} \{w_{\mathrm{H}}(\boldsymbol{v})\}. \tag{2.28}$$

The free distance determines the error correction capability of the convolutional code. A minimum distance decoder can *always* correct an error sequence, $\boldsymbol{e}$, if

$$w_{\mathrm{H}}(\boldsymbol{e}) < \frac{d_{\mathrm{free}}}{2}. \tag{2.29}$$

Other well known distance measures are the *column distance* [8] and the *row distance* [9].

Definition 2.12: The *jth order column distance* is

$$d_j^c = \min_{\boldsymbol{u},\ \boldsymbol{u}_0 \neq \boldsymbol{0}} \{w_{\mathrm{H}}\left(\boldsymbol{v}_{[0,j]}\right)\}, \tag{2.30}$$

where $\boldsymbol{u}$ is a causal information sequence with the first symbol non-zero, and $\boldsymbol{v}_{[0,j]}$ the corresponding code sequence truncated after time j. □

The column distance is the minimum Hamming distance of the first $j+1$ symbols between two code sequences when the information sequences differ in the first position. If, instead of the code sequence, the information sequence is truncated we get the row distance.

Definition 2.13: The *jth order row distance* is

$$d_j^r = \min_{\boldsymbol{u}_{[0,j]},\ \boldsymbol{u}_0 \neq \boldsymbol{0}} \{w_{\mathrm{H}}\left(\boldsymbol{v}\right)\}, \tag{2.31}$$

where $\boldsymbol{u}_{[0,j]}$ is a causal information sequence with the first symbol nonzero and otherwise freely chosen symbols up to time j, after which the encoder is driven back to the zero state, and $\boldsymbol{v}$ the corresponding code sequence. □

The column distance is a non-decreasing function that converges to the free distance. Similarly, the row distance is a non-increasing function and, if the generator matrix is non-catastrophic, it converges to the free distance. Hence, for non-catastrophic generator matrices

$$d_0^c \leqslant d_1^c \leqslant \ldots \leqslant d_\infty^c = d_{\mathrm{free}} = d_\infty^r \leqslant \ldots \leqslant d_1^r \leqslant d_0^r. \tag{2.32}$$

For catastrophic generator matrices the asymptotic value of the row distance is lower bounded by the free distance, $d_\infty^r \geqslant d_{\mathrm{free}}$.

2.4. The trellis

It is often helpful to view the information sequences and code sequences in a tree structure. In Figure 2.4 such tree is shown for the generator matrix

$$G(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix}. \tag{2.33}$$

In the figure time passes from left to right The nodes are labeled with the corresponding (physical) states of the encoder. The encoder is realized in controller canonical form, see Figure 2.5, and starts in the zero state.

The state represents all the encoder knows about the past. Therefore, the continuation of two paths with the same state at time t will be identical.

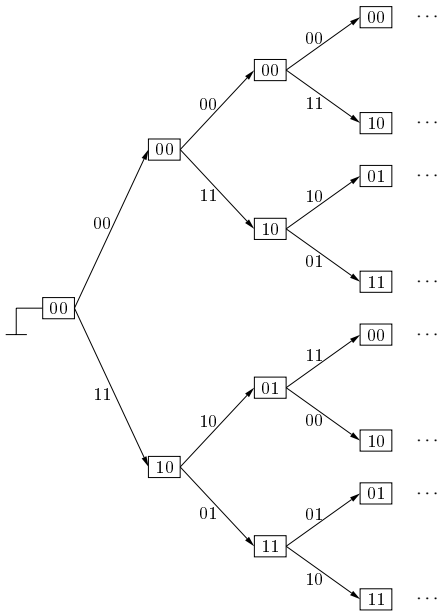

Figure 2.4: A tree structure representing the generator matrix in (2.33). If the input is 0 choose the upper branch and if it is 1 choose the lower. The labels of the branches are the corresponding code symbols and the nodes are labeled with the encoder states.

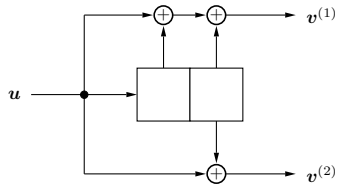

Figure 2.5: Encoder in controller canonical form of the generator matrix in (2.33).

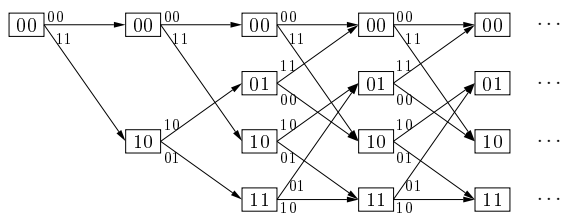

Figure 2.6: The trellis for the generator matrix (2.33).

Instead of having such identical paths, for each time they can be merged. The tree then collapses into a *trellis* [14]. The trellis for the generator matrix (2.33) is shown in Figure 2.6.

Each path in the trellis describes an information sequence and the corresponding code sequence. Also, each path can be represented by a state sequence,

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_0 \boldsymbol{\sigma}_1 \boldsymbol{\sigma}_2 \dots , \tag{2.34}$$

where $\boldsymbol{\sigma}_i$ are binary matrices.

2.5. Time-varying Convolutional Codes

So far we have considered only *time-invariant* convolutional codes, i.e., convolutional codes encoded by time-invariant generator matrices. We can often obtain powerful results if we study time-varying convolutional codes instead.

Assuming polynomial generator matrices, then from (2.3)

$$\boldsymbol{v}_t = \boldsymbol{u}_t G_0 + \boldsymbol{u}_{t-1} G_1 + \dots + \boldsymbol{u}_{t-m} G_m, \tag{2.35}$$

where G_i, $0 \leqslant i \leqslant m$, are binary $b \times c$ time-invariant matrices.

In general, a rate $R = b/c$, binary convolutional code can be *time-varying*. Then (2.35) becomes

$$\boldsymbol{v}_t = \boldsymbol{u}_t G_0(t) + \boldsymbol{u}_{t-1} G_1(t) + \dots + \boldsymbol{u}_{t-m} G_m(t), \tag{2.36}$$

where $G_i(t)$, $0 \leqslant i \leqslant m$, are binary $b \times c$ time-varying matrices. As a counterpart to the semi-infinite matrix $\boldsymbol{G}$ given in (2.4) we have

$$\boldsymbol{G}_t = \begin{pmatrix} G_0(t) & \dots & G_m(t+m) & & \\ & G_0(t+1) & \dots & & G_m(t+1+m) \\ & & \ddots & & & \ddots \end{pmatrix}. \tag{2.37}$$

Let $\mathcal{E}(b,c,m)$ be the ensemble of binary, rate $R = b/c$, time-varying convolutional codes with generator matrices of memory m in which each digit in each of the matrices $G_i(t)$ for $0 \leqslant i \leqslant m$ and $t = 0, 1, 2, \ldots$ is chosen independently and is equally likely to be 0 and 1.

Remark: With a slight abuse of terminology we call $\boldsymbol{G}_t$ a generator matrix although it might not have full rank.

As a special case of the ensemble of time-varying convolutional codes we have the ensemble of binary, rate $R = b/c$, *periodically* time-varying convolutional codes encoded by a polynomial generator matrix $\boldsymbol{G}_t$ of memory m and *period* T, in which each digit in each of the matrices $G_i(t) = G_i(t+kT)$ for $0 \leqslant i \leqslant m$, $t = 0, 1, \ldots, T-1$, and $k = 1, 2, \ldots$, is chosen independently and is equally likely to be 0 and 1. We denote this ensemble $\mathcal{E}(b,c,m,T)$.

With this new tool we can obtain powerful bounds for convolutional codes. Here we give, without proof, Costello's lower bound on the free distance [9].

Theorem 2.1 [Costello]: There exists a binary, periodically time-varying, rate $R = b/c$ convolutional code encoded by a polynomial generator matrix of memory m that has a free distance satisfying the inequality[2]

$$\delta_C \triangleq \frac{d_{\text{free}}}{mc} \geqslant \frac{R}{-\log(2^{1-R}-1)} + O\left(\frac{\log m}{m}\right). \tag{2.38}$$

□

If the period is chosen to $T = 1$ we get the ensemble of binary, rate $R = b/c$, time-invariant convolutional codes encoded by a polynomial generator matrix G of memory m.

[2] Here and hereafter $f(x) = O(g(x))$ means $|f(x)| \leqslant Ag(x)$ for x sufficiently large, where A is a positive constant and $g(x) > 0$, i.e., that the asymptotic behavior of $f(x)$ is upper bounded by $Ag(x)$.

3

Active Distances for Convolutional Codes

The column distance [9] has the property that it will not increase any more when it has reached the free distance. In this chapter we introduce a family of distances that stay "active" in the sense that we consider only those codewords which do not pass two consecutive zero encoder states. These distances, called the *active distances* [29], determine the error correcting capability of the code and they are of particular importance when we consider concatenated convolutional encoders.

The active distances can be regarded as (non-trivial) generalizations to encoder memories $m > 1$ of the *extended distances* introduced for unit-memory convolutional codes by Thommesen and Justesen [46].

In Section 3.1 the active distances will be defined for time-invariant convolutional codes. Some important properties for convolutional codes are obtained via the active distances in Section 3.2, and in Section 3.3 lower bounds on the active distances for fixed time-invariant convolutional codes are derived. In Section 3.4 restricted sets of information sequences are used to define the active distances for time-varying convolutional codes. In the final section of this chapter, Section 3.5, these definitions are used to obtain lower bounds on the active distances for the ensemble of periodically time-varying convolutional codes.

3.1. Active Distances for Convolutional Codes

For each information sequence,

$$\boldsymbol{u} = \ldots \boldsymbol{u}_{-1}\boldsymbol{u}_0\boldsymbol{u}_1\boldsymbol{u}_2 \ldots , \tag{3.1}$$

there is a corresponding state sequence,

$$\boldsymbol{\sigma} = \ldots \boldsymbol{\sigma}_{-1}\boldsymbol{\sigma}_0\boldsymbol{\sigma}_1\boldsymbol{\sigma}_2 \ldots , \tag{3.2}$$

where $\boldsymbol{\sigma}_i$ is a binary $b \times m$ matrix representing the contents of an encoder in controller canonical form of the generator matrix.

Let $\mathcal{S}_{[t_1,t_2]}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}$ denote the set of state sequences $\boldsymbol{\sigma}_{[t_1,t_2]}$ that start at depth t_1 in state $\boldsymbol{\sigma}_1$ and terminate at depth t_2 in state $\boldsymbol{\sigma}_2$ and do not have two consecutive zero states in between, i.e.,

$$\mathcal{S}_{[t_1,t_2]}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} \triangleq \Big\{ \boldsymbol{\sigma}_{[t_1,t_2]} \Big| \boldsymbol{\sigma}_{t_1} = \boldsymbol{\sigma}_1, \boldsymbol{\sigma}_{t_2} = \boldsymbol{\sigma}_2 \text{ and } (\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_{i+1}) \neq (\mathbf{0},\mathbf{0}), t_1 \leqslant i < t_2 \Big\}. \tag{3.3}$$

The notation $\boldsymbol{x}_{[t_1,t_2]}$ means the part of the (infinite) sequence in the interval $t_1 \leqslant t \leqslant t_2$,

$$\boldsymbol{x}_{[t_1,t_2]} = (\boldsymbol{x}_{t_1}\boldsymbol{x}_{t_1+1} \ldots \boldsymbol{x}_{t_2}). \tag{3.4}$$

Definition 3.1: Let $\mathcal{C}$ be a convolutional code encoded by a rational generator matrix $G(D)$ of memory m which is realized in controller canonical form. The *jth order active row distance* is

$$a_j^r \triangleq \min_{\mathcal{S}_{[0,j+1]}^{\mathbf{0},\boldsymbol{\sigma}}, \boldsymbol{\sigma}_{j+1+i}^{(1,i)} = \mathbf{0}, 1 \leqslant i \leqslant m} \{ w_{\mathrm{H}}(\boldsymbol{v}_{[0,j+m]}) \}, \tag{3.5}$$

where $\boldsymbol{\sigma}$ denotes any value of the state $\boldsymbol{\sigma}_{j+1}$ such that $\boldsymbol{\sigma}_{j+1}^{(1)} \neq \mathbf{0}$, and $\boldsymbol{\sigma}_{j+1+i}^{(1,i)}$ denotes the i first positions of the shift registers (counted from the input connections) at depth $j+i+1$, i.e., $\boldsymbol{\sigma}_{j+1+i}^{(1,i)} = \boldsymbol{\sigma}_{j+1+i}^{(1)} \ldots \boldsymbol{\sigma}_{j+1+i}^{(i)}$. □

The active row distance of order j is the minimum weight of paths that diverge from the zero state at depth 0, possibly "touches" the all-zero path only in non-consecutive zero states at depth k, where $1 + \nu_{\min} \leqslant k \leqslant j$, and the input at time j is such that the first column of the following state is non-zero, and, finally, the encoder is driven directly back to the zero state. The paths will remerge with the zero state at depth l, where $j + 1 + \nu_{\min} \leqslant l \leqslant j + 1 + m$.

For a polynomial generator matrix realized in controller canonical form we have the following equivalent formulation,

$$a_j^r = \min_{\boldsymbol{u}_j \neq 0,\, \mathcal{S}_{[0,j+1]}^{\mathbf{0},\boldsymbol{\sigma}}} \left\{ w_{\mathrm{H}} \left(\boldsymbol{u}_{[0,j]} \boldsymbol{G}_j^r \right) \right\}, \tag{3.6}$$

where $\boldsymbol{\sigma}$ denotes any value of the state $\boldsymbol{\sigma}_{j+1}$ with $\boldsymbol{\sigma}_{j+1}^{(1)} = \boldsymbol{u}_j$ and where

$$\boldsymbol{G}_j^r = \begin{pmatrix} G_0 & G_1 & \dots & G_m & & & \\ & G_0 & G_1 & \dots & G_m & & \\ & & \ddots & \ddots & & \ddots & \\ & & & G_0 & G_1 & \dots & G_m \end{pmatrix} \tag{3.7}$$

is a $(j+1) \times (j+1+m)$ truncated version of the semi-infinite matrix $\boldsymbol{G}$ in (2.4). The active row distance of order j for a polynomial generator matrix is the minimum weight of a code sequence corresponding to a burst of length $j+1$ in the information sequence.

Notice that the active row distance sometimes can decrease but, as we shall show in Section 3.5, in the ensemble of convolutional codes encoded by periodically time-varying generator matrices there exists a convolutional code encoded by a generator matrix such that its active row distance can be lower bounded by a linearly increasing function.

From the definition follows the triangle inequality. Let $G(D)$ be a rational generator matrix with $\nu_{\min} = m$. Then its active row distance satisfies the triangle inequality

$$a_j^r \leqslant a_i^r + a_{j-i-1-m}^r, \tag{3.8}$$

where $j > i + m$ and the sum of the lengths of the paths to the right of the inequality is

$$i + m + 1 + (j - i - m - 1) + m + 1 = j + m + 1, \tag{3.9}$$

i.e., equal to the length of the path to the left of the inequality. Furthermore, we have immediately the following important theorem.

Theorem 3.1: Let $\mathcal{C}$ be a convolutional code encoded by a non-catastrophic generator matrix. Then

$$\min_j \{a_j^r\} = d_{\mathrm{free}}. \tag{3.10}$$

□

The following simple example shows that the triangle inequality (3.8) would not hold if we do not include state sequences that contain isolated inner zero states in the definition of $\mathcal{S}_{[t_1,t_2]}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}$.

Example 3.1: Consider the memory $m = 1$ generator matrix

$$G(D) = \begin{pmatrix} 1 & D \end{pmatrix}. \tag{3.11}$$

The code sequences corresponding to the state sequences $(0,1,0,1,0)$ and $(0,1,1,1,0)$ are $(10,01,10,01)$ and $(10,11,11,01)$, respectively. It is easily verified that $a_0^r = 2$, $a_1^r = 4$, and $a_2^r = 4$, which satisfy the triangle inequality

$$a_2^r \leqslant a_0^r + a_0^r. \tag{3.12}$$

If we consider only state sequences without isolated inner zero states the lowest weight sequence of length four would pick up distance 6 and exceed the sum of the weight for two length two sequence, which would still be four, in violation with the triangle inequality. □

The jth order active row distance is characterized by a fixed number of almost freely chosen information tuples, $j+1$, followed by a varying number, between $\nu_{\min}$ and m, of zero state driving information tuples ("almost" since we have to avoid consecutive zero states and assure that $\boldsymbol{\sigma}_{j+1}^{(1)} \neq \mathbf{0}$). Sometimes it is useful to consider a corresponding distance where the total length, $j+1$, is fixed, but with a varying number of almost freely chosen information tuples. Hence, we introduce the active burst distance.

Definition 3.2: Let $\mathcal{C}$ be a convolutional code encoded by a rational generator matrix $G(D)$ of memory m. The *jth order active burst distance* is

$$a_j^b \triangleq \min_{\mathcal{S}_{[0,j+1]}^{\mathbf{0},\mathbf{0}}} \{w_{\mathrm{H}}(\boldsymbol{v}_{[0,j]})\}, \tag{3.13}$$

where $j \geqslant \nu_{\min}$. □

For a polynomial generator matrix we have the following equivalent formulation,

$$a_j^b \triangleq \min_{\mathcal{S}_{[0,j+1]}^{\mathbf{0},\mathbf{0}}} \{w_{\mathrm{H}}(\boldsymbol{u}_{[0,j]}\boldsymbol{G}_j^b)\}, \tag{3.14}$$

where $j \geqslant \nu_{\min}$ and

$$\boldsymbol{G}_j^b = \begin{pmatrix} G_0 & G_1 & \dots & G_m & & & \\ & G_0 & G_1 & \dots & G_m & & \\ & & \ddots & \ddots & & \ddots & \\ & & & G_0 & G_1 & \dots & G_m \\ & & & & G_0 & & G_{m-1} \\ & & & & & \ddots & \vdots \\ & & & & & & G_0 \end{pmatrix} \tag{3.15}$$

is a $(j+1) \times (j+1)$ truncated version of the semi-infinite matrix $\boldsymbol{G}$ given in (2.4). Notice that the active burst distance is undefined for $j < \nu_{\min}$.

The active row and burst distances are related via

$$a_j^b \geqslant \min_i \{a_{j-\nu_i}^r\} \tag{3.16}$$

and

$$a_j^r \geqslant \min_i \{a_{j+\nu_i}^b\}, \tag{3.17}$$

where ν_i is the ith rows constraint length of the generator matrix $G(D)$. Clearly, when $\nu_{\min} = m$, we have

$$a_j^r = a_{j+m}^b. \tag{3.18}$$

For a non-catastrophic generator matrix

$$\min_j \{a_j^b\} = d_{\text{free}}. \tag{3.19}$$

From the definition it is obvious that the active burst distance satisfies the triangle inequality,

$$a_j^b \leqslant a_i^b + a_{j-i-1}^b. \tag{3.20}$$

From this follows easily the triangle inequality for the active row distance (3.8), since for $\nu_{\min} = m$

$$a_j^r = a_{j+m}^b \leqslant a_{i+m}^b + a_{j-i-1}^b = a_i^r + a_{j-i-1-m}^r. \tag{3.21}$$

Next we consider the active counterpart to the column distance.

Definition 3.3: Let $\mathcal{C}$ be a convolutional code encoded by a rational generator matrix $G(D)$ of memory m realized in controller canonical form. The *jth order active column distance* is

$$a_j^c \triangleq \min_{\mathcal{S}_{[0,j+1]}^{\mathbf{0},\boldsymbol{\sigma}}} \{w_{\text{H}}(\boldsymbol{v}_{[0,j]})\}, \tag{3.22}$$

where $\boldsymbol{\sigma}$ denotes any encoder state. □

For a polynomial generator matrix we have the following equivalent formulation:

$$a_j^c = \min_{\mathcal{S}_{[0,j+1]}^{\mathbf{0},\boldsymbol{\sigma}}} \left\{ w_{\mathrm{H}} \left(\boldsymbol{u}_{[0,j]} \boldsymbol{G}_j^c \right) \right\}, \tag{3.23}$$

where $\boldsymbol{\sigma}$ denotes any encoder state and $\boldsymbol{G}_j^c = \boldsymbol{G}_j^b$.

It follows from the definitions that

$$a_j^c \leqslant a_j^b, \tag{3.24}$$

and, if $\nu_{\min} = m$ for $j \geqslant m$

$$a_j^c \leqslant a_{j-m}^r. \tag{3.25}$$

The reverse of the active column distance is when we consider paths that start in an arbitrary state and remerges with the all-zero path after $j+1$ steps.

Definition 3.4: Let $\mathcal{C}$ be a convolutional code encoded by a rational generator matrix $G(D)$ of memory m. The *jth order active reverse column distance* is

$$a_j^{rc} \triangleq \min_{\mathcal{S}_{[m,m+j+1]}^{\boldsymbol{\sigma},\mathbf{0}}} \{ w_{\mathrm{H}}(\boldsymbol{v}_{[m,j+m]}) \}, \tag{3.26}$$

where $\boldsymbol{\sigma}$ denotes any encoder state. □

For a polynomial generator matrix we have the following equivalent formulation to (3.26),

$$a_j^{rc} = \min_{\mathcal{S}_{[m,m+j+1]}^{\boldsymbol{\sigma},\mathbf{0}}} \left\{ w_{\mathrm{H}} \left(\boldsymbol{u}_{[0,j+m]} \boldsymbol{G}_j^{rc} \right) \right\}, \tag{3.27}$$

where $\boldsymbol{\sigma}$ denotes any encoder state and

$$\boldsymbol{G}_j^{rc} = \begin{pmatrix} G_m & & & \\ G_{m-1} & G_m & & \\ \vdots & G_{m-1} & \ddots & \\ G_0 & \vdots & & G_m \\ & G_0 & & G_{m-1} \\ & & \ddots & \vdots \\ & & & G_0 \end{pmatrix} \tag{3.28}$$

is a $(j+m+1) \times (j+1)$ truncated version of the semi-infinite matrix $\boldsymbol{G}$ given in (2.4).

The active reverse column distance of a generator matrix $G(D)$ is equal to the active column distance of the *reciprocal generator matrix*. For a polynomial generator matrix, $G(D)$, the reciprocal generator matrix is defined as

$$G_{\text{rec}}(D) \triangleq \operatorname{diag}\left(D^{\nu_1} D^{\nu_2} \dots D^{\nu_b}\right) G(D^{-1}), \tag{3.29}$$

where $\operatorname{diag}(\boldsymbol{x})$ is a diagonal matrix with the vector $\boldsymbol{x}$ on its diagonal.

Our final definition for the family of active distances is the active segment distance.

Definition 3.5: Let $\mathcal{C}$ be a convolutional code encoded by a rational generator matrix $G(D)$ of memory m. The *jth order active segment distance* is

$$a_j^s \triangleq \min_{\mathcal{S}_{[m,m+j+1]}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}} \{w_{\text{H}}(\boldsymbol{v}_{[m,j+m]})\}, \tag{3.30}$$

where $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ denote any encoder states. □

For a polynomial generator matrix we have the following equivalent formulation,

$$a_j^s = \min_{\mathcal{S}_{[m,m+j+1]}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}} \left\{w_{\text{H}}\left(\boldsymbol{u}_{[0,j+m]} \boldsymbol{G}_j^s\right)\right\}, \tag{3.31}$$

where $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ denote any encoder states, and $\boldsymbol{G}_j^s = \boldsymbol{G}_j^{rc}$.

If we consider the segment distances for two sets of consecutive paths of lengths $i+1$ and $(j-i-1)+1$, respectively, then the terminating state of the first path is not necessarily identical to the starting state of the second path. Hence, the active segment distance for the set of paths of the total length $j+1$ does not necessarily satisfy the triangle inequality. Instead it satisfies the inequality,

$$a_j^s \geqslant a_i^s + a_{j-i-1}^s, \tag{3.32}$$

where $j > i$ and the sum of the lengths of the paths to the right of the inequality is

$$i+1+j-i-1+1 = j+1, \tag{3.33}$$

i.e., equal to the length of the path to the left of the inequality.

The *start* of the active segment distance is of special interest.

Definition 3.6: Let j_0^s denote the largest j for which

$$a_j^s = 0, \tag{3.34}$$

i.e., j_0^s+1 is the largest number of information tuples that can give an output with Hamming weight zero, given that we do not have two consecutive zero states among the corresponding j_0^s+2 states. □

Example 3.2: In Figure 3.1 we show the active distances for the generator matrix

$$G(D) = \Big(1+D+D^2+D^3+D^7+D^8+D^9+D^{11} \quad 1+D^2+D^3+D^7+D^8+D^9+D^{11}\Big). \tag{3.35}$$

Notice that the active row distance of the 0th order, a_0^r, is identical to the row distance of the 0th order, $d_0^r = 15$, which upper bounds $d_{\text{free}} = 12$. The start of the active segment distance is $j_0^s = 9$. □

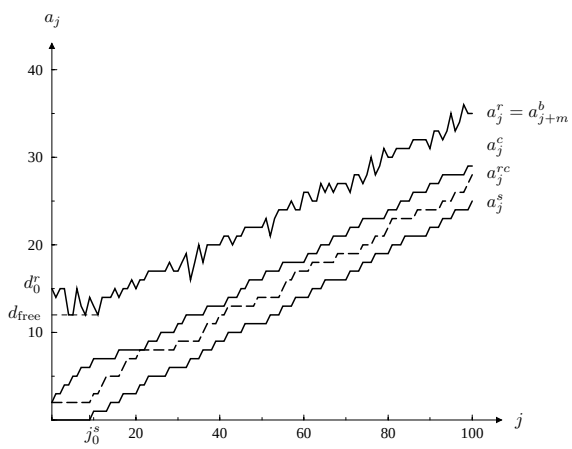

Figure 3.1: The active distances for the generator matrix in Example 3.2.

From the definitions follow that the active distances are encoder properties, not code properties. However, it also follows that the active distances are invariants over the set of canonical (or minimal-basic) generator matrices

for a convolutional code $\mathcal{C}$. Hence, when we in the sequel consider active distances for convolutional codes it is understood that these distances are evaluated for a corresponding canonical (minimal-basic) generator matrix.

3.2. Properties of Convolutional Codes

We define the *correct path* through a trellis to be the path determined by the encoded information sequence and we call the (encoder) states along the correct path *correct states*. Then we define an *incorrect segment* to be a segment starting in a correct state $\boldsymbol{\sigma}_{t_1}$ and terminating in a correct state $\boldsymbol{\sigma}_{t_2}, t_1 < t_2$, such that it differs from the correct path at some, but not necessarily all, states within this interval. Let $e_{[k,l)}$ denote the number of errors in the error pattern $\boldsymbol{e}_{[k,l)}$, where $\boldsymbol{e}_{[k,l)} = \boldsymbol{e}_k \boldsymbol{e}_{k+1} \dots \boldsymbol{e}_{l-1}$.

For a convolutional code $\mathcal{C}$ with a generator matrix of memory m consider any incorrect segment between two correct states, $\boldsymbol{\sigma}_{t_1}$ and $\boldsymbol{\sigma}_{t_2}$. A minimum distance (MD) decoder can output an incorrect segment between $\boldsymbol{\sigma}_{t_1}$ and $\boldsymbol{\sigma}_{t_2}$ only if there exists a segment of length $j+1$ c-tuples, $\nu_{\min} \leqslant j < t_2 - t_1$, between these two states such that the number of channel errors $e_{[t_1,t_2)}$ within the interval is at least $a_j^b/2$. Thus, we have the following theorem.

Theorem 3.2: A convolutional code $\mathcal{C}$ encoded by a rational generator matrix of smallest constraint length $\nu_{\min}$ can correct all error patterns $\boldsymbol{e}_{[t_1,t_2)}$ that satisfy

$$e_{[t_1+k,t_1+1+i)} < a_{i-k}^b/2 \tag{3.36}$$

for $0 \leqslant k \leqslant t_2 - t_1 - \nu_{\min} - 1$, $k + \nu_{\min} \leqslant i \leqslant t_2 - t_1 - 1$. □

We have immediately a corollary.

Corollary 3.3: A convolutional code $\mathcal{C}$ encoded by a rational generator matrix of memory m and smallest constraint length $\nu_{\min} = m$ can correct all error patterns $\boldsymbol{e}_{[t_1,t_2)}$ that satisfy

$$e_{[t_1+k,t_1+1+i)} < a_{i-k-m}^r/2 \tag{3.37}$$

for $0 \leqslant k \leqslant t_2 - t_1 - m - 1$, $k + m \leqslant i \leqslant t_2 - t_1 - 1$. □

Both the active column distance and the active reverse column distance are important parameters when we study the error correcting capability of a convolutional code. As a counterpart to Theorem 3.2 we have

Theorem 3.4: Let $\mathcal{C}$ be a convolutional code encoded by a rational generator matrix of memory m and let $\boldsymbol{e}_{[t_1,t_2)}$ be an error sequence between the two correct states $\boldsymbol{\sigma}_{t_1}$ and $\boldsymbol{\sigma}_{t_2}$. A minimum distance decoder will output a correct state $\boldsymbol{\sigma}_t$ at depth t, $t_1 < t < t_2$, if

$$\begin{cases} e_{[i,t)} < a^c_{t-i-1}/2, \ t_1 \leqslant i < t \\ e_{[t,j)} < a^{rc}_{j-t-1}/2, \ t < j \leqslant t_2. \end{cases} \tag{3.38}$$

□

Proof: Assume without loss of generality that the correct path is the all-zero path. The weight of any path of length $t-i$ diverging from the correct path at depth i, $i < t$, and not having two consecutive zero states is lower bounded by a^c_{t-i-1}. Similarly, the weight of any path of length $j-t$, $j > t$, remerging with the correct path at depth j and not having two consecutive zero states is lower bounded by a^{rc}_{j-t-1}. Hence, if $e_{[i,t)} < a^c_{t-i-1}/2$ and $e_{[t,j)} < a^{rc}_{j-t-1}/2$, then $\boldsymbol{\sigma}_t$ must be correct. ■

Since

$$a^c_{t-i-1} + a^{rc}_{j-t-1} \leqslant a^b_{j-i-1} \tag{3.39}$$

it follows that we can regard Theorem 3.2 as a corollary to Theorem 3.4.

Example 3.3: Assume that the binary, rate $R = 1/2$, memory $m = 2$ convolutional generator matrix

$$G(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix} \tag{3.40}$$

is used to communicate over a binary symmetric channel and that we have the following error pattern

$$\boldsymbol{e}_{[0,20)} = 10000100000000010000000010000000000100001 \tag{3.41}$$

or, equivalently,

$$\boldsymbol{e}_{[0,20)}(D) = (10) + (01)D^2 + (01)D^7 + (10)D^{12} + (10)D^{17} + (01)D^{19}. \tag{3.42}$$

The active distances are given in Figure 3.2. From Theorem 3.2 it is easily seen that if we assume that $\boldsymbol{\sigma}_0$ is a correct state and that there exists a $t' \geqslant 20$ such that $\boldsymbol{\sigma}_{t'}$ is a correct state then, despite the fact that the number of channel errors $e_{[0,20)} = 6 > d_{\text{free}} = 5$, the error pattern (3.41) will be corrected by a minimum distance decoder.

The error pattern

$$\boldsymbol{e}'_{[0,20)} = 1010010000000000000000000000000101001 \tag{3.43}$$

or, equivalently,

$$\boldsymbol{e}'_{[0,20)}(D) = (10) + (10)D + (01)D^2 + (10)D^{17} + (10)D^{18} + (01)D^{19} \tag{3.44}$$

contains also six channel errors but with a different distribution. We have three channel errors in both the prefix and suffix 101001. Since $\nu_{\min} = m = 2$ and $a_2^b = 5$, Theorem 3.2 does not imply that the error pattern (3.43) is corrected by a minimum distance decoder. In fact, the states $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \boldsymbol{\sigma}_{18}$, and $\boldsymbol{\sigma}_{19}$ will be erroneously decoded. From Theorem 3.4 follows that if $\boldsymbol{\sigma}_0$ is a correct state and if there exists a $t' \geqslant 20$ such that $\boldsymbol{\sigma}_{t'}$ is a correct state, then at least $\boldsymbol{\sigma}_{10}$ is a correct state. □

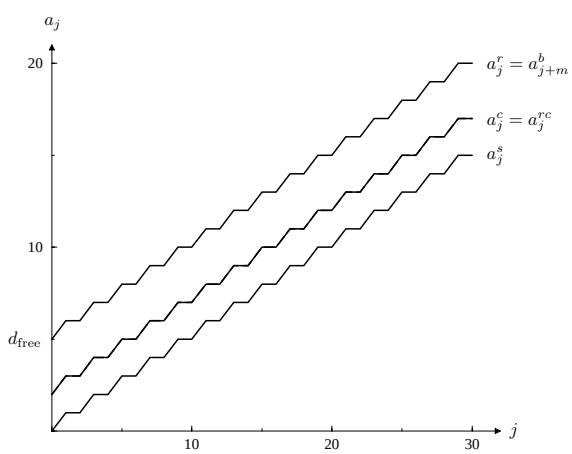

Figure 3.2: The active row, column, and segment distance for the generator matrix in (3.40).

We will now study the set of code sequences corresponding to encoder state sequences that do not contain two consecutive zero states. From the properties of the active segment distance it follows that such code sequences can contain at most $j_0^s + 1$ zero c-tuples, where j_0^s is the start of the segment distance. Lower bounds on the number of non-zero code symbols between two bursts of zeros are given in the following theorem.

Theorem 3.5: Consider a binary, rate $R = b/c$ convolutional code and let $\boldsymbol{v}^c_{[0,j']}$, $\boldsymbol{v}^{rc}_{[0,j']}$, and $\boldsymbol{v}^s_{[m,j'+m]}$ denote code sequences corresponding to state sequences in $\mathcal{S}^{\mathbf{0},\boldsymbol{\sigma}}_{[0,j'+1]}$, $\mathcal{S}^{\boldsymbol{\sigma},\mathbf{0}}_{[0,j'+1]}$, and $\mathcal{S}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}_{[m,m+j'+1]}$, respectively, where $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}_1$, and $\boldsymbol{\sigma}_2$ denote any encoder states.

(i) Let w^c_j denote the number of ones in (the weight of) a code sequence $\boldsymbol{v}^c_{[0,j']}$ counted from the beginning of the code sequence to the first burst of j consecutive zero c-tuples. Then w^c_j satisfies

$$w^c_j \geqslant a^c_{j+\lceil w^c_j/c\rceil-1}. \tag{3.45a}$$

(ii) Let w^{rc}_j denote the number of ones in (the weight of) a code sequence $\boldsymbol{v}^{rc}_{[0,j']}$ counted from the last burst of j consecutive zero c-tuples to the end of the code sequence. Then w^{rc}_j satisfies

$$w^{rc}_j \geqslant a^{rc}_{j+\lceil w^{rc}_j/c\rceil-1}. \tag{3.45b}$$

(iii) Let $w^s_{j_1,j_2}$ denote the number of ones in (the weight of) a code sequence $\boldsymbol{v}^s_{[m,j'-m]}$ counted between any two consecutive bursts of j_1 and j_2 consecutive zero c-tuples, respectively. Then $w^s_{j_1,j_2}$ satisfies

$$w^s_{j_1,j_2} \geqslant a^s_{j_1+j_2+\lceil w^s_{j_1,j_2}/c\rceil-1}. \tag{3.45c}$$

□

Proof: (i) The sub-sequence up to the beginning of the first burst of j consecutive zero c-tuples consists of at least $\lceil w^c_j/c\rceil$ c-tuples. Thus, the length of the sub-sequence that includes the first burst of j consecutive zero c-tuples is at least $j + \lceil w^c_j/c\rceil$ c-tuples and, hence, w^c_j must satisfy (3.45a).

(ii) Analogously to the proof of (i).

(iii) Since $w^s_{j_1,j_2}$ is the weight of the sub-sequence between the two bursts of j_1 and j_2 consecutive zeros, respectively, the total length including these bursts of zeros is at least $j_1 + \lceil w^s_{j_1,j_2}/c\rceil + j_2$. Clearly, the weight of a sub-sequence of this length is lower bounded by the corresponding active segment distance, which completes the proof. ■

Example 3.4: In Figure 3.3 the active distances for the generator matrix

$$G(D) = \begin{pmatrix} D^2 & 1+D & 1+D+D^2 \\ 1+D+D^2 & 1+D+D^2 & 1 \end{pmatrix} \tag{3.46}$$

are shown. From this the lower bonds on w^c_j, w^{rc}_j, and $w^s_{j_1,j_2}$ can be calculated as shown in Table 3.1. From the state paths in Figure 3.4 the minimum

	j		
	1	2	3
w_j^c	2	3	3
w_j^{rc}	2	2	3

$w^s_{j_1,j_2}$		j_2		
		1	2	3
	1	0	0	1
j_1	2	0	1	1
	3	1	1	1

Table 3.1: The lower bounds on w_j^c, w_j^{rc}, and $w^s_{j_1,j_2}$ from Theorem 3.5 for the generator matrix in (3.46).

	j		
	1	2	3
$\min\{w_j^c\}$	2	3	3
$\min\{w_j^{rc}\}$	2	2	3

$\min\{w^s_{j_1,j_2}\}$		j_2		
		1	2	3
	1	0	0	1
j_1	2	0	1	1
	3	1	1	1

Table 3.2: The minimum values of w_j^c, w_j^{rc}, and $w^s_{j_1,j_2}$ for the generator matrix in (3.46).

values of w_j^c, w_j^{rc}, and $w^s_{j_1,j_2}$ are calculated in Table 3.2. The calculated lower bounds are tight. □

Example 3.5: Consider the generator matrix

$$G(D) = \begin{pmatrix} 1 + D + D^4 & 1 + D^2 + D^3 + D^4 \end{pmatrix}. \tag{3.47}$$

From its active distances in Figure 3.5 the lower bounds on w_j^c, w_j^{rc}, and $w^s_{j_1,j_2}$ in Table 3.3 can be derived. Compared with the minimum of the true values in Table 3.4, we see that the bound on $w^s_{j_1,j_2}$ is not tight. □

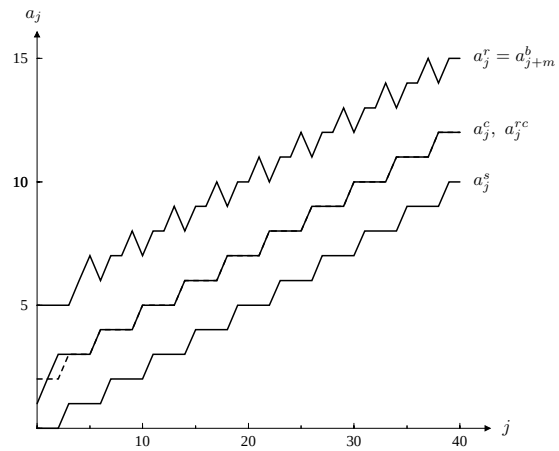

Figure 3.3: The active distances for the generator matrix in (3.46).

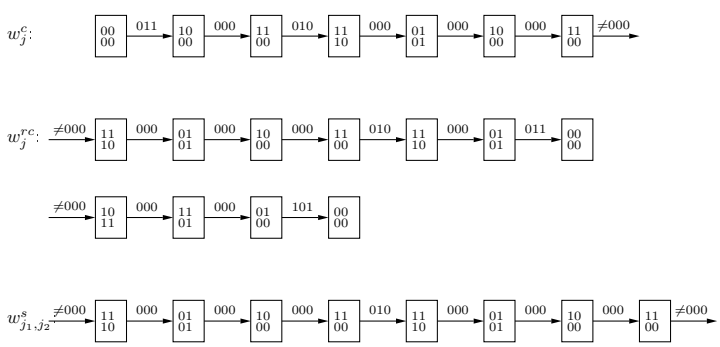

Figure 3.4: The state paths used to calculate w_j^c, w_j^{rc}, w_{j_1,j_2}^s in Example 3.4.

	j		
	1	2	3
w_j^c	3	4	4
w_j^{rc}	3	3	3

$w^s_{j_1,j_2}$		j_2		
		1	2	3
	1	0	0	1
j_1	2	0	1	1
	3	1	1	2

Table 3.3: The lower bounds on w_j^c, w_j^{rc}, and $w^s_{j_1,j_2}$ from Theorem 3.5 for the generator matrix in (3.47).

	j		
	1	2	3
$\min\{w_j^c\}$	3	4	4
$\min\{w_j^{rc}\}$	3	3	3

$\min\{w^s_{j_1,j_2}\}$		j_2		
		1	2	3
	1	0	0	1
j_1	2	0	1	1
	3	1	3	3

Table 3.4: The minimum values of w_j^c, w_j^{rc}, and $w^s_{j_1,j_2}$ for the generator matrix in (3.47).

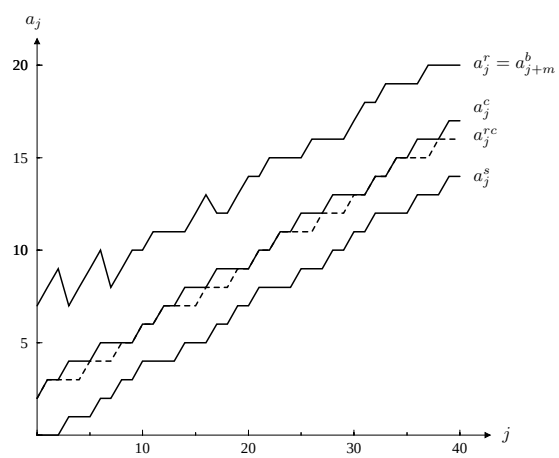

Figure 3.5: The active distances for the generator matrix in (3.47).

3.3. Lower Bounds on the Active Distances

The active row and burst distances typically have parts where they decrease, while the active column, reverse column, and segment distances are non-decreasing functions. However, for non-catastrophic generator matrices the active distances are in average increasing functions and can be lower bounded by affine functions,

$$\begin{cases} a_j^r \geqslant f^r(j) \triangleq \alpha j + \beta^r, \\ a_j^b \geqslant f^b(j) \triangleq \alpha j + \beta^b, \\ a_j^c \geqslant f^c(j) \triangleq \alpha j + \beta^c, \\ a_j^{rc} \geqslant f^{rc}(j) \triangleq \alpha j + \beta^{rc}, \\ a_j^s \geqslant f^s(j) \triangleq \alpha j + \beta^s, \end{cases} \tag{3.48}$$

where α is the asymptotic slope of the active distances and the β:s are chosen as large as possible. It follows that $\beta^r \geqslant \beta^b \geqslant \beta^c \geqslant \beta^s$ and $\beta^r \geqslant \beta^b \geqslant \beta^{rc} \geqslant \beta^s$. To see why $\beta^r \geqslant \beta^b$ assume that j is such that $a_j^r = \alpha j + \beta^r$ (clearly such a j exists). Then, for some constraint length ν_i, there is a sequence of length $j + 1 + \nu_i$ starting in the zero state and ending in the zero state, without consecutive zero states in between, with Hamming weight a_j^r. This sequence bounds the active burst distance as $a_j^r \geqslant a_{j+\nu_i}^b \geqslant \alpha(j+\nu_i) + \beta^b$, and we have

$$\alpha j + \beta^r \geqslant \alpha(j + \nu_i) + \beta^b \tag{3.49}$$

or, equivalently

$$\beta^r \geqslant \alpha\nu_i + \beta^b \geqslant \beta^b. \tag{3.50}$$

If $m = \nu_{\min}$, then we have directly $\beta^r = \alpha m + \beta^b$.

Let $\mathcal{A}$ denote the set of parameters $\{\alpha, \beta^r, \beta^c, \beta^{rc}, \beta^s, \beta^b\}$ of the lower bounds (3.48). In the following two examples we calculate $\mathcal{A}$ for two different rate $R = 1/2$ generator matrices with memory $m = 5$. These results will be used in Chapter 5 to find the set $\mathcal{A}$ for woven convolutional codes.

Example 3.6: In Figure 3.6 the active distances for the generator matrix

$$G(D) = \begin{pmatrix} 1 + D + D^2 + D^3 + D^5 & 1 + D^2 + D^3 + D^5 \end{pmatrix} \tag{3.51}$$

are shown together with the lower bounds defined by $\mathcal{A}$, where

$$\mathcal{A} = \left\{ \alpha = \frac{1}{3}, \beta^r = \frac{19}{3}, \beta^b = \frac{14}{3}, \beta^c = 2, \beta^{rc} = 1, \beta^s = -1 \right\}. \tag{3.52}$$

The difference between β^{rc} and β^c is due to the fact that the distance profile of the reverse generator matrix, $D^5 G(D^{-1})$, is not as good as the distance profile of $G(D)$. □

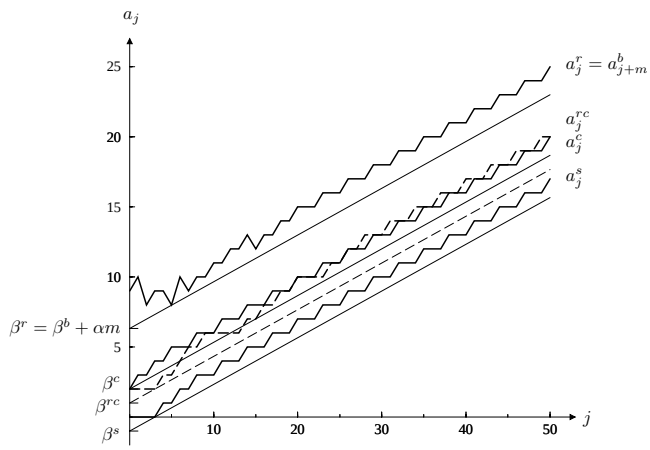

Figure 3.6: The active distances and the lower bounding affine functions for the generator matrix in (3.51).

Example 3.7: In Figure 3.7 the active distances and their lower bounds are shown for the generator matrix

$$G(D) = \begin{pmatrix} 1 + D + D^2 + D^3 + D^5 & 1 + D^2 + D^3 + D^4 + D^5 \end{pmatrix} \tag{3.53}$$

where

$$\mathcal{A} = \left\{ \alpha = \frac{2}{7}, \beta^r = \frac{46}{7}, \beta^b = \frac{36}{7}, \beta^c = \beta^{rc} = 2, \beta^s = -\frac{6}{7} \right\}. \tag{3.54}$$

The reciprocal generator matrix has the same distance properties as $G(D)$ since $D^5 G(D^{-1}) = G(D)\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, hence, $a_j^c = a_j^{rc}$ and $\beta^c = \beta^{rc}$. $\square$

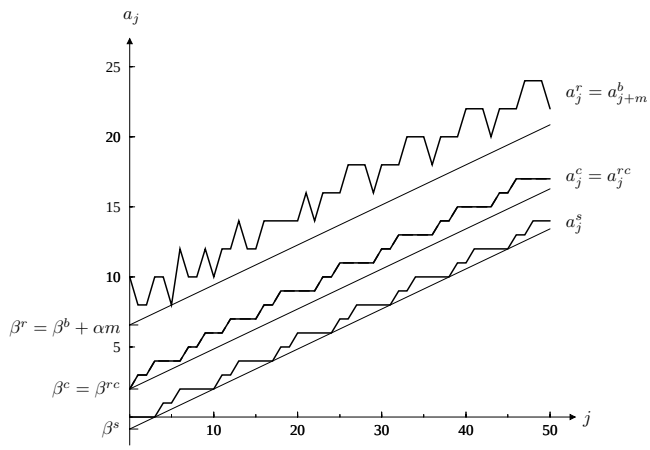

Figure 3.7: The active distances and the lower bounding affine functions for the generator matrix in (3.53).

3.4. Active Distances for Time-varying Convolutional Codes

Before we define the active distances for periodically time-varying convolutional codes encoded by time-varying polynomial generator matrices we introduce the following sets of information sequences, where we always assume that $t_1 \leqslant t_2$.

Let $\mathcal{U}^r_{[t_1-m,t_2+m]}$ denote the set of information sequences

$$\boldsymbol{u}_{[t_1-m,t_2+m]} = \boldsymbol{u}_{t_1-m}\boldsymbol{u}_{t_1-m+1}\dots\boldsymbol{u}_{t_2+m} \tag{3.55}$$

such that the first m and the last m sub-blocks are zero and such that they do not contain $m+1$ consecutive zero sub-blocks, i.e.,

$$\begin{aligned}\mathcal{U}^r_{[t_1-m,t_2+m]} \triangleq \Big\{\boldsymbol{u}_{[t_1-m,t_2+m]} \Big| &\boldsymbol{u}_{[t_1-m,t_1-1]} = \boldsymbol{0},\\ &\boldsymbol{u}_{[t_2+1,t_2+m]} = \boldsymbol{0}, \text{ and}\\ &\boldsymbol{u}_{[i,i+m]} \neq \boldsymbol{0}, t_1-m \leqslant i \leqslant t_2\Big\}.\end{aligned} \tag{3.56}$$

Let $\mathcal{U}^c_{[t_1-m,t_2]}$ denote the set of information sequences

$$\boldsymbol{u}_{[t_1-m,t_2]} = \boldsymbol{u}_{t_1-m}\boldsymbol{u}_{t_1-m+1}\dots\boldsymbol{u}_{t_2} \tag{3.57}$$

such that the first m sub-blocks are zero and such that they do not contain $m+1$ consecutive zero sub-blocks, i.e.,

$$\mathcal{U}^{c}_{[t_1-m,t_2]} \triangleq \Big\{ \boldsymbol{u}_{[t_1-m,t_2]} \Big| \boldsymbol{u}_{[t_1-m,t_1-1]} = \boldsymbol{0} \text{ and } \boldsymbol{u}_{[i,i+m]} \neq \boldsymbol{0}, t_1-m \leqslant i \leqslant t_2-m \Big\}. \tag{3.58}$$

Let $\mathcal{U}^{rc}_{[t_1-m,t_2+m]}$ denote the set of information sequences

$$\boldsymbol{u}_{[t_1-m,t_2+m]} = \boldsymbol{u}_{t_1-m}\boldsymbol{u}_{t_1-m+1}\ldots\boldsymbol{u}_{t_2+m} \tag{3.59}$$

such that the last m sub-blocks are zero and such that they do not contain $m+1$ consecutive zero sub-blocks, i.e.,

$$\mathcal{U}^{rc}_{[t_1-m,t_2+m]} \triangleq \Big\{ \boldsymbol{u}_{[t_1-m,t_2+m]} \Big| \boldsymbol{u}_{[t_2+1,t_2+m]} = \boldsymbol{0} \text{ and } \boldsymbol{u}_{[i,i+m]} \neq \boldsymbol{0}, t_1-m < i \leqslant t_2 \Big\}. \tag{3.60}$$

Let $\mathcal{U}^{s}_{[t_1-m,t_2]}$ denote the set of information sequences

$$\boldsymbol{u}_{[t_1-m,t_2]} = \boldsymbol{u}_{t_1-m}\boldsymbol{u}_{t_1-m+1}\ldots\boldsymbol{u}_{t_2} \tag{3.61}$$

such that they do not contain $m+1$ consecutive zero sub-blocks, i.e.,

$$\mathcal{U}^{s}_{[t_1-m,t_2]} \triangleq \Big\{ \boldsymbol{u}_{[t_1-m,t_2]} \Big| \boldsymbol{u}_{[i,i+m]} \neq \boldsymbol{0}, t_1-m < i < t_2-m \Big\}. \tag{3.62}$$

Next we introduce the $(j+m+1)\times(j+1)$ truncated, periodically time-varying generator matrix of memory m and period T:

$$\boldsymbol{G}_{[t,t+j]} = \begin{pmatrix} G_m(t) & & & \\ G_{m-1}(t) & G_m(t+1) & & \\ \vdots & G_{m-1}(t+1) & \ddots & \\ G_0(t) & \vdots & \ddots & G_m(t+j) \\ & G_0(t+1) & & G_{m-1}(t+j) \\ & & \ddots & \vdots \\ & & & G_0(t+j) \end{pmatrix}, \tag{3.63}$$

where $G_i(t) = G_i(t+T)$ for $0 \leqslant i \leqslant m$.

We are now well-prepared to generalize the definitions of the active distances for convolutional codes encoded by polynomial generator matrices to time-varying convolutional codes encoded by polynomial time-varying generator matrices:

Definition 3.7: Let $\mathcal{C}$ be a periodically time-varying convolutional code encoded by a periodically time-varying polynomial generator matrix of memory m and period T.

(*i*) The *jth order active row distance* is

$$a_j^r \triangleq \min_{0 \leqslant t < T} \min_{\mathcal{U}^r_{[t-m,t+j+m]}} \Big\{ w_{\mathrm{H}}\big(\boldsymbol{u}_{[t-m,t+j+m]} \boldsymbol{G}_{[t,t+j+m]}\big) \Big\}. \tag{3.64a}$$

(*ii*) The *jth order active burst distance* is for $j \geqslant m$

$$a_j^b \triangleq \min_{0 \leqslant t < T} \min_{\mathcal{U}^r_{[t-m,t+j]}} \Big\{ w_{\mathrm{H}}\big(\boldsymbol{u}_{[t-m,t+j]} \boldsymbol{G}_{[t,t+j]}\big) \Big\}. \tag{3.64b}$$

(*iii*) The *jth order active column distance* is

$$a_j^c \triangleq \min_{0 \leqslant t < T} \min_{\mathcal{U}^c_{[t-m,t+j]}} \Big\{ w_{\mathrm{H}}\big(\boldsymbol{u}_{[t-m,t+j]} \boldsymbol{G}_{[t,t+j]}\big) \Big\}. \tag{3.64c}$$

(*iv*) The *jth order active reverse column distance* is

$$a_j^{rc} \triangleq \min_{0 \leqslant t < T} \min_{\mathcal{U}^{rc}_{[t-m,t+j]}} \Big\{ w_{\mathrm{H}}\big(\boldsymbol{u}_{[t-m,t+j]} \boldsymbol{G}_{[t,t+j]}\big) \Big\}. \tag{3.64d}$$

(*v*) The *jth order active segment distance* is

$$a_j^s \triangleq \min_{0 \leqslant t < T} \min_{\mathcal{U}^s_{[t-m,t+j]}} \Big\{ w_{\mathrm{H}}\big(\boldsymbol{u}_{[t-m,t+j]} \boldsymbol{G}_{[t,t+j]}\big) \Big\}. \tag{3.64e}$$

□

For a periodically time-varying convolutional code encoded by a periodically time-varying, non-catastrophic, polynomial generator matrix with active row distance a_j^r we define its free distance by a generalization of (3.10)

$$d_{\mathrm{free}} \triangleq \min_j \{a_j^r\}. \tag{3.65}$$

When we in the next section we derive lower bounds on the active distances we need the following theorem.

Theorem 3.6: Consider a periodically time-varying, rate $R = b/c$, polynomial generator matrix of memory m and period T represented by $\boldsymbol{G}_t$, where $\boldsymbol{G}_t$ is given in (2.37).

(*i*) Let the information sequences be restricted to the set $\mathcal{U}^r_{[t-m,t+j+m]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j+m]}$ are mutually independent and equiprobable over the ensemble $\mathcal{E}(b,c,m,T)$ for all j, $0 \leqslant j < T$.

(*ii*) Let the information sequences be restricted to the set $\mathcal{U}^c_{[t-m,t+j]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{E}(b,c,m,T)$ for all j, $0 \leqslant j < \max\{m+1,T\}$.

(*iii*) Let the information sequences be restricted to the set $\mathcal{U}^{rc}_{[t-m,t+j]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{E}(b,c,m,T)$ for all j, $0 \leqslant j < \max\{m+1,T\}$.

(*iv*) Let the information sequences be restricted to the set $\mathcal{U}^s_{[t-m,t+j]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{E}(b,c,m,T)$ for all j, $0 \leqslant j < T$. □

Proof: It follows immediately that for $0 \leqslant j < T$ the code tuples $\boldsymbol{v}_i$, $i = t, t+1, \ldots, t+j$, are mutually independent and equiprobable in all four cases. Hence, the proof of (*iv*) is complete. In cases (*ii*) and (*iii*) it remains to show that the statements hold also for $T \leqslant j \leqslant m$ when $m \geqslant T$.

Consider the information sequences in the set $\mathcal{U}^c_{[t-m,t+j]}$, where $0 \leqslant j \leqslant m$. Let $t \leqslant i \leqslant t+j$, then, in the expression

$$\boldsymbol{v}_i = \boldsymbol{u}_i G_0(i) + \boldsymbol{u}_{i-1} G_1(i) + \cdots + \boldsymbol{u}_{i-m} G_m(i) \tag{3.66}$$

there exists a k, $0 \leqslant k \leqslant m$, such that at least one of the b-tuples $\boldsymbol{u}_{i-k}$ is non-zero and all the previous b-tuples $\boldsymbol{u}_{i-k'}$, $k < k' \leqslant m$, are zero. Hence, $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i'}$, $t \leqslant i < i' \leqslant t+j$, are mutually independent and equiprobable. This completes the proof of (*ii*).

Consider the information sequences in the set $\mathcal{U}^{rc}_{[t-m,t+j]}$, where $0 \leqslant j \leqslant m$. Let $t \leqslant i \leqslant t+j$, then, in (3.66) at least one of the b-tuples $\boldsymbol{u}_{i-k}$, $0 \leqslant k \leqslant m$, is non-zero and all the following b-tuples $\boldsymbol{u}_{i-k'}$, $0 \leqslant k' < k$, are zero. Hence, $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i'}$, $t \leqslant i < i' \leqslant t+j$, are mutually independent and equiprobable, which completes the proof of (*iii*).

For (*i*) it remains to show that $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i'}$ are mutually independent and equiprobable also for $T \leqslant i'-i < T+m$. From the definition of $\mathcal{U}^r_{[t-m,t+j+m]}$ it follows that $\boldsymbol{u}_{[t-m,t-1]} = \boldsymbol{0}$, $\boldsymbol{u}_t \neq \boldsymbol{0}$, $\boldsymbol{u}_{t+j} \neq \boldsymbol{0}$, and $\boldsymbol{u}_{[t+j+1,t+j+m]} = \boldsymbol{0}$. For $j = T$, we can choose, e.g., $\boldsymbol{u}_{[t-m,t+m]} = \boldsymbol{u}_{[t+T-m,t+T+m]} \in \mathcal{U}^r_{[t-m,t+T+m]}$ which implies that $\boldsymbol{v}_{[t,t+m]} = \boldsymbol{v}_{[t+T,t+T+m]}$. However, for $T-m \leqslant j < T$, $\boldsymbol{v}_i$, $t \leqslant i < t+m$, and $\boldsymbol{v}_{i'}$, $t+j < i' \leqslant t+j+m$, are mutually independent and equiprobable. ■

From Theorem 3.6 we have two corollaries.

Corollary 3.7: Consider a periodically time-varying, rate $R = b/c$, polynomial generator matrix of memory m and period T represented by $\boldsymbol{G}_t$, where $\boldsymbol{G}_t$ is given in (2.37). Let the information sequences be restricted to the set $\mathcal{U}^r_{[t-m,t+j]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{E}(b,c,m,T)$ for all j, $0 \leqslant j < T$. □

Corollary 3.8: Consider a rate $R = b/c$ polynomial generator matrix of memory m represented by $\boldsymbol{G}$, where $\boldsymbol{G}$ is given in (2.4).

(*i*) Let the information sequences be restricted to the set $\mathcal{U}^c_{[t-m,t+j]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{E}(b,c,m,1)$ for all j, $0 \leqslant j \leqslant m$.

(*ii*) Let the information sequences be restricted to the set $\mathcal{U}^{rc}_{[t-m,t+j+m]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{E}(b,c,m,1)$ for all j, $0 \leqslant j \leqslant m$. □

3.5. Lower Bounds on the Active Distances for Time-Varying Convolutional codes

In this section we shall derive lower bounds on the active distances for the ensemble of periodically time-varying convolutional codes. First we consider the active row distance and begin by proving a lemma.

Lemma 3.9: Consider the ensemble $\mathcal{E}(b,c,m,T)$ of binary, rate $R = b/c$, periodically time-varying convolutional codes encoded by polynomial generator matrices of memory m. The fraction of convolutional codes in this ensemble whose jth order active row distance a^r_j, $0 \leqslant j < T$, satisfies

$$a^r_j \leqslant \hat{a}^r_j, \tag{3.67}$$

where $\hat{a}^r_j < (j+m+1)c/2$, does not exceed

$$T 2^{\left(\frac{j+1}{j+m+1}R + h\left(\frac{\hat{a}^r_j}{(j+m+1)c}\right) - 1\right)(j+m+1)c}, \tag{3.68}$$

where $h(\cdot)$ is the binary entropy function (1.2). □

Proof: Let

$$\boldsymbol{v}_{[t,t+j+m]} = \boldsymbol{u}_{[t-m,t+j+m]}\boldsymbol{G}_{[t,t+j+m]}, \tag{3.69}$$

where $\boldsymbol{u}_{[t-m,t+j+m]} \in \mathcal{U}^r_{[t-m,t+j+m]}$ and assume that

$$\hat{a}^r_j < (j+m+1)c/2. \tag{3.70}$$

Then, it follows from Theorem 3.6 that

$$\begin{aligned} P\Big(w_{\mathrm{H}}(\boldsymbol{v}_{[t,t+j+m]}) \leqslant \hat{a}^r_j\Big) &= \sum_{i=0}^{\hat{a}^r_j} \binom{(j+m+1)c}{i} 2^{-(j+m+1)c} \\ &< 2^{\left(h\left(\frac{\hat{a}^r_j}{(j+m+1)c}\right)-1\right)(j+m+1)c}, \ 0 \leqslant j < T, \end{aligned} \tag{3.71}$$

where the last inequality follows from the standard inequality

$$\sum_{i=0}^{k} \binom{n}{i} < 2^{h\left(\frac{k}{n}\right)n}, \ k \leqslant n/2. \tag{3.72}$$

Notice that we need the denominator "2" in the right hand side of (3.70) in order to be able to apply (3.72). The cardinality of $\mathcal{U}^r_{[t-m,t+j+m]}$ is upper bounded by

$$\left|\mathcal{U}^r_{[t-m,t+j+m]}\right| \leqslant 2^{(j+1)b} = 2^{(j+1)Rc}. \tag{3.73}$$

Thus, we have

$$\begin{aligned} P\left(\min_{\mathcal{U}^r_{[t-m,t+j+m]}} \Big\{w_{\mathrm{H}}(\boldsymbol{v}_{[t,t+j+m]})\Big\} \leqslant \hat{a}^r_j\right) \\ < 2^{(j+1)Rc} 2^{\left(h\left(\frac{\hat{a}^r_j}{(j+m+1)c}\right)-1\right)(j+m+1)c} \\ = 2^{\left(\frac{j+1}{j+m+1}R + h\left(\frac{\hat{a}^r_j}{(j+m+1)c}\right)-1\right)(j+m+1)c} \end{aligned} \tag{3.74}$$

for each t, $0 \leqslant t < T$. Using the union bound completes the proof. ■

For a given f, $0 \leqslant f < 1$, let j_0 be the smallest integer j satisfying the inequality

$$\left(1 - \frac{j+1}{j+m+1}R\right)(j+m+1)c \geqslant \log\frac{T^2}{1-f}. \tag{3.75}$$

For large memories m such a value always exists. Let $\hat{a}_j^r$,

$$0 < \hat{a}_j^r < (j+m+1)c/2, \tag{3.76}$$

denote the largest integer that for given f, $0 \leqslant f < 1$, and j, $j \geqslant j_0$, satisfies the inequality

$$\left(\frac{j+1}{j+m+1}R + h\left(\frac{\hat{a}_j^r}{(j+m+1)c}\right) - 1\right)(j+m+1)c \leqslant -\log\frac{T^2}{1-f}. \tag{3.77}$$

Then, from Lemma 3.9 follows that for each j, $j_0 \leqslant j < T$, the fraction of convolutional codes with jth order active row distance satisfying (3.67) is upper bounded by

$$T2^{-\log\frac{T^2}{1-f}} = \frac{1-f}{T}. \tag{3.78}$$

Hence, we use the union bound and conclude that the fraction of convolutional codes with active row distance $a_j^r \leqslant \hat{a}_j^r$ for at least one j, $j_0 \leqslant j < T$, is upper bounded by

$$\sum_{j=j_0}^{T-m-1} \frac{1-f}{T} < 1-f. \tag{3.79}$$

We write this as a lemma.

Lemma 3.10: In the ensemble $\mathcal{E}(b,c,m,T)$ of periodically time-varying convolutional codes, the fraction of codes with active row distance

$$a_j^r > \hat{a}_j^r,\ j_0 \leqslant j < T, \tag{3.80}$$

is larger than f, where for a given f, $0 \leqslant f < 1$, j_0 is the smallest integer satisfying (3.75) and $\hat{a}_j^r$ the largest integer satisfying (3.77). □

By taking $f = 0$, we have the following corollary.

Corollary 3.11: There exists a binary, periodically time-varying, rate $R = b/c$, convolutional code encoded by a polynomial generator matrix of period T and memory m such that its jth order active row distance for $j_0 \leqslant j < T$ is lower bounded by $\hat{a}_j^r$, where $\hat{a}_j^r$ is the largest integer satisfying

$$\left(\frac{j+1}{j+m+1}R + h\left(\frac{\hat{a}_j^r}{(j+m+1)c}\right) - 1\right)(j+m+1)c \leqslant -2\log T \tag{3.81}$$

and j_0 is the smallest integer satisfying

$$\left(1 - \frac{j+1}{j+m+1}R\right)(j+m+1)c \geqslant 2\log T. \tag{3.82}$$

□

In order to get a better understanding for the significance of the previous lemma we shall study the asymptotic behavior of the parameters j_0 and $\hat{a}_j^r$ for large memories.

Let the period T grow as a power of m, say $T = m^2$. Then, since j_0 is an integer, for large values of m we have $j_0 = 0$. Furthermore, the inequality (3.81) can be rewritten as

$$h\left(\frac{\hat{a}_j^r}{(j+m+1)c}\right) \leqslant 1 - \frac{j+1}{j+m+1}R - \frac{4\log m}{(j+m+1)c}, \tag{3.83}$$

or, equivalently, as [1]

$$\hat{a}_j^r \leqslant h^{-1}\left(1 - \frac{j+1}{j+m+1}R\right)(j+m+1)c + O(\log m), \tag{3.84}$$

Combining this with Lemma 3.10 we have the following theorem.

Theorem 3.12: There exists a binary, periodically time-varying, rate $R = b/c$, convolutional code encoded by a polynomial generator matrix of memory m that has a jth order active row distance satisfying the inequality

$$a_j^r > h^{-1}\left(1 - \frac{j+1}{j+m+1}R\right)(j+m+1)c + O(\log m), \tag{3.85}$$

for $j \geqslant 0$. □

From the definitions of the active distances, Definition 3.7, it is clear that $a_j^b = a_{j+m}^r$, and we have the corresponding theorem.

Theorem 3.13: There exists a binary, periodically time-varying, rate $R = b/c$, convolutional code encoded by a polynomial generator matrix of memory m that has a jth order active burst distance satisfying the inequality

$$a_j^b > h^{-1}\left(1 - \frac{j-m+1}{j+1}R\right)(j+m+1)c + O(\log m), \tag{3.86}$$

for $j \geqslant m$. □

[1] Here and hereafter we write $h^{-1}(y)$ for the *smallest* x such that $y = h(x)$.

The main term in (3.85) can be obtained from the Gilbert-Varshamov bound for block codes using a geometrical construction that is similar to Forney's inverse concatenated construction [15].

Consider the Gilbert-Varshamov lower bound on the normalized minimum distance for block codes [39], viz.,

$$\frac{d_{\min}}{N} \geqslant h^{-1}(1-R), \tag{3.87}$$

where N denotes the block length. Let

$$\delta^r(j) = \frac{h^{-1}\left(1 - \frac{j+1}{j+1+m}R\right)(j+1+m)c}{mc} \tag{3.88}$$

denote the main term of the right hand side of (3.85) normalized by mc.

The construction is illustrated in Figure 3.8 for $R = 1/2$. The straight line between the points $(0, \delta^r(j))$ and $(R, 0)$ intersects $h^{-1}(1-R)$ in the point $(r, h^{-1}(1-r))$. The rate r is chosen to be

$$r = \frac{j+1}{j+1+m}R, \tag{3.89}$$

i.e., it divides the line between $(0,0)$ and $(R,0)$ in the proportion $(j+1):m$. Then we have

$$\frac{\delta^r(j)}{h^{-1}(1-r)} = \frac{j+1+m}{m}, \tag{3.90}$$

which is equivalent to (3.88).

We shall now derive a corresponding lower bound on the active column distance. Let

$$\boldsymbol{v}_{[t,t+j]} = \boldsymbol{u}_{[t-m,t+j]}\boldsymbol{G}_{[t,t+j]}, \tag{3.91}$$

where $\boldsymbol{u}_{[t-m,t+j]} \in \mathcal{U}^c_{[t-m,t+j]}$ and let $\hat{a}^c_j$ be an integer satisfying the inequality

$$\hat{a}^c_j < (j+1)c/2. \tag{3.92}$$

Then, as a counterpart to (3.71) we have

$$\begin{aligned} P\Big(w_{\mathrm{H}}(\boldsymbol{v}_{[t,t+j]}) \leqslant \hat{a}^c_j\Big) &= \sum_{i=0}^{\hat{a}^c_j} \binom{(j+1)c}{i} 2^{-(j+1)c} \\ &< 2^{\left(h\left(\frac{\hat{a}^c_j}{(j+1)c}\right)-1\right)(j+1)c}, \ 0 \leqslant j < T. \end{aligned} \tag{3.93}$$

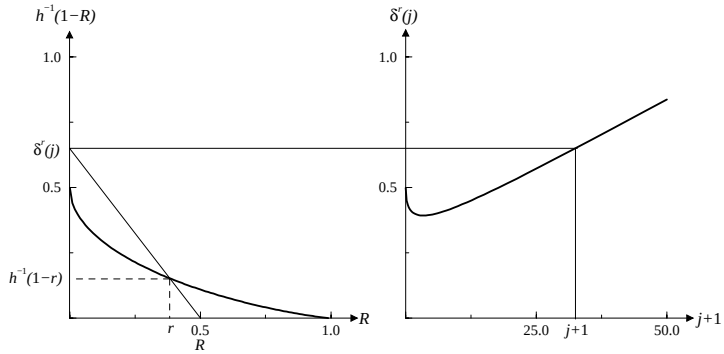

Figure 3.8: Geometrical construction of the relationship between the lower bound on the active row distance for convolutional codes and the Gilbert-Varshamov lower bound on the minimum distance for block codes.

The cardinality of $\mathcal{U}^c_{[t-m,t+j]}$ is upper bounded by $|\mathcal{U}^c_{[t-m,t+j]}| \leqslant 2^{(j+1)Rc}$ and we obtain

$$
\begin{aligned}
(3.94)\quad P\left(\min_{\mathcal{U}^c_{[t-m,t+j]}} \left\{w_{\mathrm{H}}(\boldsymbol{v}_{[t,t+j]})\right\} \leqslant \hat{a}^c_j\right) \\
< 2^{(j+1)Rc} 2^{\left(h\left(\frac{\hat{a}^c_j}{(j+1)c}\right)-1\right)(j+1)c} \\
= 2^{\left(R+h\left(\frac{\hat{a}^c_j}{(j+1)c}\right)-1\right)(j+1)c},
\end{aligned}
$$

for each t, $0 \leqslant t < T$. Minimizing over $0 \leqslant t < T$ and using the union bound complete the proof of the next lemma.

Lemma 3.14: Consider the ensemble $\mathcal{E}(b,c,m,T)$ of binary, rate $R = b/c$, periodically time-varying convolutional codes encoded by polynomial generator matrices of memory m. The fraction of convolutional codes in this ensemble whose jth order active column distance a^c_j, $0 \leqslant j < T$, satisfies

$$
(3.95)\qquad a^c_j \leqslant \hat{a}^c_j,
$$

where $\hat{a}^c_j < (j+1)c/2$, does not exceed

$$
(3.96)\qquad T 2^{\left(R+h\left(\frac{\hat{a}^c_j}{(j+1)c}\right)-1\right)(j+1)c}.
$$

□

Choose j_0 to be the smallest integer j satisfying the inequality

$$(1-R)(j+1)c \geqslant \log T^2. \tag{3.97}$$

Let $\hat{a}_j^c$,

$$0 < \hat{a}_j^c < (j+1)c/2, \tag{3.98}$$

denote the largest integer that for given j, $j \geqslant j_0$, satisfies the inequality

$$\left(R + h\left(\frac{\hat{a}_j^c}{(j+1)c}\right) - 1\right)(j+1)c \leqslant -\log T^2. \tag{3.99}$$

Then, from Lemma 3.14 follows that for each j, $j_0 \leqslant j < T$, the fraction of convolutional codes with a jth order active column distance satisfying (3.95) is upper bounded by

$$T2^{-\log T^2} = \frac{1}{T}. \tag{3.100}$$

Hence, we use the union bound and conclude that the fraction of convolutional codes with active column distance $a_j^c \leqslant \hat{a}_j^c$ for at least one j, $j_0 \leqslant j < T$, is upper bounded by

$$\sum_{j=j_0}^{T-1} \frac{1}{T} < 1. \tag{3.101}$$

Thus, we have proved the following lemma.

Lemma 3.15: There exists a periodically time-varying, rate $R = b/c$, convolutional code encoded by a polynomial generator matrix of period T and memory m such that its jth order active column distance for $j_0 \leqslant j < T$ is lower bounded by $\hat{a}_j^c$, where $\hat{a}_j^c$ is the largest integer satisfying

$$\left(R + h\left(\frac{\hat{a}_j^c}{(j+1)c}\right) - 1\right)(j+1)c \leqslant -2\log T \tag{3.102}$$

and j_0 is the smallest integer satisfying

$$(1-R)(j+1)c \geqslant 2\log T. \tag{3.103}$$

□

If we as before choose $T = m^2$, then $j_0 = O(\log m)$, and the inequality (3.102) can be rewritten as

$$h\left(\frac{\hat{a}_j^c}{(j+1)c}\right) \leqslant 1 - R - \frac{4\log m}{(j+1)c} \tag{3.104}$$

for $j = O(m)$ or, equivalently, as

$$\hat{a}_j^c \leqslant h^{-1}(1-R)(j+1)c + O(\log m). \tag{3.105}$$

This together with Lemma 3.14 gives the next theorem.

Theorem 3.16: There exists a binary, periodically time-varying, rate $R = b/c$, convolutional code encoded by a polynomial generator matrix of memory m that has a jth order active column distance satisfying the inequality

$$a_j^c > h^{-1}(1-R)(j+1)c + O(\log m), \tag{3.106}$$

for $j = O(m) > j_0 = O(\log m)$. □

Analogously we can prove

Theorem 3.17: There exists a binary, periodically time-varying, rate $R = b/c$, convolutional code encoded by a polynomial generator matrix of memory m that has a jth order active reverse column distance a_j^{rc} which is lower bounded by the right hand side of the inequality (3.106) for all $j > j_0 = O(\log m)$. □

For the active segment distance we have the following

Theorem 3.18: There exists a binary, periodically time-varying, rate $R = b/c$, convolutional code encoded by a polynomial generator matrix of memory m that has a jth order active segment distance satisfying the inequality

$$a_j^s > h^{-1}\left(1 - \frac{j+m+1}{j+1}R\right)(j+1)c + O(\log m), \tag{3.107}$$

for $j = O(m) > j_0^s$, where

$$j_0^s < \frac{R}{1-R}m + O(\log m). \tag{3.108}$$

□

Proof: Consider the ensemble $\mathcal{E}(b, c, m, T)$. First we notice that the cardinality of $\mathcal{U}^s_{[t,t+j]}$ is upper bounded by

$$|\mathcal{U}^s_{[t,t+j]}| \leqslant 2^{mb} 2^{(j+1)b} = 2^{(j+m+1)Rc}. \tag{3.109}$$

Using (3.109) instead of (3.73) and repeating the steps in the derivation of the lower bound on the active column distance will give

$$h\left(\frac{\hat{a}^s_j}{(j+1)c}\right) \leqslant 1 - \frac{j+m+1}{j+1}R - \frac{4\log m}{(j+1)c} \tag{3.110}$$

for all $j = O(m) > j^s_0$, or, equivalently,

$$\hat{a}^s_j \leqslant h^{-1}\left(1 - \frac{j+m+1}{j+1}R\right)(j+1)c + O(\log m), \tag{3.111}$$

where

$$0 < \hat{a}^s_j < (j+1)c/2, \tag{3.112}$$

instead of (3.104), (3.105), and (3.98), respectively, and the proof is complete.

The parameter j^s_0 is the start of the active segment distance (cf. Figure 3.1). ■

Next we consider our lower bounds on the active distances, viz., (3.85), (3.86), (3.106), and (3.107), and introduce the substitution

$$\ell = \frac{j+1}{m}, \tag{3.113}$$

then we obtain asymptotically—for large memories m—the following lower bounds on the *normalized active distances.*

Theorem 3.19: In the ensemble of binary, periodically time-varying, rate $R = b/c$, convolutional codes encoded by polynomial generator matrices of memory m,

(i) there exists a code whose normalized active row distance asymptotically satisfies

$$\delta^r_\ell \triangleq \frac{a^r_j}{mc} \geqslant h^{-1}\left(1 - \frac{\ell}{\ell+1}R\right)(\ell+1) + O\left(\frac{\log m}{m}\right) \tag{3.114a}$$

for $\ell \geqslant 0$.

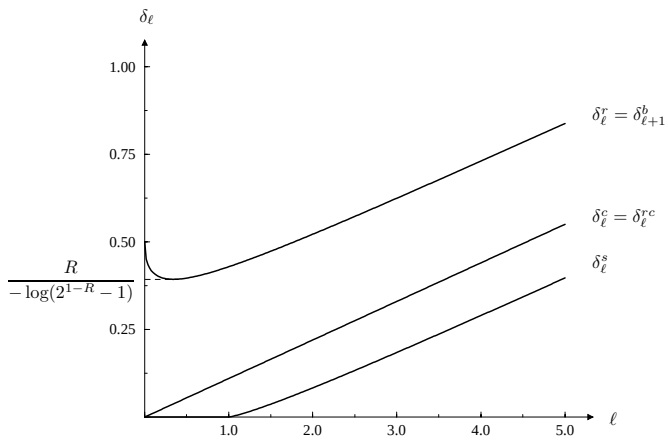

Figure 3.9: Typical behavior of the lower bounds on the normalized active distances of Theorem 3.19.

(*ii*) there exists a code whose normalized active burst distance asymptotically satisfies

$$\delta_\ell^b \triangleq \frac{a_j^b}{mc} \geqslant h^{-1}\left(1 - \frac{\ell-1}{\ell}R\right)\ell + O\left(\frac{\log m}{m}\right) \tag{3.114b}$$

for $\ell \geqslant 1$.

(*iii*) there exists a code whose normalized active column (reverse column) distance asymptotically satisfies

$$\left.\begin{aligned} \delta_\ell^c &\triangleq \frac{a_j^c}{mc} \\ \delta_\ell^{rc} &\triangleq \frac{a_j^{rc}}{mc} \end{aligned}\right\} \geqslant h^{-1}(1-R)\ell + O\left(\frac{\log m}{m}\right) \tag{3.114c}$$

for $\ell \geqslant \ell_0 = O\left(\frac{\log m}{m}\right)$.

(*iv*) there exists a code whose normalized active segment distance asymptotically satisfies

$$\delta_\ell^s \triangleq \frac{a_j^s}{mc} \geqslant h^{-1}\left(1 - \frac{\ell+1}{\ell}R\right)\ell + O\left(\frac{\log m}{m}\right) \tag{3.114d}$$

for $\ell \geqslant \ell_0^s = \frac{R}{1-R} + O\left(\frac{\log m}{m}\right)$. □

The typical behavior of the bounds in Theorem 3.19 is shown in Figure 3.9. Notice that by minimizing the lower bound on the normalized active row distance (3.114a) or, equivalently, the lower bound on the normalized active burst distance (3.114b) we obtain nothing but the main term in Costello's lower bound on the free distance [9], viz.,

$$\delta_C(R) = \frac{R}{-\log(2^{1-R} - 1)}. \tag{3.115}$$

4

Cascaded Convolutional Codes

Concatenation is a both powerful and practical method to obtain communication systems with low error probabilities. In this chapter the simplest such construction is considered, viz., a cascade of two convolutional encoders [23, 30] (see also [31]). The aim is to understand the basics of a concatenated construction. However, to get a system with good error performance for low signal to noise ratios some sort of symbol-wise permutations of the sequence between the encoders is required.

In Section 4.1 cascaded convolutional codes are introduced. Structural properties are considered first for cascaded convolutional codes with matched rates in Section 4.2 and then for unmatched rates in Section 4.3. In Section 4.4 the constituent generator matrices are replaced by equivalent ones and in Section 4.5 those equivalent generator matrices are systematic. In Section 4.6 lower bounds on the active distances for fixed time-invariant cascaded convolutional codes are derived, while Section 4.7 is devoted to ensemble properties, where lower bounds on the active distances and free distance are derived for time-varying cascaded convolutional codes.

4.1. Cascaded Convolutional Codes

A *cascaded convolutional encoder* is a cascade of one outer rate $R_o = b_o/c_o$ encoder of memory m_o and one inner rate $R_i = b_i/c_i$ encoder with memory m_i, see Figure 4.1. Each binary b_o-tuple of the information sequence is encoded into a binary c_o-tuple. The encoded sequence is serialized and directly, without any permutations, fed as the information sequence for the inner encoder, where each binary b_i-tuple is encoded into a binary c_i-tuple. Thus, the overall rate of the cascaded encoder is

$$R_c = \frac{b_c}{c_c} = \frac{b_o}{\frac{c_o}{b_i}c_i} = \frac{b_o b_i}{c_o c_i} = R_o R_i. \tag{4.1}$$

We say that the outer and inner encoders have *matched rates* if the outer code tuples serve directly as information tuples for the inner encoder, i.e., if $b_i = c_o$. Then the overall rate is $R_c = b_o/c_i$.

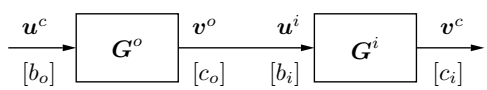

Figure 4.1: A cascade of two convolutional encoders.

A *cascaded convolutional code* is a convolutional code encoded by a cascaded convolutional encoder. From

$$\boldsymbol{v}^c = \boldsymbol{u}^i \boldsymbol{G}^i = \boldsymbol{u}^c \boldsymbol{G}^o \boldsymbol{G}^i \tag{4.2}$$

we see that the cascaded generator matrix $\boldsymbol{G}^c$ is given by

$$\boldsymbol{G}^c = \boldsymbol{G}^o \boldsymbol{G}^i, \tag{4.3}$$

where $\boldsymbol{G}^o$ and $\boldsymbol{G}^i$ are the generator matrices for the outer and inner encoders, respectively.

If the constituent encoders have matched rates, then equation (4.3) can, equivalently, be expressed as

$$G^c(D) = G^o(D) G^i(D). \tag{4.4}$$

When the rates are not matched, i.e., $b_i \neq c_o$, the matrix multiplication in (4.4) is not defined, but (4.3) is still valid.

4.2. Structural Properties of Cascaded Convolutional Codes

In this section we will show some structural properties for cascaded convolutional generator matrices and encoders. We assume that the constituent generator matrices have matched rates. In the next section we will consider the situation when the encoders have unmatched rates.

Let $G^c_{can}(D)$ be a canonical generator matrix equivalent to the cascaded generator matrix,

$$G^c(D) = G^o(D)G^i(D), \tag{4.5}$$

and denote by m_{can} and ν_{can} its memory and overall constraint length, respectively. An obvious way of realizing the cascaded generator matrix $G^c(D)$ is to realize both $G^o(D)$ and $G^i(D)$ on controller canonical form. Such a realization requires $\nu_o + \nu_i$ memory elements, where ν_o and ν_i are the overall constraint lengths for the outer and inner generator matrices, respectively. We know that an encoder realized on controller canonical form of a canonical generator matrix is a minimal encoder, i.e., it requires a minimum number of delay elements over all encoders for the code. Since such a realization of $G^c_{can}(D)$ has ν_{can} delay elements we have the following theorem.

Theorem 4.1: Let $G^c_{can}(D)$, with overall constraint length ν_{can}, be a canonical generator matrix equivalent to the cascaded generator matrix $G^c(D)$ defined by the product of the two generator matrices $G^o(D)$ and $G^i(D)$, with matched rates and overall constraint lengths ν_o and ν_i, respectively, then

$$\nu_{can} \leqslant \nu_o + \nu_i. \tag{4.6}$$

□

A similar theorem for the memory of a cascaded generator matrix can be stated.

Theorem 4.2: Let $G^c_{can}(D)$ of memory m_{can} be a canonical generator matrix equivalent to the cascaded generator matrix, $G^c(D)$, of memory m_c defined as the product of the two generator matrices $G^o(D)$ and $G^i(D)$ with matched rates and of memories m_o and m_i, respectively, then

$$m_{can} \leqslant m_c \leqslant m_o + m_i. \tag{4.7}$$

□

Proof: It can be shown that the memory of a canonical generator matrix is minimal over all equivalent generator matrices [35]. Thus, $m_{can} \leqslant m_c$, where m_c is the memory of the cascaded generator matrix, $G^c(D)$. Furthermore, the product of two generator matrices cannot have larger memory than the sum of the constituent memories, thus,

$$m_{can} \leqslant m_c \leqslant m_o + m_i. \tag{4.8}$$

■

The next example [43] on polynomial generator matrices further explains the two theorems.

Example 4.1: Consider the rate $R_o = 2/3$ outer generator matrix

$$G^o(D) = \begin{pmatrix} 1 & D & 1+D \\ D^2 & 1 & 1+D+D^2 \end{pmatrix} \tag{4.9}$$

of memory $m_o = 2$ and overall constraint length $\nu_o = 3$ and the rate $R_i = 3/4$ inner generator matrix

$$G^i(D) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1+D & D & 1 \\ 0 & D & 1+D^2 & 1+D^2 \end{pmatrix} \tag{4.10}$$

of memory $m_i = 2$ and overall constraint length $\nu_i = 3$. The corresponding cascaded generator matrix is

$$\begin{aligned} G^c(D) &= G^o(D)G^i(D) \\ &= \begin{pmatrix} 1 & 1 & D+D^3 & D^2+D^3 \\ D^2 & 1+D^3 & 1+D^2+D^3+D^4 & D+D^2+D^3+D^4 \end{pmatrix} \end{aligned} \tag{4.11}$$

of rate $R_c = 2/4$, memory $m_c = 4$ $(= m_o + m_i)$, and overall constraint length $\nu_c = 7$ $(> \nu_o + \nu_i)$.

Both $G^o(D)$ and $G^i(D)$ are minimal-basic (canonical), while $G^c(D)$ is not. As expected, the product of two canonical generator matrices is not necessarily canonical. The generator matrix

$$G^c_{can}(D) = \begin{pmatrix} 1 & 1 & D+D^3 & D^2+D^3 \\ D+D^2 & 1+D+D^3 & 1+D^3 & D+D^2 \end{pmatrix} \tag{4.12}$$

of memory $m_{can} = 3$ and overall constraint length $\nu_{can} = 6$ is a minimal-basic (canonical) equivalent of $G^c(D)$. Comparing the memories and overall constraint lengths for the constituent generator matrices with this one we get $m_{can} = 3 < m_o + m_i = 4$ and $\nu_{can} = \nu_o + \nu_i = 6$. Thus, the upper bound on the constraint lengths in Theorem 4.1 is tight. □

Example 4.1 shows that the overall constraint length of the cascaded generator matrix can actually exceed the sum of the overall constraint lengths for the constituent generator matrices, i.e., $\nu_c \nleqslant \nu_o + \nu_i$. To get an estimate of the overall constraint length for the cascaded generator matrix we take a closer look at the multiplication of the constituent generator matrices. Let $\nu_{o,k}$, $1 \leqslant k \leqslant b_o$, be the constraint length of the kth row of the outer generator matrix and $\nu_{i,l}^T$, $1 \leqslant l \leqslant c_i$, the "constraint lengths" of the lth row of the transpose of the inner generator matrix, $(G^i(D))^T$. Now, the kth constraint length of the cascaded generator matrix can be bounded by

$$\nu_{c,k} \leqslant \max_{1 \leqslant l \leqslant c_i} \left\{ \nu_{o,k} + \nu_{i,l}^T \right\} = \nu_{o,k} + m_i, \ 1 \leqslant k \leqslant b_o. \tag{4.13}$$

Therefore, the overall constraint length for the cascaded generator matrix satisfies

$$\nu_c = \sum_{j=1}^{b_o} \nu_{c,k} \leqslant \sum_{j=1}^{b_o} (\nu_{o,k} + m_i) = \nu_o + b_o m_i, \tag{4.14}$$

which we state as a theorem.

Theorem 4.3: Let $G^c(D)$ of overall constraint length ν_c be a cascaded generator matrix defined by the product of the rate $R_o = b_o/c_o$ outer generator matrix $G^o(D)$ of overall constraint length ν_o and the rate $R_i = b_i/c_i$ inner generator matrix $G^i(D)$ of memory m_i where $b_i = c_o$, then

$$\nu_c \leqslant \nu_o + b_o m_i. \tag{4.15}$$

□

In Example 4.1 this gives $\nu_c \leqslant 3 + 2 \cdot 2 = 7$ and we see that the bound in (4.15) is tight. Note that the sum $\nu_o + b_o m_i$ can be less than $\nu_o + \nu_i$. Thus, we can actually strengthen the bound of Theorem 4.1 to be the minimum of those two values.

The generator matrix $G^c(D)$ in (4.11) is not canonical, but the next theorem shows that it is in fact minimal.

Theorem 4.4: If $G^o(D)$ and $G^i(D)$ are two minimal generator matrices with matched rates, then the cascaded generator matrix defined by their product, $G^c(D) = G^o(D)G^i(D)$, is also minimal. □

Proof: The *span* of a Laurent series in D is the set of indices from the first non-zero component to the last non-zero component, if there is one, otherwise

to infinity. In [16] it is shown that a generator matrix $G(D)$ is minimal if and only if the span of the information sequence is contained in the span of the code sequence,

$$\text{span}(\boldsymbol{u}(D)) \subseteq \text{span}(\boldsymbol{u}(D)G(D)). \tag{4.16}$$

If both the outer and inner generator matrices are minimal, then

$$\begin{aligned} \text{span}(\boldsymbol{u}^c(D)) &\subseteq \text{span}(\boldsymbol{u}^c(D)G^o(D)) \\ &\subseteq \text{span}(\boldsymbol{u}^c(D)G^o(D)G^i(D)) \\ &= \text{span}(\boldsymbol{u}^c(D)G^c(D)), \end{aligned} \tag{4.17}$$

where the first inclusion is equivalent to the minimality of $G^o(D)$ and the second to the minimality of $G^i(D)$. ■

In Example 4.1 both the outer and inner generator matrices were basic, as was the cascaded generator matrix. The next theorem will show that this was not a coincident.

Theorem 4.5: If $G^o(D)$ and $G^i(D)$ are two basic generator matrices with matched rates, then the cascaded generator matrix defined by their product, $G^c(D) = G^o(D)G^i(D)$, is also basic. □

Proof: Since $G^o(D)$ and $G^i(D)$ are basic, they are both polynomial and have polynomial right inverses. The cascaded generator matrix will, of course, be polynomial and it will have a polynomial right inverse, viz.,

$$(G^c(D))^{-1} = (G^o(D)G^i(D))^{-1} = (G^i(D))^{-1}(G^o(D))^{-1}. \tag{4.18}$$

■

The last example in this section will show that also the bound in Theorem 4.2 is also tight.

Example 4.2: Consider the rate $R_o = 1/2$ outer generator matrix

$$G^o(D) = \begin{pmatrix} 1 & \frac{D}{1+D} \end{pmatrix} \tag{4.19}$$

of memory $m_o = 1$ and overall constraint length $\nu_o = 1$ and the rate $R_i = 2/3$ inner generator matrix

$$G^i(D) = \begin{pmatrix} \frac{1}{1+D} & \frac{1}{1+D} & \frac{D}{1+D} \\ 1 & \frac{D}{1+D} & \frac{1}{1+D} \end{pmatrix} \tag{4.20}$$

of memory $m_i = 1$ and overall constraint length $\nu_i = 2$.

Both constituent generator matrices are canonical and, in contrast to Example 4.1, so is their product

$$G^c_{can}(D) = G^c(D) = \begin{pmatrix} 1 & \frac{1+D+D^2}{1+D^2} & \frac{D^2}{1+D^2} \end{pmatrix}. \tag{4.21}$$

Here the overall constraint length is $\nu_{can} = 2 = \nu_o + b_o m_i < \nu_o + \nu_o = 3$. The memory $m_{can} = m_o + m_i = 2$, and the upper bound on the memories in Theorem 4.2 is tight. □

4.3. Cascaded Convolutional Codes with Unmatched Rates

In the case when $b_i \neq c_o$ the product of $G^o(D)$ and $G^i(D)$ is not defined. We can, however, still multiply the semi-infinite matrices $\boldsymbol{G}^o$ and $\boldsymbol{G}^i$ to get the cascaded generator matrix $\boldsymbol{G}^c = \boldsymbol{G}^o\boldsymbol{G}^i$. We will start by taking a closer look at the rate for this generator matrix. A block of b_i information symbols (binary b_o-tuples) will be encoded into b_i outer code symbols (c_o-tuples). This binary $b_i c_o$-tuple can be represented as c_o inner information symbols which will be encoded into c_o inner code symbols. Assume that $d = \gcd(c_o, b_i)$. Then, in the same way, b_i/d outer information symbols will generate c_o/d code symbols from the inner encoder. We write this as a lemma.

Lemma 4.6: The rate of a cascaded generator matrix defined by $\boldsymbol{G}^c = \boldsymbol{G}^o\boldsymbol{G}^i$ is

$$R_c = \frac{b_o b_d}{c_d c_i} \tag{4.22}$$

where $b_d = \dfrac{b_i}{\gcd(c_o, b_i)}$ and $c_d = \dfrac{c_o}{\gcd(c_o, b_i)}$. □

For matched rates, $b_i = c_o$, we get $b_d = c_d = 1$. From Lemma 4.6 it is clear that the matrix $G^c(D)$ has size $b_o b_d \times c_d c_i$. This matrix can be found by enlarging the sub-matrices in $\boldsymbol{G}^o$ by a factor b_d, and in $\boldsymbol{G}^i$ by c_d, in a similar way as when deriving the unit memory representative of a generator matrix [37]. The resulting matrices $G^o_{[b_d]}(D)$ and $G^i_{[c_d]}(D)$ then have matched rates and can be multiplied. The next example will explain the procedure. For simplicity we use polynomial generator matrices.

Example 4.3: Let

$$G^o(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix} \tag{4.23}$$

and

$$G^i(D) = \begin{pmatrix} 1+D & D & D & 1+D \\ D^2 & 1+D & 0 & 1+D+D^2 \\ 0 & D & 1+D^2 & 1+D^2 \end{pmatrix}. \tag{4.24}$$

These minimal-basic (canonical) matrices cannot be multiplied directly,. We have $d = \gcd(c_o, b_i) = 1$, and the enlargement factors are $b_d = 3$ and $c_d = 2$. The rate of the cascaded convolutional generator matrix is $R_c = \frac{1}{2}\frac{3}{4} = \frac{3}{8}$. The outer generator matrix can be written as

$$\boldsymbol{G}^o = \begin{pmatrix} 11 & 10 & 11 & & & & & & \\ & 11 & 10 & 11 & & & & & \\ & & 11 & 10 & 11 & & & & \\ & & & 11 & 10 & 11 & & & \\ & & & & 11 & 10 & 11 & & \\ & & & & & 11 & 10 & 11 & \\ & & & & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{4.25}$$

Instead of 1×2 sub-matrices we consider three times larger ones, 3×6, and express the matrix in the $\mathcal{D}$-transform representation. The three times enlarged variant of $G^o(D)$ is

$$\begin{aligned} G^o_{[3]}(D) &= \begin{pmatrix} 11 & 10 & 11 \\ 00 & 11 & 10 \\ 00 & 00 & 11 \end{pmatrix} + \begin{pmatrix} 00 & 00 & 00 \\ 11 & 00 & 00 \\ 10 & 11 & 00 \end{pmatrix} D \\ &= \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ D & D & 1 & 1 & 1 & 0 \\ D & 0 & D & D & 1 & 1 \end{pmatrix}. \end{aligned} \tag{4.26}$$

The inner generator matrix is

$$\boldsymbol{G}^i = \begin{pmatrix} 1001 & 1111 & 0000 & & & \\ 0101 & 0101 & 1001 & & & \\ 0011 & 0100 & 0011 & & & \\ & 1001 & 1111 & 0000 & & \\ & 0101 & 0101 & 1001 & & \\ & 0011 & 0100 & 0011 & & \\ & & 1001 & 1111 & 0000 & \\ & & 0101 & 0101 & 1001 & \\ & & 0011 & 0100 & 0011 & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{4.27}$$

To double the size of the generator matrix in the $\mathcal{D}$-transform representation

we get

$$
\begin{aligned}
(4.28)\qquad G^i_{[2]}(D) &= \begin{pmatrix} 1001 & 1111 \\ 0101 & 0101 \\ 0011 & 0100 \\ & 1001 \\ & 0101 \\ & 0011 \end{pmatrix} + \begin{pmatrix} 0000 & \\ 1001 & \\ 0011 & \\ 1111 & 0000 \\ 0101 & 1001 \\ 0100 & 0011 \end{pmatrix} D \\
&= \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ D & 1 & 0 & 1+D & 0 & 1 & 0 & 1 \\ 0 & 0 & 1+D & 1+D & 0 & 1 & 0 & 0 \\ D & D & D & D & 1 & 0 & 0 & 1 \\ 0 & D & 0 & D & D & 1 & 0 & 1+D \\ 0 & D & 0 & 0 & 0 & 0 & 1+D & 1+D \end{pmatrix}.
\end{aligned}
$$

Using these enlarged matrices, the semi-infinite matrix multiplication $\boldsymbol{G}^c = \boldsymbol{G}^o\boldsymbol{G}^i$ can be equivalently represented by

$$
\begin{aligned}
(4.29)\qquad G^c(D) &= G^o_{[3]}(D)G^i_{[2]}(D) \\
&= \begin{pmatrix} 1+D & 1 & 1+D & 1+D & 1+D & 0 & D & 0 \\ D^2 & D & 1 & 1+D+D^2 & 1 & 0 & D & D \\ D+D^2 & D^2 & D & D & D & 1 & 1 & 0 \end{pmatrix}.
\end{aligned}
$$

This matrix is actually minimal-basic (canonical), thus $G^c_{can}(D) = G^c(D)$. □

The same principle of enlarging generator matrices can easily be applied on rational generator matrices. The multiplication $\boldsymbol{v} = \boldsymbol{u}\boldsymbol{G}$ represents the convolution $\boldsymbol{v} = \boldsymbol{u} * \boldsymbol{g}$, where $\boldsymbol{g}$ is the impulse response of the encoder. The generator matrix $G(D)$ is the $\mathcal{D}$-transform of $\boldsymbol{g}$. For rational generator matrices we have an infinite (periodical) impulse response instead of a finite as for polynomial generator matrices. The enlarging procedure for rational generator matrices will be shown by the next example.

Example 4.4: Consider the generator matrix

$$
(4.30)\qquad G(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}.
$$

We wish to enlarge it by a factor two. The impulse response is

$$
(4.31)\qquad \boldsymbol{g} = (10)\ [(01)\ (01)\ (00)]^\infty,
$$

where $[\cdot]^\infty$ denotes infinitely many repetitions. The generator matrix $\boldsymbol{G}$ is therefore

$$\boldsymbol{G} = \begin{pmatrix} 11 & 01 & 01 & 00 & 01 & 01 & 00 & \cdots \\ & 11 & 01 & 01 & 00 & 01 & 01 & \cdots \\ & & 11 & 01 & 01 & 00 & 01 & \cdots \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{4.32}$$

Consider 2×4 sub-matrices instead of 1×2 sub-matrices, and write the impulse response,

$$\boldsymbol{g}_{[2]} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \left[\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \right]^\infty. \tag{4.33}$$

In the $\mathcal{D}$-transform representation this gives

$$G_{[2]}(D) = \mathcal{D}(\boldsymbol{g}_{[2]}) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} & 0 & \frac{1+D}{1+D+D^2} \\ 0 & \frac{D+D^2}{1+D+D^2} & 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}. \tag{4.34}$$

□

From the definition of abstract states it is clear that they are not affected by the enlargement. Especially, if $G(D)$ has a minimum number of abstract states, so does $\boldsymbol{G}$, and consequently also $G_{[e]}(D)$. That is, if $G(D)$ is minimal, so is $G_{[e]}(D)$. Thus, Theorem 4.4 still holds for unmatched rates.

Theorem 4.7: If $G^o(D)$ and $G^i(D)$ are two minimal generator matrices, then the cascaded generator matrix defined by the product of the enlarged variants of $G^o(D)$ and $G^i(D)$ is also minimal. □

Let $G(D)$ be a generator matrix with the set of abstract states $\mathcal{S}$. Define μ as the logarithm of the number of abstract states, $\mu = \log|\mathcal{S}|$, then a minimal realization of $G(D)$ must consist of μ memory elements. Since $\mathcal{S}$ is preserved while enlarging it is clear that a minimal realization of the enlarged generator matrix also consists of μ memory elements. Thus, the cascaded generator matrix can be realized with $\mu_o + \mu_i$ memory elements. Since we always have that $\mu \leqslant \nu$, for canonical generator matrices equality, and that canonical generator matrices have a minimum number of abstract states, we conclude that

$$\nu_{can} = \mu_{can} \leqslant \mu_o + \mu_i \leqslant \nu_o + \nu_i. \tag{4.35}$$

Thus, Theorem 4.2 holds for unmatched rates.

Theorem 4.8: Let $G^c_{can}(D)$ of overall constraint length ν_{can} be a canonical generator matrix equivalent to the cascaded generator matrix $G^c(D)$ defined as the product of the enlarged variants of the two generator matrices $G^o(D)$ and $G^i(D)$ of overall constraint lengths ν_o and ν_i, respectively, then

$$\nu_{can} \leqslant \nu_o + \nu_i. \tag{4.36}$$

□

So far we have mostly dealt with rational generator matrices. We can, however, say a little bit more about polynomial generator matrices. One interesting subject is the polynomial inverse of a basic generator matrix. This can be enlarged in the same way as described before.

Example 4.5: The outer generator matrix in Example 4.3,

$$G^o(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix}, \tag{4.37}$$

has the polynomial right inverse

$$(G^o(D))^{-1} = \begin{pmatrix} D \\ 1 + D \end{pmatrix}, \tag{4.38}$$

or as a semi-infinite generator matrix

$$(\boldsymbol{G}^o)^{-1} = \begin{pmatrix} 0 & 1 & & & & \\ 1 & 1 & & & & \\ & & 0 & 1 & & \\ & & 1 & 1 & & \\ & & & & 0 & 1 \\ & & & & 1 & 1 \\ & & & & & \ddots \end{pmatrix}, \tag{4.39}$$

which gives us the 3 times enlarged variant

$$(G^o_{[3]}(D))^{-1} = (G^o(D))^{-1}_{[3]} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ D & 0 & 0 \\ D & 0 & 1 \end{pmatrix}. \tag{4.40}$$

□

Since $G^o(D)(G^o(D))^{-1} = I_1 = 1$ we have that $\boldsymbol{G}^o(\boldsymbol{G}^o)^{-1} = I_\infty$ and $G^o_{[3]}(D)(G^o_{[3]}(D))^{-1} = I_3$, where I_n is the identity matrix of size $n \times n$. It is also clear that enlarging a polynomial matrix only give us another polynomial matrix, thus the enlarged variant of a basic generator matrix is basic. This means that Theorem 4.5 is still valid when $b_i \neq c_o$.

Theorem 4.9: If $G^o(D)$ and $G^i(D)$ are two basic generator matrices, then the cascaded generator matrix defined by the product of the two enlarged variants of $G^o(D)$ and $G^i(D)$ is also basic. □

In Example 4.4 the rational generator matrix $G(D)$ has overall constraint length $\nu = 2$, while the overall constraint length for the enlarged variant, $G_{[2]}(D)$ is $\nu_{[2]} = 4$. Thus, the overall constraint length is not preserved when enlarging a rational generator matrix, but, as the next lemma show, for polynomial generator matrices it is.

Lemma 4.10: If $G(D)$ is a polynomial generator matrix with overall constraint length ν and $G_{[e]}(D)$ is an enlarged variant with overall constraint length $\nu_{[e]}$. Then

$$\nu_{[e]} = \nu. \tag{4.41}$$

□

Proof: Assume that the $b \times c$ generator matrix $G(D)$ is enlarged e times. Consider an arbitrary row, say row i, of $G(D)$ with constraint length ν_i, and examine how this row contributes to $\nu_{[e]}$. Let this number be ν'_i. Thus,

$$\nu_{[e]} = \sum_{i=1}^{b} \nu'_i. \tag{4.42}$$

The ith row of $\boldsymbol{G}$, $i < b$ is

$$\boldsymbol{g}_i = g_0 g_1 \dots g_{e-1} g_e \dots g_{ke-1} g_{ke} \dots g_{\nu_i} 0 \dots , \tag{4.43}$$

where g_j is the ith row of the sub-matrix $\boldsymbol{G}_j$. This row contributes to $\nu_{[e]}$ with $\lfloor \nu_i/e \rfloor$. The next time ν_i contributes is on row $i+b$, which is a one step shifted version of $\boldsymbol{g}_i$

$$\boldsymbol{g}_{i+b} = 0 g_0 g_1 \dots g_{e-1} g_e \dots g_{ke-1} g_{ke} \dots g_{\nu_i} 0 \dots \tag{4.44}$$

thus it gives $\lfloor (\nu_i + 1)/e \rfloor$ in contribution. This continues e times, and we conclude that

$$\nu'_i = \sum_{l=0}^{e-1} \left\lfloor \frac{\nu_i + l}{e} \right\rfloor . \tag{4.45}$$

We need to show that this sum is equal to ν_i. First rewrite the constraint length as $\nu_i = ke + n$, where $0 \leqslant n < e$ and $k \in \mathbb{N}$, then

$$\left\lfloor \frac{\nu_i + l}{e} \right\rfloor = \left\lfloor k + \frac{l+n}{e} \right\rfloor = \begin{cases} k, & 0 \leqslant l \leqslant e-n-1 \\ k+1, & e-n \leqslant l \leqslant e-1 \end{cases} \tag{4.46}$$

The sum in (4.45) becomes

$$\begin{aligned} \nu_i' &= \sum_{l=0}^{e-1} \left\lfloor \frac{\nu_i + l}{e} \right\rfloor = \sum_{l=0}^{e-n-1} k + \sum_{l=e-n}^{e-1} k + 1 \\ &= (e-n)k + n(k+1) = ek + n = \nu_i. \end{aligned} \tag{4.47}$$

■

Corollary 4.11: If $G(D)$ is a minimal-basic generator matrix then its e times enlarged variant, $G_{[e]}(D)$, is also minimal-basic.

Proof: From Example 4.5 we know that if $G(D)$ is basic, so is $G_{[e]}(D)$. If $G_{[e]}(D)$ is not minimal-basic there exists an equivalent basic generator matrix, $G'_{[e]}(D)$ with overall constraint length $\nu'_{[e]} < \nu_{[e]} = \nu$. That is, we can construct an encoder for the code with fewer than ν delay-elements, which contradicts that $G(D)$ is minimal-basic. ■

Again, it is seen from Example 4.4 that this corollary only holds for minimal-basic generator matrices and not for all canonical generator matrices.

Parts of the proof of Lemma 4.10 can be used to estimate the memory for the cascaded generator matrix.

Theorem 4.12: Let $G^c_{mb}(D)$ be a minimal-basic generator matrix of memory m_{mb} that is equivalent to the generator matrix $G^c(D)$ defined as the product of the enlarged variants of the polynomial generator matrices $G^o(D)$ of memory m_o and $G^i(D)$ of memory m_i. Then,

$$m_{mb} \leqslant \left\lceil \frac{m_o}{b_d} \right\rceil + \left\lceil \frac{m_i}{c_d} \right\rceil, \tag{4.48}$$

where $b_d = b_i / \gcd(b_i, c_o)$ and $c_d = c_o / \gcd(b_i, c_o)$. □

Proof: Consider a polynomial rate $R = b/c$ generator matrix $G(D)$ and enlarge it e times into $G_{[e]}(D)$. Choose i to be a row in $G(D)$ with $\nu_i = m$.

Then the row $i + b(e-1)$ of $G_{[e]}(D)$ will have a constraint length equal to the memory, i.e., $\nu_{[e],i+b(e-1)} = m_{[e]}$. Thus,

$$m_{[e]} = \left\lfloor \frac{m+e-1}{e} \right\rfloor = \left\lfloor \frac{m}{e} + \frac{e-1}{e} \right\rfloor = \begin{cases} \frac{m}{e} & , e \mid m \\ \left\lfloor \frac{m}{e} \right\rfloor + 1 & , e \nmid m \end{cases} \tag{4.49}$$

which is equivalent to

$$m_{[e]} = \left\lceil \frac{m}{e} \right\rceil . \tag{4.50}$$

Enlarge the outer and inner generator matrices by the factors b_d and c_d, respectively, and apply Theorem 4.2 to complete the proof. ■

In Example 4.3 we had $m_o = 2$, $m_i = 2$, and $m_c = 2$. Inequality (4.48) in Theorem 4.12 states that

$$m_{mb} \leqslant \left\lceil \frac{m_o}{b_d} \right\rceil + \left\lceil \frac{m_i}{c_d} \right\rceil = \left\lceil \frac{2}{3} \right\rceil + \left\lceil \frac{2}{2} \right\rceil = 1 + 1 = 2. \tag{4.51}$$

Thus, the bound in Theorem 4.12 is tight.

In the remaining parts of this chapter it will be assumed that the constituent rates are matched. Generalizations to cascaded convolutional codes with unmatched rates are straight forward.

4.4. Non-equivalent Cascaded Encoders Obtained from Equivalent Constituent Encoders

In this section we shall analyze various examples of cascaded convolutional codes. Replace the outer generator matrix $G^o(D)$ with the equivalent generator matrix $G^{o\prime}(D) = T^o(D)G^o(D)$ and the inner generator $G^i(D)$ matrix with $G^{i\prime}(D) = T^i(D)G^i(D)$, where $T^o(D)$ and $T^i(D)$ are non-singular. Then the generator matrix for the cascaded encoder is

$$G^{c\prime}(D) = T^o(D)G^o(D)T^i(D)G^i(D). \tag{4.52}$$

We shall see that $G^{c\prime}(D)$ and $G^c(D) = G^o(D)G^i(D)$ are, in general, not equivalent.

If only the outer generator matrix $G^o(D)$ is replaced by an equivalent generator matrix, then the new cascaded generator matrix $G^{c\prime}(D)$ will be equivalent to the cascaded generator matrix $G^c(D)$ since

$$\begin{aligned} G^{c\prime}(D) &= G^{o\prime}(D)G^i(D) \\ &= T(D)G^o(D)G^i(D) = T(D)G^c(D). \end{aligned} \tag{4.53}$$

It is the code sequences from the outer encoder that serve as information sequences for the inner encoder. Therefore, the cascaded convolutional code is a proper (assuming $R_o < 1$) subset of the inner convolutional code, $\mathcal{C}^c \subset \mathcal{C}^i$. Replacing the inner encoder with an equivalent inner encoder changes the mapping from the inner information sequences to the inner code sequences and, consequently, also the subset of the inner convolutional code. In general, we will obtain a different cascaded convolutional code when we replace the inner encoder by an equivalent one. This fact is illustrated by the following two examples.

Example 4.6: Choose as outer and inner generator matrices

$$G^o(D) = \begin{pmatrix} 1+D & D \end{pmatrix} \tag{4.54}$$

and

$$G^i(D) = \begin{pmatrix} 1 & 1 & D \\ 1+D & D & 1 \end{pmatrix}, \tag{4.55}$$

respectively. The cascaded generator matrix is

$$\begin{aligned} G^c(D) &= G^o(D)G^i(D) \\ &= \begin{pmatrix} 1+D^2 & 1+D+D^2 & D^2 \end{pmatrix} \end{aligned} \tag{4.56}$$

which encodes a rate $R = 1/3$ convolutional code with $d_{\text{free}} = 6$.

Let

$$T_j(D) = \begin{pmatrix} 1 & 0 \\ 0 & D^j \end{pmatrix} \tag{4.57}$$

and replace the inner generator matrix with the equivalent generator matrix $T_j(D)G^i(D)$. Then for $j = 1$ the new cascaded generator matrix is

$$\begin{aligned} G_1^c(D) &= G^o(D)T_1(D)G^i(D) \\ &= \begin{pmatrix} 1+D+D^2+D^3 & 1+D+D^3 & D \end{pmatrix}, \end{aligned} \tag{4.58}$$

which gives $d_{\text{free}} = 8$. For $j = 3$ we have

$$G_3^c(D) = \begin{pmatrix} 1+D+D^4+D^5 & 1+D+D^5 & D+D^2+D^4 \end{pmatrix}, \tag{4.59}$$

which gives $d_{\text{free}} = 10$.

In Figure 4.2 the active row distances for the cascaded generator matrices are plotted. The active row distances for $G^c(D)$ and $G_1^c(D)$ have the same slope, but the free distance for $G_1^c(D)$ is superior to that of $G^c(D)$. The active row distance for $G_3^c(D)$ starts off better than the other two, i.e., it

has a larger free distance, but the others have steeper slope. The active row distance for $G_1^c(D)$ is above that of $G_3^c(D)$ after $j = 17$. Similarly, the active row distance for $G^c(D)$ is above that of $G_3^c(D)$ after $j = 22$. This will only have effect on very poor channels. For good and mediocre channels it is the early part of the active row distance that determines the error correcting capability. □

Clearly, as Example 4.6 shows, the cascaded convolutional code is changed when the inner encoder is replaced by an equivalent inner encoder. Furthermore, for a given generator matrix $G^i(D)$ there exists an infinite number of cascaded convolutional codes encoded by generator matrices of the type $G_j^c(D) = G^o(D)T_j(D)G^i(D)$. The generator matrices $G^o(D)$, $G^i(D)$, $G^c(D)$, $G_1^c(D)$, and $G_3^c(D)$ in Example 4.6 are, surprisingly enough, all minimal-basic, but $G_j^i(D) = T_j(D)G^i(D), j = 1, 3$, are neither basic nor minimal.

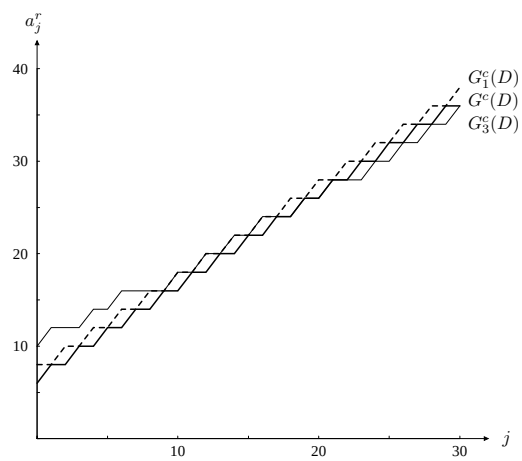

Figure 4.2: The active row distances for the generator matrices in Example 4.6.

Our next example deals with catastrophic generator matrices. Since $\mathcal{C}^c \subset \mathcal{C}^i$ for $R_o < 1$, catastrophicity of the inner generator matrix does not imply catastrophicity of the generator matrix for the cascade.

Example 4.7: Choose the outer and inner generator matrices as in Exam-

ple 4.6 and let

$$T_j(D) = \begin{pmatrix} 1+D^j & 0 \\ 0 & 1 \end{pmatrix}. \tag{4.60}$$

Then $G^{i\prime}(D) = T_j(D)G^i(D)$ is catastrophic, i.e., the infinite weight input

$$\boldsymbol{u}(D) = \begin{pmatrix} \frac{1}{1+D^j} & 0 \end{pmatrix} \tag{4.61}$$

generates the finite weight output

$$\boldsymbol{v}(D) = \begin{pmatrix} 1 & 1 & D \end{pmatrix}. \tag{4.62}$$

However, the cascaded generator matrix, $G_j^c(D) = G^o(D)T_j(D)G^i(D)$ is non-catastrophic for $j \geqslant 1$. In Figure 4.3 the active row distances for $G^c(D) = G^o(D)G^i(D)$ and $G_6^c(D)$ are plotted. We see that for $j = 6$ we even get a larger free distance than in Example 4.6, viz., $d_{\text{free}} = 11$. □

Remark: Although we do not discuss decoding in this chapter we remark that if the decoding is performed in two steps, i.e., first we decode the inner convolutional code and then the outer convolutional code, the catastrophicity of the inner generator matrix might still be harmful.

Can the inner generator matrix be chosen such that we obtain the same cascaded convolutional code, i.e., such that $G^{c\prime}(D)$ is equivalent to $G^c(D)$? Indeed it can, which is illustrated by the following simple example.

Example 4.8: Let the outer generator matrix be

$$G^o = \begin{pmatrix} 1+D+D^2 & 1+D^2 \end{pmatrix} \tag{4.63}$$

and let the inner generator matrix $G^i(D)$ be of rate $R_i = 2/c_i$.

If $G^{i\prime}(D) = T_1(D)G^i(D)$, where

$$T_1(D) = \begin{pmatrix} \frac{1}{1+D+D^2} & 0 \\ 0 & \frac{1}{1+D+D^2} \end{pmatrix}, \tag{4.64}$$

then the cascaded generator matrix is

$$\begin{aligned} G_1^{c\prime}(D) &= G^o(D)T_1(D)G^i(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix} G^i(D) \\ &= \frac{1}{1+D+D^2} G^o(D)G^i(D) = \frac{1}{1+D+D^2} G^c(D), \end{aligned} \tag{4.65}$$

Hence, $G_1^{c\prime}(D)$ and $G^c(D)$ are equivalent generator matrices.

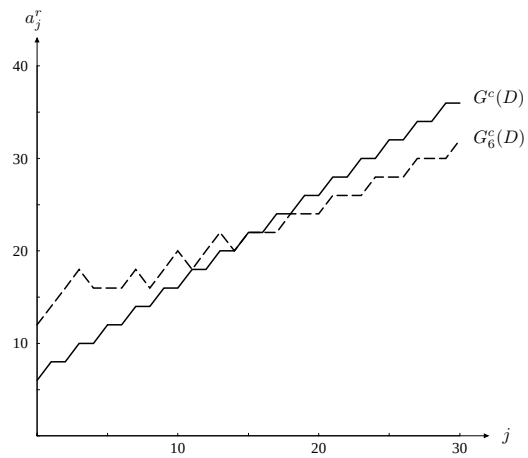

Figure 4.3: The active row distances for the generator matrices in Example 4.7.

On the other hand, if we choose

$$T_2(D) = \begin{pmatrix} \frac{1}{1+D+D^2} & 0 \\ 0 & \frac{1}{1+D^2} \end{pmatrix}, \tag{4.66}$$

then we obtain

$$G_2^{c\prime}(D) = G^o(D)T_2(D)G^i(D) = \begin{pmatrix} 1 & 1 \end{pmatrix} G^i(D), \tag{4.67}$$

which clearly is not equivalent to $G^c(D)$. □

As a straight-forward generalization of Example 4.8 it follows that if $G^{c\prime}(D)$ and $G^c(D)$ are equivalent generator matrices, then for some invertible $b_o \times b_o$ matrix $S(D)$

$$\begin{aligned} G^{c\prime}(D) &= G^o(D)T(D)G^i(D) \\ &= S(D)G^c(D) = S(D)G^o(D)G^i(D), \end{aligned} \tag{4.68}$$

or, equivalently,

$$G^o(D)T(D) = S(D)G^o(D). \tag{4.69}$$

4.5. Systematic Cascaded Encoders

Example 4.6 shows that we can change the cascaded convolutional code without violating the restriction on the overall constraint length $\nu_c' \leqslant \nu_o + \nu_i$. Below we use systematic generator matrices to obtain a general construction that obeys this restriction. Furthermore, for decoding it might be desirable to let both the inner and outer generator matrices be systematic rational generator matrices.

From a generator matrix we can easily obtain an equivalent systematic generator matrix. Hence, we can easily find invertible $b_o \times b_o$ and $b_i \times b_i$ matrices T^o and T^i, respectively, such that

$$G^c_{\text{sys}}(D) = G^o_{\text{sys}}(D)G^i_{\text{sys}}(D) = T^o(D)G^o(D)T^i(D)G^i(D), \tag{4.70}$$

where $G^c_{\text{sys}}(D)$, $G^o_{\text{sys}}(D)$, and $G^i_{\text{sys}}(D)$ all are systematic generator matrices. In general, $G^c_{\text{sys}}(D)$ is not equivalent to $G^c(D) = G^o(D)G^i(D)$.

Forney [13] showed that systematic generator matrices are always minimal. Thus, it follows that the systematic cascaded generator matrix

$$G^c_{\text{sys}}(D) = G^o_{\text{sys}}(D)G^i_{\text{sys}}(D), \tag{4.71}$$

where $G^o_{\text{sys}}(D)$ and $G^i_{\text{sys}}(D)$ are equivalent to $G^o(D)$ and $G^i(D)$, is a minimal generator matrix for $\mathcal{C}^c_{\text{sys}}$. Furthermore, if $G^o(D)$ and $G^i(D)$ are minimal, then a minimal realization of $G^c_{\text{sys}}(D)$ has at most the same number of delay elements as the cascaded encoder $G^c(D)$ has when realized as a cascade of the two minimal encoders $G^o(D)$ and $G^i(D)$.

Example 4.9: Choose as outer generator matrix

$$G^o(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix} \tag{4.72}$$

and as inner generator matrix

$$G^i(D) = \begin{pmatrix} D^2 & 1 + D & 1 + D + D^2 \\ 1 + D + D^2 & 1 + D + D^2 & 1 \end{pmatrix}, \tag{4.73}$$

which both are minimal-basic. The cascaded generator matrix yields

$$G^c(D) = G^o(D)G^i(D) = \begin{pmatrix} 1 + D + D^2 & D + D^4 & D^4 \end{pmatrix}, \tag{4.74}$$

which is minimal-basic and encodes a rate $R = 1/3$ convolutional code with $d_{\text{free}} = 6$.

With the invertible matrices

$$T^o(D) = \left(\tfrac{1}{1+D+D^2}\right) \tag{4.75}$$

and

$$T^i(D) = \frac{1}{1+D^2+D^4}\begin{pmatrix} 1+D+D^2 & 1+D \\ 1+D+D^2 & D^2 \end{pmatrix} \tag{4.76}$$

we can find the systematic outer and inner generator matrices

$$G^o_{\text{sys}}(D) = T^o(D)G^o(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix} \tag{4.77}$$

and

$$G^i_{\text{sys}}(D) = T^i(D)G^i(D) = \begin{pmatrix} 1 & 0 & \frac{D+D^2+D^4}{1+D^2+D^4} \\ 0 & 1 & \frac{1+D^4}{1+D^2+D^4} \end{pmatrix}, \tag{4.78}$$

which are equivalent to $G^o(D)$ and $G^i(D)$, respectively. The generator matrix for the cascade of the systematic generator matrices is

$$\begin{aligned} G^c_{\text{sys}}(D) &= G^o_{\text{sys}}(D)G^i_{\text{sys}}(D) \\ &= \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} & \frac{1+D+D^2+D^4+D^5}{1+D+D^3+D^5+D^6} \end{pmatrix}, \end{aligned} \tag{4.79}$$

which encodes a rate $R = 1/3$ convolutional code with $d_{\text{free}} = 12$. Since $G^o(D)$ and $G^i(D)$ both are minimal-basic (canonical) we know that their minimal encoders consists of 2 and 4 delay elements, respectively. Therefore a minimal encoder for the convolutional code $\mathcal{C}^c_{\text{sys}}$ can be realized by at most 6 delay elements, i.e., the systematic generator matrix $G^c_{\text{sys}}(D)$ can be realized by at most 6 delay elements. As a matter of fact, it is easily seen that a realization of $G^c_{\text{sys}}(D)$ requires 6 delay elements.

In Figure 4.4 we have plotted the active row distances for the generator matrices $G^c(D)$ and $G^c_{\text{sys}}(D)$. The active row distance for $G^c_{\text{sys}}(D)$ is superior to that of $G^c(D)$ up to $j = 25$, where they cross. □

Remark: The cascaded generator matrix $G^c(D)$ can be systematic even if neither the outer nor the inner generator matrices are systematic. If, for example, the inner generator matrix is chosen as

$$G^i(D) = \begin{pmatrix} (G^o(D))^{-1} & A(D) \end{pmatrix}, \tag{4.80}$$

where $(G^o(D))^{-1}$ is the right inverse of the outer generator matrix $G^o(D)$, then the cascaded generator matrix is systematic, i.e.,

$$G^c(D) = G^o(D)G^i(D) = \begin{pmatrix} I & G^o(D)A(D) \end{pmatrix}. \tag{4.81}$$

It is, however, preferable from a decoding point of view that both the outer and the inner generator matrices are systematic.

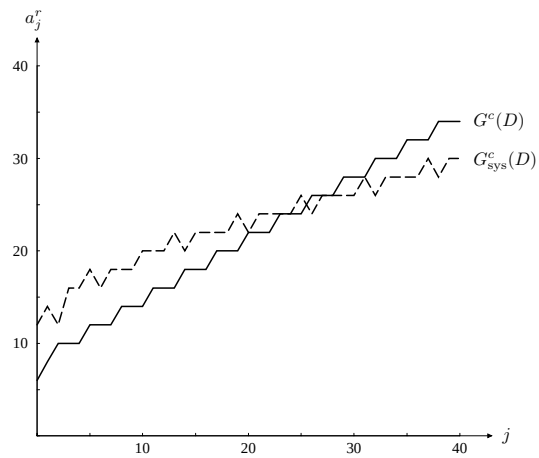

Figure 4.4: The active row distances for the generator matrices in Example 4.9.

4.6. Lower Bounds on the Active Distances

The active distances are in general difficult to calculate. In this section we will derive lower bounds on the active distances for polynomial time-invariant cascaded convolutional generator matrices. We will use the lower bounds in (3.48).

Assume that both the outer and the inner generator matrices are minimal-basic and that the corresponding encoders are realized on controller canonical form. The cascaded encoder is considered to be in the zero state if both the inner and outer encoders are in the zero state.

A sub-sequence of the inner information sequence consisting of all-zero b_i-tuples, preceded by a non-zero b_i-tuple and followed by a non-zero b_i-tuple, such that it drives the inner encoder to the zero state is called a *burst of zeros*. The length—the number of b_i-tuples—of a burst of zeros depends on the encoder state but, clearly, it is lower bounded by

$$l_{\text{burst}} \geqslant \nu^i_{\min}, \tag{4.82}$$

where $\nu^i_{\min}$ is the minimum constraint length of the inner generator matrix. If the outer encoder must not get to the zero state the length of a burst of

zeros can also be upper bounded by

$$l_{\text{burst}} \leqslant \left\lfloor \frac{j_0^{so} c_o + 2(c_o - 1)}{b_i} \right\rfloor, \tag{4.83}$$

where j_0^{so} is the longest sequence of all-zero c_o-tuples generated by the outer encoder when it does not have consecutive zero states. Denote this maximum length by l_b.

From Theorem 3.5 we can get some knowledge about the output of the outer encoder. Let w_b be the lower bound on the weight between the beginning of the sequence and the first burst of zeros occupying l_b symbols, i.e., the least $w_{l_b}^c$ that satisfies (3.45a). Similarly, let w_e be the minimum weight from the last burst of zeros to the end of the sequence, i.e., the least $w_{l_b}^{rc}$ that satisfies (3.45b), and let w_s be the minimum weight between two bursts of l_b zeros, i.e., the least w_{l_b,l_b}^s that satisfies (3.45c).

To simplify the notation, let j_i denote the degree of the inner information sequence when the degree of the outer information sequence is j, where we assume that the information sequences are written as Laurent series. Since the outer generator matrix is minimal-basic it has the predictable degree property and, hence, j_i is lower bounded by

$$j_i \geqslant (j + \nu_{\min}^o)\frac{c_o}{b_i}. \tag{4.84}$$

When the cascaded encoder is not in the zero state we might still have either the outer or the inner encoder in the zero state. If the inner encoder is not in the zero state, then a rough estimate of the minimum output weight from the inner encoder is $a_{j_i}^{ri}$ both when the outer encoder is allowed to reach the zero state and when it is not. On the other hand, if the inner encoder is in the zero state the outer encoder must not be in the zero state. Hence, there are two different cases to consider:

(i) The inner encoder is not in the zero state.

(ii) The inner encoder is in the zero state, but the outer is not.

From (4.82) and (4.83) we know the range for the length of a zero sequence that will drive the inner encoder to the zero state. Since the output from the encoder is zero, while in zero state, the burst should be chosen as long as possible to get an estimate of the minimum output weight. Define φ_b as the reduction in Hamming weight when a burst of zeros is inserted. If the burst starts after k information symbols this reduction will be, see Figure 4.5,

$$\begin{aligned} \varphi_b &= f^{ri}(j_i) - \big(f^{ri}(k-1) + f^{ri}(j_i - k - l_b)\big) \\ &= \alpha^i(l_b + 1) - \beta^{ri}, \end{aligned} \tag{4.85}$$

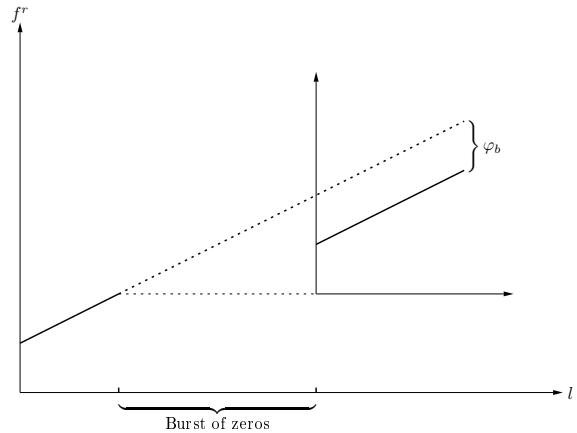

Figure 4.5: One burst of zeros splitting the sequence into two sub-sequences.

where $f^{ri}(j)$ is the affine function, defined in (3.48), that lower bounds the active row distance for the inner generator matrix.

If φ_b is positive, then we have the same reduction for a second burst of zeros. Finally, assume that the sequence has been split by N_r bursts of zeros. Then,

- the length before the first burst of zeros is at least $\lceil w_b/b_i \rceil$.
- the length after the last burst of zeros is at least $\lceil w_e/b_i \rceil$.
- the length between two (consecutive) bursts of zeros is at least $\lceil w_s/b_i \rceil$.

Consequently, the largest number of bursts of zeros is the largest N_r such that

$$N_r l_b + (N_r - 1)\left\lceil \frac{w_s}{b_i} \right\rceil + \left\lceil \frac{w_b}{b_i} \right\rceil + \left\lceil \frac{w_e}{b_i} \right\rceil \leqslant j_i + 1. \tag{4.86}$$

Hence, the active row distance for the cascaded encoder can be lower bounded by

$$\begin{aligned} a_j^{rc} \geqslant \min\Bigg\{ & f^{ri}(j_i), \\ & f^{ri}\left(\left\lceil \frac{w_b}{b_i} \right\rceil - 1\right) + (N_r - 1) f^{ri}\left(\left\lceil \frac{w_s}{b_i} \right\rceil - 1\right) + f^{ri}\left(\left\lceil \frac{w_e}{b_i} \right\rceil - 1\right) \Bigg\} \end{aligned} \tag{4.87}$$

and the free distance by

$$d^c_{\text{free}} \geqslant \min_j \{a^{rc}_j\}. \tag{4.88}$$

The lower bound on the active burst distance can be derived analogously. The largest number of bursts of zeros is the largest number N_b such that

$$N_b l_b + (N_b - 1)\left\lceil \frac{w_s}{b_i} \right\rceil + \left\lceil \frac{w_b}{b_i} \right\rceil + \left\lceil \frac{w_e}{b_i} \right\rceil \leqslant j\frac{c_o}{b_i} + 1. \tag{4.89}$$

The active burst distance for the cascaded encoder can be lower bounded by

$$\begin{aligned} a^{bc}_j \geqslant \min\Bigg\{ & f^{bi}(j\frac{c_o}{b_i}), \\ & f^{ri}\Big(\left\lceil \frac{w_b}{b_i} \right\rceil - 1\Big) + (N_r - 1) f^{ri}\Big(\left\lceil \frac{w_s}{b_i} \right\rceil - 1\Big) + f^{bi}\Big(\left\lceil \frac{w_e}{b_i} \right\rceil - 1\Big)\Bigg\}. \end{aligned} \tag{4.90}$$

When calculating the active column distance and the active reverse column distance we consider the same cases, i.e., when the inner encoder does not have consecutive zero states and when it is fed with as many bursts of zeros as possible. For the active column distance the minimum number of bursts of zeros is the largest N_c such that

$$N_c l_b + N_c \left\lceil \frac{w_s}{b_i} \right\rceil + \left\lceil \frac{w_b}{b_i} \right\rceil \leqslant j\frac{c_o}{b_i} + 1. \tag{4.91}$$

Similarly, when lower bounding the active reverse column distance the maximum number of bursts of zeros is given by the largest N_{rc} such that

$$N_{rc} l_b + N_{rc} \left\lceil \frac{w_s}{b_i} \right\rceil + \left\lceil \frac{w_e}{b_i} \right\rceil \leqslant j\frac{c_o}{b_i} + 1. \tag{4.92}$$

Then, the bounds on the active column distance and the active reverse column distance for the cascaded code can be calculated as

$$\begin{aligned} a^{cc}_j \geqslant \min\Bigg\{ & f^{ci}\left(j\frac{c_o}{b_i}\right), \\ & f^{ri}\Big(\left\lceil \frac{w_b}{b_i} \right\rceil - 1\Big) + (N_c - 1) f^{ri}\Big(\left\lceil \frac{w_s}{b_i} \right\rceil - 1\Big) + f^{ci}\Big(\left\lceil \frac{w_s}{b_i} \right\rceil - 1\Big)\Bigg\} \end{aligned} \tag{4.93}$$

and

$$\begin{aligned} a^{rcc}_j \geqslant \min\Bigg\{ & f^{rci}\left(j\frac{c_o}{b_i}\right), \\ & f^{rci}\Big(\left\lceil \frac{w_s}{b_i} \right\rceil - 1\Big) + (N_{rc} - 1) f^{ri}\Big(\left\lceil \frac{w_s}{b_i} \right\rceil - 1\Big) + f^{ri}\Big(\left\lceil \frac{w_e}{b_i} \right\rceil - 1\Big)\Bigg\}, \end{aligned} \tag{4.94}$$

respectively.

The first case to consider when deriving the active segment distance is, as before, when there are no bursts of zeros. For the second case one burst of zeros splits the information sequence into two sub-sequences. Each of these sub-sequences can be split further. Then, the maximum number of bursts of zeros is the largest N_s such that

$$N_s l_b + N_s \left\lceil \frac{w_s}{b_i} \right\rceil \leqslant j\frac{c_o}{b_i} + 1. \tag{4.95}$$

Thus, to calculate the active segment distance there are three different cases to consider, which results in

$$\begin{aligned} a_j^{sc} \geqslant \min\Bigg\{ & f^{ci}\left(j\frac{c_o}{b_i}\right), \\ & f^{rci}\Big(\left\lceil \frac{w_b}{b_i} \right\rceil - 1\Big) + f^{ci}\Big(j\frac{c_o}{b_i} - \left\lceil \frac{w_e}{b_i} \right\rceil - l_b\Big), \\ & f^{rci}\Big(\left\lceil \frac{w_b}{b_i} \right\rceil - 1\Big) + (N_s - 1) f^{ri}\Big(\left\lceil \frac{w_s}{b_i} \right\rceil - 1\Big) + f^{ci}\Big(\left\lceil \frac{w_e}{b_i} \right\rceil - 1\Big)\Bigg\}. \end{aligned} \tag{4.96}$$

4.7. Time-varying Cascaded Convolutional Codes

Consider the ensemble $\mathcal{EC}(b_c, c_c, m_c, T_c)$ of periodically time-varying, cascaded convolutional codes constructed in the following way:

Choose as an outer convolutional code a binary, periodically time-varying, rate $R_o = b_o/c_o$ convolutional code with period T, encoded by a polynomial generator matrix of memory m_o and whose starting point of its active segment distance is

$$j_0^{so} = \frac{R_o}{1 - R_o} m_o + O(\log m_o). \tag{4.97}$$

The existence of such a convolutional code was shown in Chapter 3.

The ensemble of inner convolutional codes is the ensemble of binary, periodically time-varying convolutional codes with period T, rate $R_i = b_i/c_i$, where $b_i = c_o$, encoded by time-varying polynomial generator matrices of memory $m_i \geqslant j_0^{so}$. Clearly, we have $b_c = b_o$, $c_c = c_i$, $m_c = m_o + m_i$, and $T_c = T$.

To derive bounds on the active distances for the ensemble of periodically time-varying cascaded convolutional codes we need a counterpart to Theorem 3.6.

Theorem 4.13: Consider a periodically time-varying, rate $R_c = b_c/c_c$, cascaded convolutional code encoded by a convolutional encoder of memory m_c.

(*i*) Let the information sequences be restricted to the set $\mathcal{U}^r_{[t-m_c,t+j+m_c]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j+m_c]}$ are mutually independent and equiprobable over the ensemble $\mathcal{EC}(b_c, c_c, m_c, T_c)$ for all j, $0 \leqslant j < T_c$.

(*ii*) Let the information sequences be restricted to the set $\mathcal{U}^c_{[t-m_c,t+j]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{EC}(b_c, c_c, m_c, T_c)$ for all j, $0 \leqslant j < T_c$.

(*iii*) Let the information sequences be restricted to the set $\mathcal{U}^{rc}_{[t-m_c,t+j]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{EC}(b_c, c_c, m_c, T_c)$ for all j, $0 \leqslant j < T_c$.

(*iv*) Let the information sequences be restricted to the set $\mathcal{U}^s_{[t-m_c,t+j]}$. Then the code symbols in the segment $\boldsymbol{v}_{[t,t+j]}$ are mutually independent and equiprobable over the ensemble $\mathcal{EC}(b_c, c_c, m_c, T_c)$ for all j, $0 \leqslant j < T_c$. □

Proof: Analogously to the proof of Theorem 3.6. ■

If we let

$$\ell = \frac{j+1}{m_c}, \tag{4.98}$$

then, asymptotically—for large memories m_c—we obtain the following counterpart to Theorem 3.19 for the ensemble of periodically time-varying convolutional codes $\mathcal{E}(b, c, m, T)$.

Theorem 4.14: In the ensemble $\mathcal{EC}(b_c, c_c, m_c, T_c)$ there exists

(*i*) a cascaded convolutional code whose normalized active row distance asymptotically satisfies

$$\delta^{rc}_\ell \triangleq \frac{a^{rc}_j}{m_c c_c} \geqslant h^{-1}\Big(1 - \frac{\ell}{\ell+1} R_c\Big)(\ell+1) + O\left(\frac{\log m_c}{m_c}\right) \tag{4.99a}$$

for $\ell \geqslant 0$.

(*ii*) a cascaded convolutional code whose normalized active burst distance asymptotically satisfies

$$\delta^{bc}_\ell \triangleq \frac{a^{bc}_j}{m_c c_c} \geqslant h^{-1}\Big(1 - \frac{\ell-1}{\ell} R_c\Big)\ell + O\left(\frac{\log m_c}{m_c}\right) \tag{4.99b}$$

for $\ell \geqslant 1$.

(*iii*) a cascaded convolutional code whose normalized active column (reverse column) distance asymptotically satisfies

$$\left.\begin{aligned} \delta_\ell^{cc} &\triangleq \frac{a_j^{cc}}{m_c c_c} \\ \delta_\ell^{rcc} &\triangleq \frac{a_j^{rcc}}{m_c c_c} \end{aligned}\right\} \geqslant h^{-1}(1-R_c)\ell + O\left(\frac{\log m_c}{m_c}\right) \tag{4.99c}$$

for $\ell \geqslant \ell_0 = O\left(\frac{\log m_c}{m_c}\right)$.

(*iv*) a cascaded convolutional code whose normalized active segment distance asymptotically satisfies

$$\delta_\ell^{sc} \triangleq \frac{a_j^{sc}}{m_c c_c} \geqslant h^{-1}\Big(1 - \frac{\ell+1}{\ell} R_c\Big)\ell + O\left(\frac{\log m_c}{m_c}\right) \tag{4.99d}$$

for $\ell \geqslant \ell_s^o = \frac{R_c}{1-R_c} + O\left(\frac{\log m_c}{m_c}\right)$. □

The theorem shows that the performance for all active distances for the ensemble of time-varying cascaded convolutional codes is the same as for the ensemble of time-varying convolutional codes although the former ensemble is essentially smaller. Thus, we have the same typical behavior for the bounds as in Figure 3.9

The free distance for a convolutional code is obtained as the minimum weight of the non-zero codewords. The only restriction on the input sequence is that it should be non-zero. Since the inputs to the inner encoder are restricted to be codewords of the outer encoder we will not obtain a useful estimate of d_{free}^c from the free distance of the inner code, d_{free}^i.

It is somewhat surprising that, given only a restriction on the memory of the inner code, there exists a convolutional code obtained as a cascade with a free distance satisfying the Costello bound [9].

Theorem 4.15: In the ensemble $\mathcal{EC}(b_c, c_c, m_c, T_c)$ there exists a cascaded convolutional code whose free distance satisfies the inequality

$$\begin{aligned} d_{\text{free}}^c &\geqslant \frac{m_c c_c R_c}{-\log(2^{1-R_c}-1)} + O(\log m_c) \\ &\geqslant \frac{m_o c_o R_o}{-\log(2^{1-R_o}-1)} + \frac{m_i c_i R_i}{-\log(2^{1-R_i}-1)} \\ &\quad + O(\log m_o) + O(\log m_i). \end{aligned} \tag{4.100}$$

□

Proof: By minimizing (4.99a), we obtain

$$
\begin{aligned}
d_{\text{free}}^{c} &\geqslant \frac{m_c b_o}{-\log(2^{1-R_c}-1)} + O(\log m_c) \\
&= \frac{m_c c_i R_c}{-\log(2^{1-R_c}-1)} + O(\log m_c) \\
&= \frac{m_o b_o}{-\log(2^{1-R_c}-1)} + \frac{m_i c_i R_c}{-\log(2^{1-R_c}-1)} + O(\log m_c) \\
&\geqslant \frac{m_o c_o R_o}{-\log(2^{1-R_o}-1)} + \frac{m_i c_i R_i}{-\log(2^{1-R_i}-1)} + O(\log m_c),
\end{aligned}
\tag{4.101}
$$

where we in the last inequality have used that

$$
-\log(2^{1-R}-1) \tag{4.102}
$$

is an increasing function and

$$
\frac{R}{-\log(2^{1-R}-1)} \tag{4.103}
$$

is a decreasing function of R. ■

Assume that for the cascaded encoder in Theorem 4.15 we have $\nu_{\min,o} = m_o$ and $\nu_{\min,i} = m_i$ and that $m_i = \frac{R_o}{1-R_o} m_o$ holds. Then the total number of states in the outer and inner encoders are

$$
\begin{aligned}
2^{m_o b_o} + 2^{m_i b_i} &= 2^{m_o b_o}(1 + 2^{m_i b_i - m_o b_o}) \\
&= 2^{(m_o+m_i) b_o}(1 + 2^{-m_i b_o}),
\end{aligned}
\tag{4.104}
$$

which is essentially equal to the total number of states of a generator matrix $G^c(D)$ with $\nu_{\min,c} = m_c$. The second equality follows from the equalities $m_i = \frac{R_o}{1-R_o} m_o$ and $b_i = c_o$. Thus, given the restriction on the ratio m_i/m_o, from the Costello lower bound point of view we do not lose in free distance by splitting a given amount of convolutional encoder memory into two cascaded convolutional encoders!

5

Woven Convolutional Codes

In the previous chapter a cascade of two convolutional encoders were investigated. The error performance of such construction is not very promising. On the receiver side in a communication system the inner decoder will group the errors in bursts, which the outer decoder cannot handle. The bursts of errors from the outer decoders must in some way be split. The traditional way to accomplish this is to feed the sequence into a buffer where the bits are permuted. We will consider a further development of cascaded convolutional codes. Instead of letting one inner encoder follow one single outer encoder, we put a set of parallel encoders in the place of the outer and inner encoder. In the constructions convolutional codes are *woven* together in a manner that resembles the structure of a fabric. We call these constructions *woven convolutional codes* [25] and distinguish between *outer* and *inner warp*.

In Section 5.1 the encoders for three types of woven convolutional codes will be described, woven convolutional codes with outer warp, with inner warp, and the twill. In Sections 5.2 and 5.3 structural properties of the generator matrices for the warp will be investigated, and in Section 5.4 the same is done for the woven convolutional codes. In Section 5.5 the free distance and the active distances for the constructions are lower bounded.

5.1. Introduction to Woven Convolutional Codes

We will introduce three related woven constructions, woven convolutional codes with outer warp, woven convolutional codes with inner warp, and the twill. The two first constructions are degenerated special cases of the latter. For simplicity we describe the two special cases of the twill first.

In the first construction we consider l_o rate $R_o = b_o/c_o$, outer, binary, convolutional encoders in parallel. The information sequence is divided into sub-blocks of $l_o b_o$ information symbols each. These sub-blocks are fed in parallel into the l_o parallel outer encoders. The c_o code sequences from each outer encoder are serialized and written row-wise into a buffer consisting of l_o rows. These l_o code sequences constitute the *warp*. The binary code symbols are read column-wise and used as inputs to one rate $R_i = b_i/c_i$, inner, binary, convolutional encoder—the *weft*[1], see Figure 5.1 This *woven convolutional code with outer warp* has overall rate $R_{ow} = b_{ow}/c_{ow}$, where $b_{ow} = l_o b_o$ and $c_{ow} = l_o c_o c_i/b_i = l_o c_o/R_i$. Hence, we have

$$R_{ow} = \frac{b_{ow}}{c_{ow}} = \frac{l_o b_o}{l_o c_o \frac{c_i}{b_i}} = \frac{b_o b_i}{c_o c_i} = R_o R_i. \tag{5.1}$$

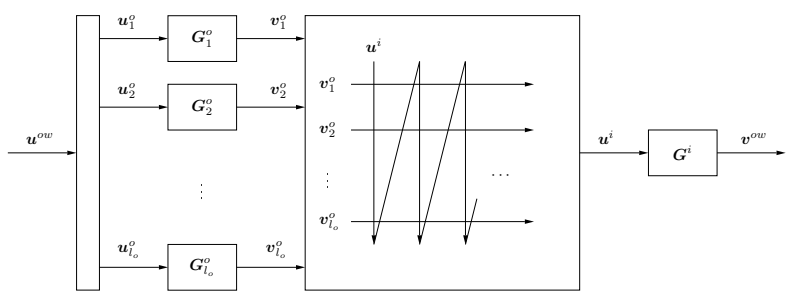

Figure 5.1: Woven convolutional encoder with outer warp.

The second construction is reversed compared with the previous one. Use one outer convolutional encoder as the weft. The code sequence is written column-wise into a buffer with l_i rows and the binary symbols in the rows are regarded as information symbols for the l_i parallel inner convolutional encoders—the warp of the construction, see Figure 5.2.

[1] The words warp and weft are borrowed from the art of textile handicraft. In a loom the parallel threads that build the foundation of the cloth is named *warp*, while the inserted thread is called *weft* (also *filling* or *woof*).

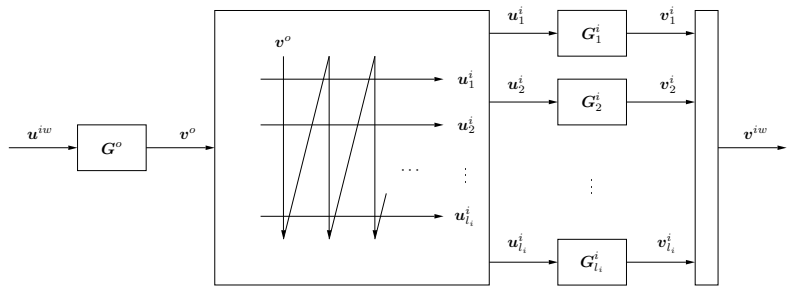

Figure 5.2: Woven convolutional encoder with inner warp.

If we consider $c_{iw} = l_i c_i$ as an output tuple, then the corresponding input tuple is $b_{iw} = l_i b_i b_o / c_o$, and, hence, the overall rate for the *woven convolutional code with inner warp* is

$$R_{iw} = \frac{b_{iw}}{c_{iw}} = \frac{l_i b_i \frac{b_o}{c_o}}{l_i c_i} = \frac{b_o b_i}{c_o c_i} = R_o R_i. \tag{5.2}$$

Both constructions can be considered as special cases of the *twill* which has both outer and inner warp. The outer warp consists of l_o rate $R_o = b_o/c_o$ outer encoders and the inner warp of l_i rate $R_i = b_i/c_i$ encoders. The information sequence is divided into sub-blocks of $b_{tw} = l_o l_i b_o$ binary information symbols each. These sub-blocks are fed into the l_o parallel encoders. After encoding and serializing, these l_o code sequences constitute the outer warp. Then l_i binary code symbols from each of the outer warp sequence are read column-wise and used as inputs to the l_i inner encoders, see Figure 5.3. Hence, each sub-block of $b_{tw} = l_o l_i b_o$ binary information symbols is encoded by the twill encoder into a sub-block of $c_{tw} = l_o l_i c_o c_i / b_i$ binary code symbols and we have

$$R_{tw} = \frac{b_{tw}}{c_{tw}} = \frac{l_o l_i b_o}{l_o l_i c_o \frac{c_i}{b_i}} = \frac{b_o b_i}{c_o c_i} = R_o R_i. \tag{5.3}$$

Clearly, the woven convolutional code with outer (inner) warp is a twill with $l_i = 1$ ($l_o = 1$). When $l_o = l_i = 1$ we have a cascaded convolutional code.

The next example implies that l_o and l_i should be chosen relatively prime, i.e.,

$$\gcd(l_o, l_i) = 1. \tag{5.4}$$

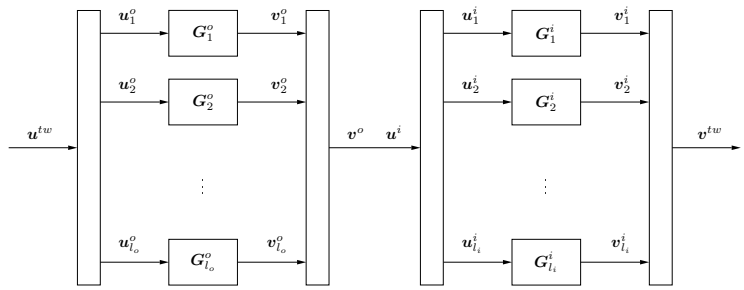

Figure 5.3: Encoder for the twill.

Example 5.1: First, consider a twill with $l_o = 10$ and $l_i = 3$. Then the buffer between the outer and inner warps can be viewed as in Figure 5.4. The ten arrows on the left in the figure represent the outer encoders, and the output from each of them is written row-wise into the buffer. The buffer is read column-wise, and the first bit goes to the first inner encoder, the second bit to the second encoder, the third bit to the third encoder, the fourth bit, to the first encoder, and so forth. The grey shaded squares in the figure represent the information sequence for the first inner encoder. We see that the bits of the information sequences for each of the inner encoders are taken from all the outer code sequences.

Next, add one inner encoder to get $l_i = 4$. In Figure 5.5 the buffer structure is redrawn. We still write row-wise into the buffer and read column-wise. Thus, bits number 1, 5, 9, 13, ... serve as input for the first inner encoder. These bits are marked with dark grey shaded squares in the figure. Likewise, the light grey shaded squares represent the bits that serve as input for the third encoder.

We see that the sequences from the odd outer encoders are distributed over the odd inner encoders, while the sequences from the even outer encoders are distributed over the even inner encoders. This scheme is actually split into two completely separated parallel twills with five outer and two inner encoders each. To avoid such splits the number of outer and inner encoders should be chosen relatively prime. □

The name twill comes from the pattern of the shaded squares in the buffer of Figure 5.4. That same pattern is used when weaving the fabric twill, e.g., the fabric of jeans.

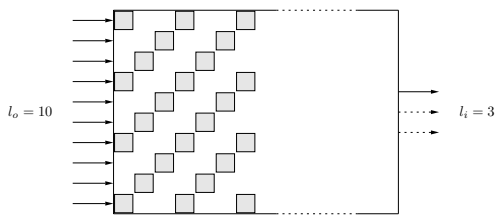

Figure 5.4: The buffer structure of a twill with ten outer and three inner encoders.

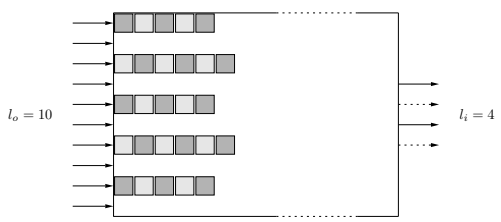

Figure 5.5: The buffer structure of a twill with ten outer and four inner encoders.

5.2. The Generator Matrix of the Warp

The warp can be considered as the main building block of a twill. Assume first that all constituent encoders are identical. In Figure 5.6 we show a warp with l_w constituent rate $R = b/c$ encoders, each with generator matrix $G(D)$. The rate of the warp is the same as the rate of its constituent encoders.

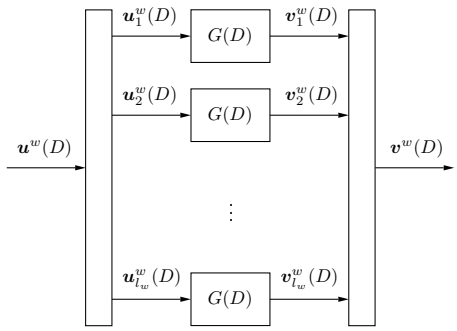

Figure 5.6: A warp with l_w identical parallel encoders.

The information sequence of the warp is divided into sub-blocks of bl_w information symbols. Then we have

$$\boldsymbol{u}^w(D) = \Big(\boldsymbol{u}^{(1)}(D) \dots \boldsymbol{u}^{(b)}(D)\Big), \tag{5.5}$$

where

$$\boldsymbol{u}^{(i)}(D) = \Big(u_1^{(i)}(D) \dots u_{l_w}^{(i)}(D)\Big). \tag{5.6}$$

The information sequence of the lth constituent encoder is

$$\boldsymbol{u}_l^w(D) = \Big(u_l^{(1)}(D) \dots u_l^{(b)}(D)\Big) \tag{5.7}$$

and the corresponding code sequence

$$\begin{aligned} \boldsymbol{v}_l^w(D) &= \boldsymbol{u}_l^w(D)G(D) \\ &= \Big(\boldsymbol{u}_l^w(D)\boldsymbol{g}_1(D) \dots \boldsymbol{u}_l^w(D)\boldsymbol{g}_c(D)\Big) \\ &= \Big(v_l^{(1)}(D) \dots v_l^{(c)}(D)\Big), \end{aligned} \tag{5.8}$$

where $\boldsymbol{g}_j(D)$ is the jth column of the generator matrix $G(D)$, i.e.,

$$G(D) = \begin{pmatrix} g_{11}(D) & \dots & g_{1c}(D) \\ \vdots & & \vdots \\ g_{b1}(D) & \dots & g_{bc}(D) \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}_1(D) & \dots & \boldsymbol{g}_c(D) \end{pmatrix}. \tag{5.9}$$

Each of these l_w code sequences are serialized and written row-wise into a buffer. The code sequence for the warp, $\boldsymbol{v}^w(D)$, is read column-wise from the buffer and we obtain

$$\boldsymbol{v}^w(D) = \Big(\boldsymbol{v}^{(1)}(D) \dots \boldsymbol{v}^{(c)}(D)\Big), \tag{5.10}$$

where

$$\begin{aligned} \boldsymbol{v}^{(i)}(D) &= \Big(v_1^{(i)}(D) \dots v_{l_w}^{(i)}(D)\Big) \\ &= \Big(\boldsymbol{u}_1^w(D)\boldsymbol{g}_i(D) \dots \boldsymbol{u}_{l_w}^w(D)\boldsymbol{g}_i(D)\Big). \end{aligned} \tag{5.11}$$

To obtain a compact notation for the generator matrix of the warp we use the matrix (Kronecker) tensor product. This is a matrix consisting of all possible products with one element taken from each matrix.

Definition 5.1: Let A and B be two matrices of size $m \times n$ and $p \times q$, respectively. The *tensor product* $A \otimes B$ is the $mp \times nq$ matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}. \tag{5.12}$$

□

If the matrix products are defined, then

$$(A \otimes B)(C \otimes D) = AC \otimes BD. \tag{5.13}$$

This immediately gives that the inverse of the tensor product is the tensor product of the inverses,

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \tag{5.14}$$

Equation (5.10) can now be rewritten as

$$
\begin{aligned}
\boldsymbol{v}^w(D) = \Big(& \big(u_1^{(1)}(D)g_{11}(D)+\cdots+u_1^{(b)}(D)g_{b1}(D)\big) \\
& \ldots \ \big(u_{l_w}^{(1)}(D)g_{11}(D)+\cdots+u_{l_w}^{(b)}(D)g_{b1}(D)\big) \\
& \ldots \ \big(u_1^{(1)}(D)g_{1c}(D)+\cdots+u_1^{(b)}(D)g_{bc}(D)\big) \\
& \ldots \ \big(u_{l_w}^{(1)}(D)g_{1c}(D)+\cdots+u_{l_w}^{(b)}(D)g_{bc}(D)\big)\Big) \\
= & \Big(u_1^{(1)}(D)\ldots u_{l_w}^{(1)}(D) \ \ldots \ u_1^{(b)}(D)\ldots u_{l_w}^{(b)}(D)\Big) \\
& \times \begin{pmatrix}
g_{11}(D) & & & & & g_{1c}(D) & & & \\
& \ddots & & & & & \ddots & & \\
& & g_{11}(D) & & & & & g_{1c}(D) & \\
& \vdots & & & & & \vdots & & \\
g_{b1}(D) & & & & & g_{bc}(D) & & & \\
& \ddots & & & & & \ddots & & \\
& & g_{b1}(D) & & & & & g_{bc}(D) &
\end{pmatrix} \\
= & \boldsymbol{u}^w(D)\big(G(D)\otimes I_{l_w}\big),
\end{aligned}
\tag{5.15}
$$

where $\boldsymbol{u}^w(D)$ is given by (5.5) and I_{l_w} is the $l_w \times l_w$ identity matrix. From (5.15) we conclude that the $bl_w \times cl_w$ generator matrix $G^w(D)$ for a warp with l_w identical constituent encoders with a rational generator matrix $G(D)$ is

$$
G^w(D) = G(D) \otimes I_{l_w}. \tag{5.16}
$$

Clearly, we also have

$$
\boldsymbol{G}^w = \boldsymbol{G} \otimes I_{l_w}, \tag{5.17}
$$

where the generator matrix $\boldsymbol{G}$ is

$$
\boldsymbol{G} = \begin{pmatrix}
G_0 & G_1 & \cdots & G_m & & \\
& G_0 & G_1 & \cdots & G_m & \\
& & \ddots & \ddots & & \ddots
\end{pmatrix} \tag{5.18}
$$

if $G(D)$ is polynomial, otherwise

$$
\boldsymbol{G} = \begin{pmatrix}
G_0 & G_1 & G_2 & \cdots & \\
& G_0 & G_1 & G_2 & \cdots \\
& & \ddots & \ddots & \ddots
\end{pmatrix}, \tag{5.19}
$$

where the sequence $G_0\ G_1\ G_2\ \ldots$ is ultimately periodic.

Let m and ν be the memory and constraint length, respectively, of the constituent generator matrix. Then the generator matrix for the warp has memory

$$m_w = m \tag{5.20}$$

and overall constraint length

$$\nu_w = l_w \nu. \tag{5.21}$$

The right inverse of $G^w(D)$ is

$$(G^w(D))^{-1} = (G(D) \otimes I_{l_w})^{-1} = G^{-1}(D) \otimes I_{l_w}. \tag{5.22}$$

In Figure 5.7 we show a warp with l_w different encoders of rate $R_l = b_l/c_l = b/c$. To be able to do bit-wise multiplexing of the information and code sequences the rates must be identical.

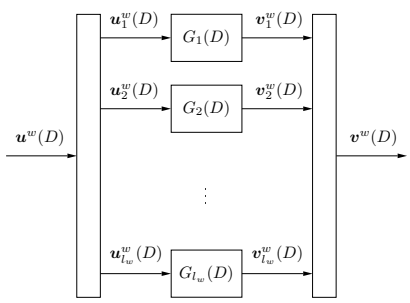

Figure 5.7: A warp with l_w constituent encoders with identical rates.

For this case calculations similar to those above can be performed, and we obtain

$$G^w(D) = \begin{pmatrix} g_{11}^{(1)}(D) & & & & g_{1c}^{(1)}(D) & & \\ & \ddots & & \cdots & & \ddots & \\ & & g_{11}^{(l_w)}(D) & & & & g_{1c}^{(l_w)}(D) \\ & \vdots & & & & \vdots & \\ g_{b1}^{(1)}(D) & & & & g_{bc}^{(1)}(D) & & \\ & \ddots & & \cdots & & \ddots & \\ & & g_{b1}^{(l_w)}(D) & & & & g_{bc}^{(l_w)}(D) \end{pmatrix} \tag{5.23}$$

$$= \sum_{l=1}^{l_w} G_l(D) \otimes \mathrm{diag}(\boldsymbol{e}_l),$$

where $\boldsymbol{e}_l$ denotes an l_w long binary vector with a one in the lth position and zeros otherwise, and $\mathrm{diag}(\boldsymbol{e}_l)$ a diagonal matrix with $\boldsymbol{e}_l$ as diagonal. Let m_l and ν_l be the memory and overall constraint length, respectively, of the lth constituent generator matrix, $G_l(D)$. Then the generator matrix for the warp has memory

$$m_w = \max_{1 \leqslant l \leqslant l_w} \{m_l\}, \tag{5.24}$$

and overall constraint length

$$\nu_w = \sum_{l=1}^{l_w} \nu_l. \tag{5.25}$$

Clearly, when all generator matrices are identical, (5.23) is equivalent to (5.16). In the sequel we will consider only warps with identical generator matrices but similar derivations can be performed for warps with different generator matrices.

5.3. Structural Properties of the Warp

We shall now show that the generator matrix for the warp with identical constituent encoders inherit many of the structural properties of the constituent generator matrices.

Theorem 5.1: Let $G_1^w(D) = G_1(D) \otimes I_{l_w}$ and $G_2^w(D) = G_2(D) \otimes I_{l_w}$. Then $G_1^w(D)$ and $G_2^w(D)$ are equivalent if and only if $G_1(D)$ and $G_2(D)$ are equivalent. □

Proof: Assume first that $G_1(D)$ and $G_2(D)$ are equivalent. Then there exists a non-singular matrix $T(D)$ such that $G_1(D) = T(D)G_2(D)$. We have

$$\begin{aligned} G_1^w(D) &= G_1(D) \otimes I_{l_w} = T(D)G_2(D) \otimes I_{l_w} \\ &= (T(D) \otimes I_{l_w})(G_2(D) \otimes I_{l_w}) \\ &= T^w(D)G_2^w(D), \end{aligned} \tag{5.26}$$

where

$$T^w(D) = T(D) \otimes I_{l_w} \tag{5.27}$$

is non-singular since $T(D)$ is non-singular. Hence, $G_1^w(D)$ and $G_2^w(D)$ are equivalent.

Next we assume that $G_1^w(D)$ and $G_2^w(D)$ are equivalent. Then there exists a non-singular matrix $T'(D)$ such that

$$G_1^w(D) = T'(D)G_2^w(D) \tag{5.28}$$

or, equivalently,

$$G_1(D) \otimes I_{l_w} = T'(D)(G_2(D) \otimes I_{l_w}). \tag{5.29}$$

Multiplying (5.29) by the right inverse of $G_2(D) \otimes I_{l_w}$ yields

$$\begin{aligned} T'(D) &= (G_1(D) \otimes I_{l_w})(G_2(D) \otimes I_{l_w})^{-1} \\ &= (G_1(D) \otimes I_{l_w})(G_2^{-1}(D) \otimes I_{l_w}) \\ &= G_1(D)G_2^{-1}(D) \otimes I_{l_w} \\ &= T''(D) \otimes I_{l_w}, \end{aligned} \tag{5.30}$$

where $T''(D) = G_1(D)G_2^{-1}(D)$ is non-singular since $T'(D)$ is non-singular. Hence, $G_1(D)$ and $G_2(D)$ are equivalent. ■

Theorem 5.2: Let $G^w(D)$ be the generator matrix of a warp consisting of l_w identical constituent encoders with a rational generator matrix $G(D)$. Then

(*i*) $G^w(D)$ is basic if and only if $G(D)$ is basic.

(*ii*) $G^w(D)$ is minimal if and only if $G(D)$ is minimal.

(*iii*) $G^w(D)$ is canonical if and only if $G(D)$ is canonical.

□

Proof: We begin with the proof for part (i). If $G(D)$ is a basic generator matrix it is polynomial and it has a polynomial inverse, $G^{-1}(D)$. Clearly, if $G(D)$ is polynomial so is $G^w(D) = G(D) \otimes I_{l_w}$. From (5.14) it is also clear that there exists a polynomial inverse for $G^w(D) = G(D) \otimes I_{l_w}$, viz., $(G^w(D))^{-1} = (G(D) \otimes I_{l_w})^{-1} = G^{-1}(D) \otimes I_{l_w}$. Hence, $G^w(D)$ is basic.

Similarly, if $G^w(D) = G(D) \otimes I_{l_w}$ is polynomial $G(D)$ is polynomial. From (5.14) follows that if $(G^w(D))^{-1} = (G(D) \otimes I_{l_w})^{-1} = G^{-1}(D) \otimes I_{l_w}$ is polynomial, which implies that $G^{-1}(D)$ is a polynomial inverse of $G(D)$. Hence, if $G^w(D) = G(D) \otimes I_{l_w}$ is basic so is $G(D)$.

Next consider part (ii) of the theorem. We will exploit that fact that a generator matrix is minimal if and only if [16]

$$\text{span}(\boldsymbol{u}(D)) \subseteq \text{span}(\boldsymbol{u}(D)G(D)) \tag{5.31}$$

for all information sequences $\boldsymbol{u}(D)$. The span is the interval from the first non-zero symbol to the last non-zero symbol.

Assume first that $G^w(D) = G(D) \otimes I_{l_w}$ is minimal. Then it follows that

$$\text{span}(\boldsymbol{u}^w(D)) \subseteq \text{span}(\boldsymbol{u}^w(D)(G(D) \otimes I_{l_w})). \tag{5.32}$$

From the construction of the warp (Figure 5.6) we conclude that for all indices l, $1 \leqslant l \leqslant l_w$, we have

$$\text{span}(\boldsymbol{u}_l^w(D)) \subseteq \text{span}(\boldsymbol{u}_l^w(D)G(D)) \tag{5.33}$$

and, hence, that $G(D)$ is minimal.

Next we assume that $G(D)$ is minimal. Then, for all information sequences for the lth encoder, $1 \leqslant l \leqslant l_w$, we have

$$\text{span}(\boldsymbol{u}_l^w(D)) \subseteq \text{span}(\boldsymbol{u}_l^w(D)G(D)). \tag{5.34}$$

Again, from the construction of the warp we conclude that for all information sequences $\boldsymbol{u}^w(D)$ we have

$$\text{span}(\boldsymbol{u}^w(D)) \subseteq \text{span}(\boldsymbol{u}^w(D)(G(D) \otimes I_{l_w})), \tag{5.35}$$

and, hence, that $G^w(D) = G(D) \otimes I_{l_w}$ is minimal.

Finally, we show part (iii) of the theorem. Assume that $G(D)$ is a canonical generator matrix with overall constraint length ν. Then the number of abstract states of $G(D)$ is equal to 2^ν. Since the code sequences of different constituent generator matrices do not affect each other, the number of abstract states of $G^w(D) = G(D) \otimes I_{l_w}$ is $(2^\nu)^{l_w} = 2^{l_w \nu}$. Since $G^w(D) = G(D) \otimes I_{l_w}$ is minimal (follows from (ii)) it is not possible to find an equivalent generator matrix with fewer abstract states. Thus, we cannot find an equivalent generator matrix with less overall constraint length than $l_w \nu$, which is the overall constraint length of $G^w(D) = G(D) \otimes I_{l_w}$. Hence, $G^w(D)$ is canonical.

Next, assume that $G^w(D) = G(D) \otimes I_{l_w}$ is canonical. Then its overall constraint length, ν, is minimal over all equivalent generator matrices. Especially, from Theorem 5.1 it is minimal over all generator matrices of the form $G'(D) \otimes I_{l_w}$, where $G'(D)$ is equivalent to $G(D)$. Since there is a direct connection between the overall constraint length of the constituent generator matrix and the generator matrix of the warp, this is the same as saying that the overall constraint length of $G(D)$ is minimal over all equivalents. Thus, $G(D)$ is canonical and the proof is completed. ■

We immediately have the following two corollaries.

Corollary 5.3: The generator matrix $G(D) \otimes I_{l_w}$ is minimal-basic if and only if the generator matrix $G(D)$ is minimal-basic. □

Proof: Follows from Theorem 5.2, parts (i) and (iii). ■

Corollary 5.4: If the warp consists of identical minimal encoders, then the warp is a minimal encoder itself. □

Proof: Let $G_{can}(D)$, with overall constraint length ν_{can}, be a canonical generator matrix equivalent to the generator matrix of the constituent encoders, $G(D)$, and let $\mathcal{C}^w$ be the code encoded by the warp. Then $(G_{can}(D) \otimes I_{l_w})$ is equivalent to $(G(D) \otimes I_{l_w})$. Since $(G_{can}(D) \otimes I_{l_w})$ is canonical we know that a minimal encoder for $\mathcal{C}^w$ consists of $l_w \nu_{can}$ delay-elements. The warp consists of l_w identical minimal encoders, each with ν_{can} delay-elements. Hence, the total number of delay-elements in the warp is $l_w \nu_{can}$. ■

Similar to Theorem 5.2 the next theorem follows from the structure of the warp.

Theorem 5.5: The generator matrix $G^w(D) = G(D) \otimes I_{l_w}$ for a warp with identical constituent generator matrices, $G(D)$, is non-catastrophic if and only if $G(D)$ is non-catastrophic. □

5.4. Structural Properties of the Twill

From the generator matrices of the warp it is easy to obtain the generator matrices for the woven encoders. The twill is a cascade of one outer warp

with l_o encoders and one inner warp with l_i encoders. The generator matrix for this construction can be written as

$$G^{tw} = (G^o \otimes I_{l_o})(G^i \otimes I_{l_i}). \tag{5.36}$$

The special cases woven convolutional codes with outer warp and with inner warp are obtained by letting $l_i = 1$ and $l_o = 1$, respectively. Thus, the generator matrix for a woven convolutional code with outer warp of l_o encoders can be written as

$$G^{ow} = (G^o \otimes I_{l_o})G^i, \tag{5.37}$$

and the generator matrix for a woven convolutional code with inner warp of l_i encoders as

$$G^{iw} = G^o(G^i \otimes I_{l_i}). \tag{5.38}$$

The same matrices can be derived as rational functions in D. Let e_o and e_i be the least integer scaling factors for the generator matrices for the outer and inner warps, respectively, such that the multiplication is defined. Then,

$$G^{tw}(D) = \Big(G^o(D) \otimes I_{l_o}\Big)_{[e_o]} \Big(G^i(D) \otimes I_{l_i}\Big)_{[e_i]}, \tag{5.39}$$

where $(G^o(D) \otimes I_{l_o})_{[e_o]}$ is the e_o times enlarged variant of $(G^o(D) \otimes I_{l_o})$ and $(G^i(D) \otimes I_{l_i})_{[e_i]}$ the e_i times enlarged variant of $(G^i(D) \otimes I_{l_i})$.

From Chapter 4 we can state some properties of the woven generator matrices. This will be done for the twill but is valid also for the other constructions.

Theorem 5.6: Let $G^o(D)$ and $G^i(D)$ be the generator matrices for the outer and inner constituent encoders, and $G^{tw}(D)$ the generator matrix for the twill. Then,

(*i*) $G^{tw}(D)$ is minimal if $G^o(D)$ and $G^i(D)$ both are minimal.

(*ii*) $G^{tw}(D)$ is basic if $G^o(D)$ and $G^i(D)$ both are basic.

□

Proof: From Theorem 5.2 part (*ii*) follows that the generator matrices for the outer and inner warp, $(G^o(D) \otimes I_{l_o})$ and $(G^o(D) \otimes I_{l_o})$, respectively, are minimal. Theorem 4.7 then shows that the corresponding cascade is minimal.

Theorem 5.2 also shows that the generator matrices for the outer and inner warp are basic if their respective constituent generator matrices are basic. Theorem 4.9 completes the proof. ■

We can, however, *not* make a similar statement for canonical or minimal-basic generator matrices. It was shown in Example 4.1 that the cascade of two minimal-basic generator matrices is not necessarily minimal-basic itself. This also means that we do not know if a woven construction consisting of minimal encoders is a minimal encoder or not. What we do know is that it is minimal and therefore can be realized with a minimum number of delay elements. Hence, if we find a canonical equivalent to $G^{tw}(D)$,

$$G^{tw}_{can}(D) = T(D)G^{tw}(D), \tag{5.40}$$

its overall constraint length can be upper bounded by

$$\nu_{tw} \leqslant l_o\nu_o + l_i\nu_i. \tag{5.41}$$

This follows from (5.21) and Theorem 4.8.

5.5. Distance Properties

We will start by estimating the free distance and the active distances for woven convolutional codes with outer and inner warp, respectively. Then the estimates for the free distances will be generalized to the twill.

To estimate the free distance and the active distances of a woven convolutional code with outer warp we need two new definitions derived from the active distances. We will use the affine functions from Section 3.3 that lower bound the active distances,

$$\begin{cases} a^r_j \geqslant f^r(j) \triangleq \alpha j + \beta^r, \\ a^b_j \geqslant f^b(j) \triangleq \alpha j + \beta^b, \\ a^c_j \geqslant f^c(j) \triangleq \alpha j + \beta^c, \\ a^{rc}_j \geqslant f^{rc}(j) \triangleq \alpha j + \beta^{rc}, \\ a^s_j \geqslant f^s(j) \triangleq \alpha j + \beta^s. \end{cases} \tag{5.42}$$

where α is the asymptotic slope of the active distances and the β:s are chosen as large as possible.

Definition 5.2: Let $j^b_{2\text{free}}$ denote the smallest j for which

$$f^b(j) \geqslant 2d_{\text{free}}, \tag{5.43}$$

i.e., $j^b_{2\text{free}}+1$ information tuples guarantee an output whose Hamming weight is at least $2d_{\text{free}}$, given that the encoder starts in the zero state and we do not have two consecutive zero states among the corresponding $j^b_{2\text{free}} + 2$ states, after which the encoder is back in the zero state. □

Consider the case when the output sequence from one of the outer encoders have two consecutive non-zero bits. The buffer is read column-wise to form the (binary) inner information sequence. This causes the two bits to be parted by $l_o - 1$ bits. If the inner encoder has feedback, then the first non-zero bit can cause it to leave the zero state, while the second non-zero bit can cause it to remerge with the zero state. To guarantee that those two non-zero bits generate an output of at least $2d^i_{\text{free}}$ from the inner encoder we must have

$$l_o \geqslant (j^{bi}_{2\text{free}} + 1)b_i. \tag{5.44}$$

Furthermore, to guarantee that three or more consecutive non-zero bits will generate at least d^i_{free} each in the output from the inner encoder we use the following definition.

Definition 5.3: Let j^{α}_{free} denote the smallest j for which

$$\alpha j \geqslant d_{\text{free}}, \tag{5.45}$$

i.e., $j^{\alpha}_{\text{free}} + 1$ additional information tuples guarantee an increase of at least d_{free} in the bounds on the active distances. □

Theorem 5.7: There exist woven convolutional codes with outer warp such that their free distances d^{ow}_{free} satisfy the inequality

$$d^{ow}_{\text{free}} \geqslant d^{o}_{\text{free}} d^{i}_{\text{free}} \tag{5.46}$$

where d^o_{free} and d^i_{free} denote the free distances of the outer and inner convolutional codes, respectively. □

Proof: Assume that the l_o outer encoders encode the same convolutional code and that l_o is large enough so each non-zero bit in a given row will generate the Hamming weight d^i_{free} at the output,

$$l_o \geqslant \max\left\{(j^{bi}_{2\text{free}} + 1)b_i,\ (j^{\alpha i}_{\text{free}} + 1)b_i\right\}. \tag{5.47}$$

Then the inequality in (5.46) is satisfied with equality. ■

Corollary 5.8: There exist woven convolutional codes with outer warp with polynomial constituent generator matrices such that their free distances d^{ow}_{free} satisfy the inequalities

$$d^{o}_{\text{free}} d^{i}_{\text{free}} \leqslant d^{ow}_{\text{free}} \leqslant d^{o}_{\text{free}} d^{ri}_{0}, \tag{5.48}$$

where d^o_{free} and d^i_{free} denote the free distances of the outer and inner convolutional codes, respectively, and d_0^{ri} denotes the 0th order row distance of the inner convolutional encoder. □

Proof: If all generator matrices are polynomial and $l_o > (m_i + 1)b_i$ the free distance of a woven convolutional code with outer warp cannot exceed $d^o_{\text{free}} d_0^{ri}$. ■

If we use at least two different outer convolutional codes with identical d^o_{free}, then the free distance of the combination *might* exceed the product of the free distances of the outer and inner convolutional codes!

Example 5.2: Use

$$G_1^o(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix} \tag{5.49}$$

and

$$G_2^o(D) = \begin{pmatrix} 1 + D^2 & 1 + D + D^2 \end{pmatrix} \tag{5.50}$$

as the outer encoders and

$$G^i(D) = \begin{pmatrix} 1 + D + D^2 + D^3 & 1 + D^2 + D^3 \end{pmatrix} \tag{5.51}$$

as the inner encoder. Their free distances are $d^o_{\text{free}} = 5$ and $d^i_{\text{free}} = 6$, respectively, and $d_0^{ri} = 7$. If we let the warp consist of 16 encoders with $G_1^o(D)$ and $G_2^o(D)$ alternating, then this woven convolutional code with outer warp has free distance

$$d^{ow}_{\text{free}} = 35, \tag{5.52}$$

which equals the upper bound in (5.48) and exceeds the product

$$d^o_{\text{free}} d^i_{\text{free}} = 30. \tag{5.53}$$

□

The next example shows that the upper bound in the corollary does not hold for all rational inner generator matrices.

Example 5.3: Let $G_1^o(D)$ and $G_2^o(D)$ be the same generator matrices as in Example 5.2. The warp consists of repeated blocks of four encoders: $G_1^o(D)$, $G_1^o(D)$, $G_2^o(D)$, and $G_2^o(D)$. Use

$$G^i(D) = \begin{pmatrix} 1 & \frac{1+D+D^2}{1+D^2} \end{pmatrix} \tag{5.54}$$

as inner generator matrix. The 0th order row distance for this generator matrix is the same as the free distance, and is generated by the information sequence $\boldsymbol{u}^i_{\text{free}} = (1\,0\,1\,0\,0\ldots)$. Assume that the first outer encoder generates the sequence with weight d_{free}, then the corresponding code sequence is $\boldsymbol{v}^o_1 = (11\ 10\ 11\ 00\ 00\ldots)$. To produce the weight $d^o_{\text{free}} d^{ri}_0$ the inner encoder must receive the sequence $\boldsymbol{u}^i_{\text{free}}$ for each non-zero bit in $\boldsymbol{v}^o_1$. Thus, the third outer encoder must also produce the code sequence $\boldsymbol{v}^o_3 = (11\ 10\ 11\ 00\ 00\ldots)$, but this encoder has $G^o_2(D)$ as generator matrix and $\boldsymbol{v}^o_3$ is not a valid codeword. The same reasoning hold if we start with any other outer encoder. Therefore,

$$d^{ow}_{\text{free}} > d^o_{\text{free}} d^{ri}_0. \tag{5.55}$$

□

Next we will lower bound the active distances for woven convolutional codes with outer warp. The number of bits per information tuple for the woven encoder is (by definition) $b_{ow} = l_o b_o$. One information tuple for the woven encoder gives one information tuple each for the l_o outer encoders. Each of these encoders generates c_o bits to be fed into the inner encoder, which gives $c_{ow} = l_o c_o c_i / b_i$ bits per code tuple.

Theorem 5.9: Consider a rate $R_{ow} = b_{ow}/c_{ow}$ woven convolutional code with outer warp where

$$l_o \geqslant \max\left\{(j^{bi}_{2\text{free}} + 1)b_i, (j^{\alpha i}_{\text{free}} + 1)b_i\right\}. \tag{5.56}$$

The rate of the outer and inner encoders are $R_o = b_o/c_o$ and $R_i = b_i/c_i$, respectively, where $b_{ow} = l_o b_o$ and $c_{ow} = l_o c_o c_i / b_i$. The active distances for this woven convolutional code are lower bounded by

$$a^{row}_j \geqslant \alpha^o d^i_{\text{free}} j + \beta^{ro} d^i_{\text{free}}, \tag{5.57a}$$

if $\beta^{co} \geqslant \alpha^o + 1$ or $\beta^{rco} \geqslant \alpha^o + 1$,

$$a^{bow}_j \geqslant \alpha^o d^i_{\text{free}} j + \beta^{bo} d^i_{\text{free}}, \tag{5.57b}$$

if $\beta^{co} \geqslant \alpha^o + 1$ or $\beta^{rco} \geqslant \alpha^o + 1$,

$$a^{cow}_j \geqslant \alpha^o d^i_{\text{free}} j + \beta^{co} d^i_{\text{free}}, \tag{5.57c}$$

if $\beta^{co} \leqslant \beta^{bo} + \beta^{so} - \alpha^o - 1$ or $\beta^{co} \geqslant \alpha^o + 1$,

$$a^{rcow}_j \geqslant \alpha^o d^i_{\text{free}} j + \beta^{rco} d^i_{\text{free}}, \tag{5.57d}$$

if $\beta^{rco} \leqslant \beta^{bo} + \beta^{so} - \alpha^o - 1$ or $\beta^{rco} \geqslant \alpha^o + 1$, and

$$a_j^{sow} \geqslant \alpha^o d_{\text{free}}^i j + \beta^{so} d_{\text{free}}^i, \tag{5.57e}$$

if $\beta^{co} \geqslant \alpha^o + 1$ or $\beta^{rco} \geqslant \alpha^o + 1$. The parameters α^o, β^{ro}, β^{bo}, β^{co}, β^{rco}, and β^{so} defines the lower bounds on the active distances for the outer convolutional codes, and d_{free}^i is the free distance of the inner convolutional code. □

Proof: We will only consider the case when all the encoders of the warp do not have consecutive zero states at the same time. Thus, we let the states of the encoders of the warp represent the state of the woven convolutional encoder with outer warp. Then there will be long (about $j_0^{so} c_o l_o / b_i$ b_i-tuples) sub-sequences of the inner information sequence that are zero, which allow the inner encoder to be in the zero state for that time. This will give a result that is less than the case when it is the inner encoder that is not allowed to have consecutive zero states.

Assume that the first non-zero bit in the information sequence, $\boldsymbol{u}^{ow}$, is fed to the first outer encoder, $\mathcal{E}_1^o$. Start by lower bounding the active column distance when the path followed by one of the outer encoders differs from the all-zero path during the total interval. The first encoder $\mathcal{E}_1^o$ diverge from the all-zero path and remerges after $j+1$ steps. Its output sequence will have at least weight a_j^{co}. Each of these non-zero bits will generate at least d_{free}^i ones in the output sequence from the woven encoder, $\boldsymbol{v}^{ow}$. Thus,

$$a_j^{cow} \geqslant a_j^{co} d_{\text{free}}^i \geqslant (\alpha^o j + \beta^{co}) d_{\text{free}}^i = \alpha^o d_{\text{free}}^i j + \beta^{co} d_{\text{free}}^i. \tag{5.58}$$

Similar calculations can be carried out for the remaining active distances, and we get

$$\begin{cases} \alpha^{ow} = \alpha^o d_{\text{free}}^i \\ \beta^{row} = \beta^{ro} d_{\text{free}}^i \\ \beta^{bow} = \beta^{bo} d_{\text{free}}^i \\ \beta^{cow} = \beta^{co} d_{\text{free}}^i \\ \beta^{rcow} = \beta^{rco} d_{\text{free}}^i \\ \beta^{sow} = \beta^{so} d_{\text{free}}^i. \end{cases} \tag{5.59}$$

Next consider the situation when all outer encoders have consecutive zero states. When one encoder remerges with the all-zero path some other must differ from it. Alternatively, if one encoder diverges from the all-zero path another must differ from the all-zero path up to that point. We will lower bound the active distances for these cases and find restrictions for which the formulas in (5.59) still holds.

Again we begin with the active column distance. All encoders are in the zero state at time zero and at least one diverges in the next step. When this encoder remerges with the all-zero path, at time $k+1$ say, there is another encoder that differs from the all-zero path and we assume that this will continue until time $j+1$. From the first part, from time zero to $k+1$, we get at least weight a_k^{bo} and from the second part, from $k+1$ to $j+1$, we get at least a_{j-k-1}^{so}. If the first part is due to the last encoder, $\mathcal{E}_{l_o}^o$, and the second part to the first encoder, $\mathcal{E}_1^o$, the last bit in the first part will be directly followed by the first bit in the second part in the information sequence for the inner encoder. Thus, they will generate at least weight d_{free}^i, instead of $2d_{\text{free}}^i$. The number of columns that give rise to d_{free}^i non-zero bits in the inner encoder is then at least $a_k^{bo} + a_{j-k-1}^{so} - 1$, so

$$
\begin{aligned}
a_j^{cow} &\geqslant (a_k^{bo} + a_{j-k-1}^{so} - 1)d_{\text{free}}^i \\
&\geqslant (\alpha^o(j-1) + \beta^{bo} + \beta^{so} - 1)d_{\text{free}}^i \\
&= (\alpha^o j + \beta^{co} - \beta^{co} + \beta^{bo} + \beta^{so} - \alpha^o - 1)d_{\text{free}}^i \\
&\geqslant \alpha^o d_{\text{free}}^i j + \beta^{co} d_{\text{free}}^i,
\end{aligned}
\tag{5.60}
$$

where the last inequality requires $\beta^{co} \leqslant \beta^{bo} + \beta^{so} - \alpha^o - 1$. Alternatively, assume that the second part starts in the zero state at time $k+1$,

$$
\begin{aligned}
a_j^{cow} &\geqslant (a_k^{co} + a_{j-k-1}^{co} - 1)d_{\text{free}}^i \\
&\geqslant (\alpha^o j + \beta^{co} + \beta^{co} - \alpha^o - 1)d_{\text{free}}^i \\
&\geqslant \alpha^o d_{\text{free}}^i j + \beta^{co} d_{\text{free}}^i,
\end{aligned}
\tag{5.61}
$$

if $\beta^{co} \geqslant \alpha^o + 1$. Since both cases occur simultaneously only one of the requirements needs to be fulfilled for (5.58) to be true. Note that both those cases cover bounces at the zero state, since the active segment distance can start in the zero state and the active column distance can end in the zero state.

The active reverse column distance can be calculated in the same way. Let the last encoder $\mathcal{E}_{l_o}^o$ start in an arbitrary state and the encoder $\mathcal{E}_1^o$ diverge from the all-zero path at time $k+1$. Then,

$$
\begin{aligned}
a_j^{rcow} &\geqslant (a_k^{so} + a_{j-k-1}^{bo} - 1)d_{\text{free}}^i \\
&\geqslant (\alpha^o j + \beta^{rco} - \beta^{rco} + \beta^{so} + \beta^{bo} - \alpha^o - 1)d_{\text{free}}^i \\
&\geqslant \alpha^o d_{\text{free}}^i j + \beta^{rco} d_{\text{free}}^i,
\end{aligned}
\tag{5.62}
$$

if $\beta^{rco} \leqslant \beta^{so} + \beta^{bo} - \alpha^o - 1$. Alternatively, let the last encoder remerge with

the zero path at time $k+1$,

$$
\begin{aligned}
a_j^{rcow} &\geqslant (a_k^{rco} + a_{j-k-1}^{rco} - 1)d_{\text{free}}^i \\
&\geqslant (\alpha^o j + \beta^{rco} + \beta^{rco} - \alpha^o - 1)d_{\text{free}}^i \\
&\geqslant \alpha^o d_{\text{free}}^i j + \beta^{rco} d_{\text{free}}^i,
\end{aligned} \tag{5.63}
$$

if $\beta^{rco} + \alpha^o + 1$.

To lower bound the active segment distance the derivation is again split into two parts. First, let an encoder start in an arbitrary state and remerge with the all-zero path at time $k+1$. From that point until $j+1$ there is another encoder that differs from the all-zero path. This gives

$$
\begin{aligned}
a_j^{sow} &\geqslant (a_k^{rco} + a_{j-k-1}^{so} - 1)d_{\text{free}}^i \\
&\geqslant (\alpha^o j + \beta^{so} + \beta^{rco} - \alpha^o - 1)d_{\text{free}}^i \\
&\geqslant \alpha^o d_{\text{free}}^i j + \beta^{so} d_{\text{free}}^i,
\end{aligned} \tag{5.64}
$$

if $\beta^{rco} \geqslant \alpha^o + 1$. The second case gives the active segment distance from time zero to $k+1$ and the active column distance from $k+1$ until $j+1$. Thus,

$$
\begin{aligned}
a_j^{sow} &\geqslant (a_k^{so} + a_{j-k-1}^{co} - 1)d_{\text{free}}^i \\
&\geqslant (\alpha^o j + \beta^{so} + \beta^{co} - \alpha^o - 1)d_{\text{free}}^i \\
&\geqslant \alpha^o d_{\text{free}}^i j + \beta^{so} d_{\text{free}}^i,
\end{aligned} \tag{5.65}
$$

if $\beta^{co} \geqslant \alpha^o + 1$.

The same cases can be considered for the active row distance. Let one encoder diverge from the all-zero path at time zero and remerge at time $k+1$. Another encoder differs from the all-zero path during the remaining time until $j+1+m$ when all outer encoders are in the zero state,

$$
\begin{aligned}
a_j^{row} &\geqslant (a_{k-m}^{ro} + a_{j-k-1+m}^{rco} - 1)d_{\text{free}}^i \\
&\geqslant (\alpha^o j + \beta^{ro} + \beta^{rco} - \alpha^o - 1)d_{\text{free}}^i \\
&\geqslant \alpha^o d_{\text{free}}^i j + \beta^{ro} d_{\text{free}}^i,
\end{aligned} \tag{5.66}
$$

if $\beta^{rco} \geqslant \alpha^o + 1$. For the second case, an encoder diverges from the all-zero path at time $k+1$ and remerges at time $j+1+m$. Before that another encoder diverged from the all-zero path at time zero and did not remerge until after time k,

$$
\begin{aligned}
a_j^{row} &\geqslant (a_k^{co} + a_{j-k-1}^{ro} - 1)d_{\text{free}}^i \\
&\geqslant (\alpha^o j + \beta^{ro} + \beta^{co} - \alpha^o - 1)d_{\text{free}}^i \\
&\geqslant \alpha^o d_{\text{free}}^i j + \beta^{ro} d_{\text{free}}^i,
\end{aligned} \tag{5.67}
$$

if $\beta^{rco} \geqslant \alpha^o + 1$. Hence, we end up with the same restrictions as for the active segment distance.

Finally, we have again the same cases for the active burst distance. The first case gives

$$\begin{aligned} a_j^{bow} &\geqslant (a_k^{bo} + a_{j-k-1}^{rco} - 1)d_{\text{free}}^i \\ &\geqslant (\alpha^o j + \beta^{bo} + \beta^{rco} - \alpha^o - 1)d_{\text{free}}^i \\ &\geqslant \alpha^o d_{\text{free}}^i j + \beta^{ro} d_{\text{free}}^i, \end{aligned} \tag{5.68}$$

if $\beta^{rco} \geqslant \alpha^o + 1$. The second case gives

$$\begin{aligned} a_j^{bow} &\geqslant (a_k^{co} + a_{j-k-1}^{bo} - 1)d_{\text{free}}^i \\ &\geqslant (\alpha^o j + \beta^{bo} + \beta^{co} - \alpha^o - 1)d_{\text{free}}^i \\ &\geqslant \alpha^o d_{\text{free}}^i j + \beta^{bo} d_{\text{free}}^i, \end{aligned} \tag{5.69}$$

if $\beta^{co} \geqslant \alpha^o + 1$, and the proof is complete. ■

Example 5.4: Construct a woven encoder with l_o outer encoders corresponding to

$$G^o(D) = \begin{pmatrix} 1 + D + D^2 + D^3 + D^5 & 1 + D^2 + D^3 + D^5 \end{pmatrix} \tag{5.70}$$

and one inner encoder corresponding to

$$G^i(D) = \begin{pmatrix} 1 + D + D^2 + D^3 + D^5 & 1 + D^2 + D^3 + D^4 + D^5 \end{pmatrix}. \tag{5.71}$$

From Example 3.6 and Example 3.7 the lower bounds on the active distances for the outer and inner encoders are defined by

$$\mathcal{A}^o = \left\{ \alpha^o = \frac{1}{3},\ \beta^{ro} = \frac{19}{3},\ \beta^{bo} = \frac{14}{3},\ \beta^{co} = 2,\ \beta^{rco} = 1,\ \beta^{so} = -1 \right\}. \tag{5.72}$$

and

$$\mathcal{A}^i = \left\{ \alpha^i = \frac{2}{7},\ \beta^{ri} = \frac{46}{7},\ \beta^{bi} = \frac{36}{7},\ \beta^{ci} = \beta^{rci} = 2,\ \beta^{si} = -\frac{6}{7} \right\}, \tag{5.73}$$

respectively. Both the outer and the inner generator matrices give the free distance $d_{\text{free}} = 8$. Equation (5.73) yields that $j_{\text{free}}^{bi} = 38$ and $j_{\text{free}}^{\alpha i} = 28$, and we must use at least $l_o \geqslant \max\{39, 29\} = 39$ outer encoders.

The requirements for the inequalities in (5.57) are fulfilled since

$$\frac{7}{3} = \beta^{bo} + \beta^{so} - \alpha^o - 1 \geqslant \begin{cases} \beta^{co} = 2 \\ \beta^{rco} = 1 \end{cases} \tag{5.74}$$

and

$$2 = \beta^{co} \geqslant \alpha^{o} + 1 = \frac{4}{3}. \tag{5.75}$$

The lower bounds for this woven convolutional code with outer warp are given by

$$\begin{cases} \alpha^{ow} = \alpha^{o} d^{i}_{\text{free}} = \dfrac{8}{3} \\ \beta^{row} = \beta^{ro} d^{i}_{\text{free}} = \dfrac{152}{3} (\approx 50.7) \\ \beta^{bow} = \beta^{bo} d^{i}_{\text{free}} = \dfrac{112}{3} (\approx 37.3) \\ \beta^{cow} = \beta^{co} d^{i}_{\text{free}} = 16 \\ \beta^{rcow} = \beta^{rco} d^{i}_{\text{free}} = 8 \\ \beta^{sow} = \beta^{so} d^{i}_{\text{free}} = -8. \end{cases} \tag{5.76}$$

The slope, α^{ow}, is calculated per information tuples for the woven encoder, i.e., per $b_o l_o$ bits. Calculated per b_o bits,

$$\alpha^{ow}_{(b_o)} = \frac{\alpha^{ow}_{(b_o l_o)}}{l_o} \leqslant \frac{8/3}{39} = \frac{8}{117}, \tag{5.77}$$

which is about one fifth of the slope for the outer encoders. Since the outer encoders are identical, the free distance is

$$d^{ow}_{\text{free}} = d^{o}_{\text{free}} d^{i}_{\text{free}} = 64. \tag{5.78}$$

□

There exist also woven convolutional codes with inner warp whose free distances satisfy the inequalities given in Theorem 5.7 and Corollary 5.8. To show this we first need two definitions derived from the active distances, cf. Definition 5.2 and Definition 5.3.

Definition 5.4: Let j^{c}_{free} denote the smallest j for which

$$f^{c}(j) \geqslant d_{\text{free}}, \tag{5.79}$$

i.e., $j^{c}_{\text{free}} + 1$ information tuples guarantee an output whose Hamming weight is at least d_{free} given that the encoder starts in the zero state and we do not have two consecutive zero states among the first $j^{c}_{\text{free}} + 2$ states. □

Definition 5.5: Let j_{free}^{rc} denote the smallest j for which

$$f^{rc}(j) \geqslant d_{\text{free}}, \tag{5.80}$$

i.e., $j_{\text{free}}^{rc}+1$ information tuples guarantee an output whose Hamming weight is at least d_{free} given that the encoder terminates in the zero state and we do not have two consecutive zero states among the last $j_{\text{free}}^{rc}+2$ states. □

Theorem 5.10: There exist woven convolutional codes with inner warp such that their free distances d_{free}^{iw} satisfy the inequality

$$d_{\text{free}}^{iw} \geqslant d_{\text{free}}^{o} d_{\text{free}}^{i} \tag{5.81}$$

where d_{free}^{o} and d_{free}^{i} denote the free distances of the outer and inner convolutional codes, respectively. □

Proof: Assume that the l_i inner encoders are identical and that l_o is large enough so a non-zero output from the outer encoder will give rise to non-zero inputs for at least d_{free}^{o} inner encoders,

$$l_i \geqslant \min\left\{(j_{\text{free}}^{co}+1)c_o, (j_{\text{free}}^{rco}+1)c_o\right\}. \tag{5.82}$$

Then the inequality in (5.81) is satisfied with equality. ■

Corollary 5.11: There exist woven convolutional codes with inner warp with polynomial constituent generator matrices such that their free distances d_{free}^{iw} satisfy the inequalities

$$d_{\text{free}}^{o} d_{\text{free}}^{i} \leqslant d_{\text{free}}^{iw} \leqslant d_{\text{free}}^{o} d_{0}^{ri}, \tag{5.83}$$

where d_{free}^{o} and d_{free}^{i} denote the free distances of the outer and inner convolutional codes, respectively, and d_0^{ri} denotes the 0th order row distance of the inner convolutional encoders. □

Proof: If all generator matrices are polynomial and l_i satisfies (5.82) the free distance of a woven convolutional code with inner warp cannot exceed $d_{\text{free}}^{o} d_0^{ri}$. ■

Example 5.5: Let the outer encoder be

$$G^o(D) = \begin{pmatrix} 1+D+D^2 & 1+D^2 \end{pmatrix} \tag{5.84}$$

and let the inner warp consist of six encoders;

$$G_1^i(D) = \begin{pmatrix} 1 + D^2 + D^4 & 1 + D + D^2 + D^3 + D^4 \end{pmatrix} \tag{5.85}$$

and

$$G_2^i(D) = \begin{pmatrix} 1 + D + D^3 & 1 + D + D^2 + D^3 + D^4 \end{pmatrix} \tag{5.86}$$

alternating. The free distance for the outer code is $d_{\text{free}}^o = 5$ and both of the inner codes have $d_{\text{free}}^i = 6$. The 0th order row distance for both of the inner encoders are $d_0^{ri} = 8$. The free distance for this construction is

$$d_{\text{free}}^{iw} = 34, \tag{5.87}$$

which exceeds the product

$$d_{\text{free}}^o d_{\text{free}}^i = 30. \tag{5.88}$$

□

As for woven convolutional codes with outer warp the upper bound in the corollary does not hold for all rational inner generator matrices.

Example 5.6: Let the outer generator matrix be

$$G^o(D) = \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix}, \tag{5.89}$$

and let the inner warp consist of the two systematic rational generator matrices

$$G_1^i(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix} \tag{5.90}$$

and

$$G_2^i(D) = \begin{pmatrix} 1 & \frac{1+D+D^2}{1+D^2} \end{pmatrix} \tag{5.91}$$

alternating. The 0th order row distance for $G_1^i(D)$ is generated by $\boldsymbol{u}_1^i = (1\ 1\ 1\ 0\ 0 \ldots)$ and for $G_2^i(D)$ by $\boldsymbol{u}_2^i = (1\ 0\ 1\ 0\ 0 \ldots)$. To get an overall code sequence of weight $d_{\text{free}}^o d_0^{ri}$ the outer encoder gives the free distance sequence, $\boldsymbol{v}_1^o = (11\ 10\ 11\ 00\ 00 \ldots)$, in the first column of the buffer. If l_i is large enough the outer encoder is back in the zero state when starting to produce the symbols for the second column. To achieve d_0^{ri} in each non-zero row it must give $\boldsymbol{v}_2^o = (10\ 10\ 10\ 00\ 00 \ldots)$ which is impossible. Thus,

$$d_{\text{free}}^{iw} > d_{\text{free}}^o d_0^{ri}. \tag{5.92}$$

□

Assume that when the outer encoder differs from the zero state in l_i/c_o steps it will generate at least d^o_{free} non-zero rows in the buffer, i.e., l_i satisfies (5.82). This implies that at each time either all of the inner encoders are in the zero state, or at least d^o_{free} of them differs from it. In our investigation of the active distances we will look into the state paths for the inner encoders corresponding to these non-zero rows.

We begin with the active column distance and let all encoders start in the zero state. Then there are always some of the inner encoders that differ from the all-zero path. If a non-zero row is such that the corresponding state sequence for the inner encoder differs from the all-zero path in $j+1$ steps its contribution to the total Hamming weight is at least a^{ci}_j.

On the other hand, if the encoder remerges with the all-zero path there is another encoder that diverges or already differs from it. The contribution from these two encoders is at least

$$\begin{aligned} a^{bi}_k + a^{si}_{j-k-1} &\geqslant \alpha^i(k+j-k-1) + \beta^{bi} + \beta^{si} \\ &= \alpha^i j + \beta^{ci} - \beta^{ci} + \beta^{bi} + \beta^{si} - \alpha^i \\ &\geqslant \alpha^i j + \beta^{ci}, \end{aligned} \tag{5.93}$$

if $\beta^{ci} \leqslant \beta^{bi} + \beta^{si} - \alpha^i$. Alternatively, let the second encoder diverge from the zero state at time $k+1$. Then the least contribution is

$$\begin{aligned} a^{bi}_k + a^{si}_{j-k-1} &\geqslant \alpha^i j + \beta^{ci} + \beta^{ci} - \alpha^i \\ &\geqslant \alpha^i j + \beta^{ci}, \end{aligned} \tag{5.94}$$

if $\beta^{ci} \geqslant \alpha^i$. Since there are at least d^o_{free} non-zero columns at each time we get

$$a^{ciw}_j \geqslant d^o_{\text{free}}(\alpha^i j + \beta^{ci}) = d^o_{\text{free}}\alpha^i j + d^o_{\text{free}}\beta^{ci}, \tag{5.95}$$

if either of the restrictions is fulfilled. The derivations for the bounds on the remaining active distances will be similar. Therefore, if the restrictions

$$\beta^{ci} \leqslant \beta^{bi} + \beta^{si} - \alpha^i \text{ or } \beta^{ci} \geqslant \alpha^i, \tag{5.96}$$

$$\beta^{rci} \leqslant \beta^{bi} + \beta^{si} - \alpha^i \text{ or } \beta^{rci} \geqslant \alpha^i, \tag{5.97}$$

and

$$\beta^{rci} \geqslant \alpha^i \text{ or } \beta^{ci} \geqslant \alpha^i \tag{5.98}$$

are satisfied the lower bounds on the active distances are described by

$$\begin{cases} \alpha^{iw} = d^o_{\text{free}}\alpha^i \\ \beta^{riw} = d^o_{\text{free}}\beta^{ri} \\ \beta^{biw} = d^o_{\text{free}}\beta^{bi} \\ \beta^{ciw} = d^o_{\text{free}}\beta^{ci} \\ \beta^{rciw} = d^o_{\text{free}}\beta^{rci} \\ \beta^{siw} = d^o_{\text{free}}\beta^{si}. \end{cases} \tag{5.99}$$

We summarize the lower bounds as a theorem.

Theorem 5.12: Consider a rate $R_{iw} = b_{iw}/c_{iw}$ woven convolutional code with inner warp and with

$$l_i \geqslant \min\left\{(j^{co}_{\text{free}} + 1)c_o, (j^{rco}_{\text{free}} + 1)c_o\right\} \tag{5.100}$$

inner encoders. The rates of the outer and inner encoders are $R_o = b_o/c_o$ and $R_i = b_i/c_i$, respectively, where $b_{iw} = l_i b_i b_o/c_o$ and $c_{iw} = l_i c_i$. The active distances for this woven convolutional code are lower bounded by

$$a^{riw}_j \geqslant \alpha^i d^o_{\text{free}} j + \beta^{ri} d^o_{\text{free}}, \tag{5.101a}$$

if $\beta^{ci} \geqslant \alpha^i$ or $\beta^{rci} \geqslant \alpha^i$,

$$a^{biw}_j \geqslant \alpha^i d^o_{\text{free}} j + \beta^{bi} d^o_{\text{free}}, \tag{5.101b}$$

if $\beta^{ci} \geqslant \alpha^i$ or $\beta^{rci} \geqslant \alpha^i$,

$$a^{ciw}_j \geqslant \alpha^i d^o_{\text{free}} j + \beta^{ci} d^o_{\text{free}}, \tag{5.101c}$$

if $\beta^{ci} \leqslant \beta^{bi} + \beta^{si} - \alpha^i$ or $\beta^{ci} \geqslant \alpha^i$,

$$a^{rciw}_j \geqslant \alpha^i d^o_{\text{free}} j + \beta^{rci} d^o_{\text{free}}, \tag{5.101d}$$

if $\beta^{rci} \leqslant \beta^{bi} + \beta^{si} - \alpha^i$ or $\beta^{rci} \geqslant \alpha^i$, and

$$a^{siw}_j \geqslant \alpha^i d^o_{\text{free}} j + \beta^{si} d^o_{\text{free}}, \tag{5.101e}$$

if $\beta^{ci} \geqslant \alpha^i$ or $\beta^{rci} \geqslant \alpha^i$. The parameters α^i, β^{ri}, β^{bi}, β^{ci}, β^{rci}, and β^{si} define the lower bounds on the active distances for the inner convolutional codes, and d^o_{free} is the free distance of the outer convolutional code. □

Example 5.7: Let the generator matrices for the outer and inner encoders be the same as in Example 5.4. It is easily checked that the conditions in Theorem 5.12 are satisfied. From (5.72) we get $j^{co}_{\text{free}} = 18$ and $j^{rco}_{\text{free}} = 21$ and the inner warp must consist of $l_i \geqslant \min\{38, 44\} = 38$ encoders. The lower bounds on the active distances for this woven convolutional code with inner warp are given by

$$\begin{cases} \alpha^{iw} = \alpha^i d^o_{\text{free}} = \dfrac{16}{7} \\ \beta^{riw} = \beta^{ri} d^o_{\text{free}} = \dfrac{368}{7} (\approx 52.6) \\ \beta^{biw} = \beta^{bi} d^o_{\text{free}} = \dfrac{288}{7} (\approx 41.1) \\ \beta^{ciw} = \beta^{ci} d^o_{\text{free}} = 16 \\ \beta^{rciw} = \beta^{rci} d^o_{\text{free}} = 16 \\ \beta^{siw} = \beta^{si} d^o_{\text{free}} = -\dfrac{48}{7} (\approx -6.9). \end{cases} \tag{5.102}$$

□

The bounds on the free distance for woven convolutional codes with outer and inner warps can be viewed as special cases of the following theorems with bounds on the free distance for the twill.

Theorem 5.13: There exist twills with relatively prime numbers of outer and inner encoders such that their free distances d^{tw}_{free} satisfy the inequality

$$d^{tw}_{\text{free}} \geqslant d^o_{\text{free}} d^i_{\text{free}}, \tag{5.103}$$

where d^o_{free} and d^i_{free} denotes the free distance of the outer and the inner convolutional codes, respectively. □

Proof: The code sequences from the l_o outer encoders are serialized and written row-wise into a buffer with l_o rows. The buffer is read column-wise and the corresponding sequence is split bit by bit into the l_i information sequences. Any l_o consecutive symbols read column-wise from the buffer arise from different encoders. Furthermore, since l_o and l_i are relatively prime, any l_i consecutive symbols from a specific row of the buffer will be read by different inner encoders. Therefore, if l_o is large enough so each non-zero bit in a given row will generate at least the Hamming weight d^i_{free} at the output, i.e., if

$$l_o \geqslant \max\left\{(j^{bi}_{2\text{free}} + 1)b_i, (j^{\alpha i}_{\text{free}} + 1)b_i\right\}, \tag{5.104}$$

or l_i is large enough so a non-zero output in any row give rise to non-zero inputs for at least d^o_{free} inner encoders, i.e., if

$$l_i \geqslant \min\left\{(j^{co}_{\text{free}}+1)c_o, (j^{rco}_{\text{free}}+1)c_o\right\}, \tag{5.105}$$

we get at least free distance $d^{tw}_{\text{free}} \geqslant d^o_{\text{free}} d^i_{\text{free}}$. ■

Since the information sequences for the inner encoders are not read from a single row or a single column (at a time) we do not, as in the case of single warps, necessarily have equality in (5.103) when the warps consists of identical encoders.

Corollary 5.14: There exist twills with polynomial constituent generator matrices and relatively prime numbers of outer and inner encoders, such that their free distances d^{iw}_{free} satisfy the inequalities

$$d^o_{\text{free}} d^i_{\text{free}} \leqslant d^{tw}_{\text{free}} \leqslant d^o_{\text{free}} d^{ri}_0, \tag{5.106}$$

where d^o_{free} and d^i_{free} denote the free distances of the outer and inner convolutional codes, respectively, and d^{ri}_0 denotes the 0th order row distance of the inner convolutional encoders. □

Proof: If all generator matrices are polynomial and l_o and l_i are chosen such that (5.104) or (5.105) is satisfied and $\gcd(l_o, l_i) = 1$, the free distance of the twill cannot exceed $d^o_{\text{free}} d^{ri}_0$. ■

6

Decoding of Woven Convolutional Codes

The encoder of a woven convolutional code can be regarded as a two step encoder. Therefore, it is suitable to apply an iterative decoding scheme. Iterative decoding became popular in 1993 when Turbo codes [5, 22] were introduced. It uses soft decision decoding and information from one constituent decoder is passed on to another. For the decoder of woven convolutional codes a windowed soft-in-soft-out constituent decoder is needed. Here the BCJR algorithm [2] is considered. This is an algorithm that can be used for example when performing maximum *a posteriori* (MAP) decoding or as here in an iterative decoding scheme for calculating the *a posteriori* probabilities (APP).

A windowed variant of the BCJR-algorithm will be described in Section 6.1 and in Section 6.2 it will be used to construct an iterative decoding scheme. In Section 6.3 simulation results will be shown for the twill and for woven convolutional codes with outer warp. To decrease the decoding error probabilities, an additional interleaver will be introduced.

6.1. The BCJR Algorithm

In order to apply an iterative decoding scheme we need a soft output decoding algorithm for the constituent codes. The most well known such algorithms are the BCJR[1] algorithm [2] and the soft output Viterbi (SOVA) algorithm [19]. Here we will use a variant of the BCJR algorithm. Given the received sequence $\boldsymbol{r}$ and the *a priori* probabilities for the bits of either the information sequence or the code sequence, $P(u_t^{(j)})$ or $P(v_t^{(j)})$, respectively, we will calculate the corresponding bit-wise *a posteriori* probabilities, $P(u_t^{(j)}|\boldsymbol{r})$ or $P(v_t^{(j)}|\boldsymbol{r})$, see Figure 6.1. This can be used as an *a posteriori* probability (APP) module in an iterative scheme or together with a maximizing device as a maximum *a posteriori* (MAP) decoder.

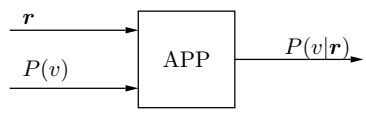

Figure 6.1: The module used to calculate the *a posteriori* probabilities (APP).

In its original form the BCJR algorithm must be applied to truncated frames and it has substantial numerical problems. By changing the notation slightly, see also [3], we are able to normalize the values at each time as well as using a windowed variant of the algorithm. The windowed version is, of course, sub-optimal but it can work on arbitrarily (infinitely) long frames.

When deriving the algorithm we will first consider a trellis for a truncated time-invariant convolutional code, and then apply a sliding window technique. Denote the information sequence, including the zero state driving sequence, by

$$\boldsymbol{u}_{[0,T)} = (\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{T-1}), \tag{6.1}$$

where each symbol $\boldsymbol{u}_t$ is a binary b-tuple and T is the length of the frame. The corresponding code and state sequences are

$$\boldsymbol{v}_{[0,T)} = (\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{T-1}), \tag{6.2}$$

[1] The name is taken from the initials of the inventors. It is also called the forward-backward algorithm or the two-way algorithm.

where $\boldsymbol{v}_t$ are binary c-tuples, and

$$\boldsymbol{S}_{[0,T]} = (\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_T), \tag{6.3}$$

respectively. The state $\boldsymbol{\sigma}_t$ is a binary matrix representing the physical state at time t of the encoder. We will denote the set of all possible states by $\mathcal{S}$. Similarly, the received sequence is denoted by

$$\boldsymbol{r}_{[0,T)} = (\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_{T-1}), \tag{6.4}$$

where each received symbol $\boldsymbol{r}_t$ is a c-tuple of channel symbols. We will assume a memoryless AWGN-channel, quantized into a finite number of levels, e.g., eight levels. Given the submitted code sequence the probability for the received sequence $\boldsymbol{r}_{[0,T)}$ is

$$P(\boldsymbol{r}_{[0,T)}|\boldsymbol{v}_{[0,T)}) = \prod_{t=0}^{T-1} P(\boldsymbol{r}_t|\boldsymbol{v}_t) = \prod_{t=0}^{T-1}\prod_{j=1}^{c} P(r_t^{(j)}|v_t^{(j)}), \tag{6.5}$$

where $P(r_t^{(j)}|v_t^{(j)})$ are the channel transition probabilities.

From the *a priori* probabilities of the information or code symbols it is easy to obtain the *a priori* probabilities for the branches of the trellis, $P(S_{t+1} = \boldsymbol{\sigma}'|S_t = \boldsymbol{\sigma})$ or $P(S_t = \boldsymbol{\sigma}|S_{t+1} = \boldsymbol{\sigma}')$. Consider a branch from state $\boldsymbol{\sigma}$ at time t to state $\boldsymbol{\sigma}'$ at time $t+1$. Let $\mathcal{U}(\boldsymbol{\sigma})$ and $\mathcal{V}(\boldsymbol{\sigma})$ be the set of information and code symbols, respectively, corresponding to the branches leaving state $\boldsymbol{\sigma}$ in the trellis. Naturally, $\mathcal{U}(\boldsymbol{\sigma})$ is the set of all binary b-tuples. Similarly, let $\mathcal{U}'(\boldsymbol{\sigma}')$ and $\mathcal{V}'(\boldsymbol{\sigma}')$ be the set of information and code symbols, respectively, corresponding to the branches leading to state $\boldsymbol{\sigma}'$ in the trellis. We will use rational systematic generator matrices with maximum a_0^c and a_0^{rc}. Then, it follows that $\mathcal{U}(\boldsymbol{\sigma}) = \mathcal{U}'(\boldsymbol{\sigma}')$ and $\mathcal{V}(\boldsymbol{\sigma}) = \mathcal{V}'(\boldsymbol{\sigma}')$. With the initial condition $P(S_0 = \boldsymbol{\sigma}) = 1/|\mathcal{S}|$, where $|\mathcal{S}|$ is the number of states, we have $P(S_t = \boldsymbol{\sigma}) = 1/|\mathcal{S}|$, $t = 1, 2, \ldots$, which leads to the conclusion

$$P(S_{t+1} = \boldsymbol{\sigma}'|S_t = \boldsymbol{\sigma}) = P(S_t = \boldsymbol{\sigma}|S_{t+1} = \boldsymbol{\sigma}'). \tag{6.6}$$

To simplify notation in the algorithm we introduce the following four variables:

$$\alpha_t(\boldsymbol{\sigma}) \triangleq P(S_t = \boldsymbol{\sigma}|\boldsymbol{r}_{[0,t)}), \tag{6.7a}$$

$$\beta_t(\boldsymbol{\sigma}) \triangleq P(S_t = \boldsymbol{\sigma}|\boldsymbol{r}_{[t,T)}), \tag{6.7b}$$

$$\gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \triangleq P(S_{t+1} = \boldsymbol{\sigma}'; \boldsymbol{r}_t|S_t = \boldsymbol{\sigma}), \tag{6.7c}$$

and

$$\delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \triangleq P(S_{t+1} = \boldsymbol{\sigma}'; S_t = \boldsymbol{\sigma}|\boldsymbol{r}_{[0,T)}). \tag{6.7d}$$

The first three will be used to calculate the forth, which can be used to get the *a posteriori* probabilities as

$$
\begin{aligned}
P(u_t^{(i)} = u|\boldsymbol{r}_{[0,T)}) &= \sum_{(\boldsymbol{\sigma},\boldsymbol{\sigma}'):\boldsymbol{\sigma}\xrightarrow{u}\boldsymbol{\sigma}'} P(S_t = \boldsymbol{\sigma}; S_{t+1} = \boldsymbol{\sigma}'|\boldsymbol{r}_{[0,T)}) \\
&= \sum_{(\boldsymbol{\sigma},\boldsymbol{\sigma}'):\boldsymbol{\sigma}\xrightarrow{u}\boldsymbol{\sigma}'} \delta_t(\boldsymbol{\sigma},\boldsymbol{\sigma}'),
\end{aligned} \tag{6.8}
$$

and

$$
\begin{aligned}
P(v_t^{(i)} = v|\boldsymbol{r}_{[0,T)}) &= \sum_{(\boldsymbol{\sigma},\boldsymbol{\sigma}'):\boldsymbol{\sigma}\xrightarrow{v}\boldsymbol{\sigma}'} P(S_t = \boldsymbol{\sigma}; S_{t+1} = \boldsymbol{\sigma}'|\boldsymbol{r}_{[0,T)}) \\
&= \sum_{(\boldsymbol{\sigma},\boldsymbol{\sigma}'):\boldsymbol{\sigma}\xrightarrow{v}\boldsymbol{\sigma}'} \delta_t(\boldsymbol{\sigma},\boldsymbol{\sigma}').
\end{aligned} \tag{6.9}
$$

The notation $(\boldsymbol{\sigma},\boldsymbol{\sigma}') : \boldsymbol{\sigma} \xrightarrow{u} \boldsymbol{\sigma}'$ means that $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ is taken from the set of state pairs such that there is a branch in the trellis from state $\boldsymbol{\sigma}$ at time t to state $\boldsymbol{\sigma}'$ at time $t+1$ with the ith bit of the corresponding information symbol equal to $u_t^{(i)} = u$. Similarly, $(\boldsymbol{\sigma},\boldsymbol{\sigma}') : \boldsymbol{\sigma} \xrightarrow{v} \boldsymbol{\sigma}'$ denotes the state pairs such that there is a branch in the trellis from state $\boldsymbol{\sigma}$ at time t to state $\boldsymbol{\sigma}'$ at time $t+1$ with the ith bit of the corresponding code symbol equal to $v_t^{(i)} = v$. We will also make use of notations like $\boldsymbol{\sigma} : \boldsymbol{\sigma} \to \boldsymbol{\sigma}'$ and $\boldsymbol{\sigma}' : \boldsymbol{\sigma} \to \boldsymbol{\sigma}'$, which means the set of states at time t leading to the state $\boldsymbol{\sigma}'$ at time $t+1$, and the set of states at time $t+1$ stemming from the state $\boldsymbol{\sigma}$ at time t, respectively.

The joint branch and received symbol probability, $\gamma_t(\boldsymbol{\sigma},\boldsymbol{\sigma}')$, can be calculated as

$$
\begin{aligned}
\gamma_t(\boldsymbol{\sigma},\boldsymbol{\sigma}') &= P(S_{t+1} = \boldsymbol{\sigma}'; \boldsymbol{r}_t|S_t = \boldsymbol{\sigma}) \\
&= P(\boldsymbol{r}_t|S_t = \boldsymbol{\sigma}; S_{t+1} = \boldsymbol{\sigma}')P(S_{t+1} = \boldsymbol{\sigma}'|S_t = \boldsymbol{\sigma}) \\
&= P(\boldsymbol{r}_t|v(\boldsymbol{\sigma},\boldsymbol{\sigma}'))P(S_{t+1} = \boldsymbol{\sigma}'|S_t = \boldsymbol{\sigma}),
\end{aligned} \tag{6.10}
$$

where $v(\boldsymbol{\sigma},\boldsymbol{\sigma}')$ is the code symbol corresponding to the trellis branch from state $\boldsymbol{\sigma}$ at time t to state $\boldsymbol{\sigma}'$ at time $t+1$.

If, at time t, the values $\alpha_t(\boldsymbol{\sigma})$, $\forall\boldsymbol{\sigma} \in \mathcal{S}$ are known we can derive the

corresponding values at time $t+1$ as

$$
\begin{aligned}
\alpha_{t+1}(\boldsymbol{\sigma}') &= P(S_{t+1} = \boldsymbol{\sigma}' | \boldsymbol{r}_{[0,t+1)}) \\
&= \sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} P(S_{t+1} = \boldsymbol{\sigma}'; S_t = \boldsymbol{\sigma} | \boldsymbol{r}_{[0,t)}; \boldsymbol{r}_t) \\
&= \frac{1}{P(\boldsymbol{r}_t | \boldsymbol{r}_{[0,t)})} \sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} P(S_{t+1} = \boldsymbol{\sigma}'; S_t = \boldsymbol{\sigma}; \boldsymbol{r}_t | \boldsymbol{r}_{[0,t)}) \\
&= k_{t+1}^{\alpha} \sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} P(S_t = \boldsymbol{\sigma} | \boldsymbol{r}_{[0,t)}) P(S_{t+1} = \boldsymbol{\sigma}'; \boldsymbol{r}_t | S_t = \boldsymbol{\sigma}) \\
&= k_{t+1}^{\alpha} \sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} \alpha_t(\boldsymbol{\sigma}) \gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}').
\end{aligned} \tag{6.11}
$$

In the second last equality we used that given the state at time t the continuation does not depend on the past received sequence, $\boldsymbol{r}_{[0,t)}$ i.e., the state at time t represents all that is known to the encoder about the past. The constant term, $k_{t+1}^{\alpha} = 1/P(\boldsymbol{r}_t | \boldsymbol{r}_{[0,t)})$, can be determined by the relation

$$
\sum_{\boldsymbol{\sigma}' \in \mathcal{S}} \alpha_{t+1}(\boldsymbol{\sigma}') = 1. \tag{6.12}
$$

The derivation of the α-values is sometimes referred to as the forward recursion. Similarly, there is a backward recursion for deriving the β-values. If, at time $t+1$, the values $\beta_{t+1}(\boldsymbol{\sigma}')$, $\forall \boldsymbol{\sigma}' \in \mathcal{S}$ are known we can derive the corresponding values at time t as

$$
\begin{aligned}
\beta_t(\boldsymbol{\sigma}) &= P(S_t = \boldsymbol{\sigma} | \boldsymbol{r}_{[t,T)}) \\
&= \sum_{\boldsymbol{\sigma}':\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} P(S_t = \boldsymbol{\sigma}; S_{t+1} = \boldsymbol{\sigma}' | \boldsymbol{r}_t; \boldsymbol{r}_{[t+1,T)}) \\
&= \frac{1}{P(\boldsymbol{r}_t | \boldsymbol{r}_{[t+1,T)})} \sum_{\boldsymbol{\sigma}':\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} P(S_t = \boldsymbol{\sigma}; S_{t+1} = \boldsymbol{\sigma}'; \boldsymbol{r}_t | \boldsymbol{r}_{[t+1,T)}) \\
&= k_t^{\beta} \sum_{\boldsymbol{\sigma}':\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} P(S_{t+1} = \boldsymbol{\sigma}' | \boldsymbol{r}_{[t+1,T)}) P(S_{t+1} = \boldsymbol{\sigma}'; \boldsymbol{r}_t | S_t = \boldsymbol{\sigma}) \\
&= k_t^{\beta} \sum_{\boldsymbol{\sigma}':\boldsymbol{\sigma}\to\boldsymbol{\sigma}'} \beta_{t+1}(\boldsymbol{\sigma}') \gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}').
\end{aligned} \tag{6.13}
$$

In the second last equality we used that the state at time $t+1$ represents the future, and given this the received sequence from time $t+1$ and forward can be dropped. The constant term $k_t^{\beta} = 1/P(\boldsymbol{r}_t | \boldsymbol{r}_{[t+1,T)})$ can be determined by the sum

$$
\sum_{\boldsymbol{\sigma} \in \mathcal{S}} \beta_t(\boldsymbol{\sigma}) = 1. \tag{6.14}
$$

For the branch defined by the states $(S_t, S_{t+1}) = (\boldsymbol{\sigma}, \boldsymbol{\sigma}')$ $\alpha_t(\boldsymbol{\sigma})$ represents the *a posteriori* information of the sequences up to time t and $\beta_{t+1}(\boldsymbol{\sigma}')$ from time $t+1$ and forward. The information about the branch between the states is found in $\gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$. Collecting these things together results in

(6.15)
$$\begin{aligned}
\delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') &= P(S_t = \boldsymbol{\sigma}; S_{t+1} = \boldsymbol{\sigma}' | \boldsymbol{r}_{[0,T)}) \\
&= \frac{1}{P(\boldsymbol{r}_{[0,T)})} P(S_t = \boldsymbol{\sigma}; \boldsymbol{r}_{[0,t)}) P(S_{t+1} = \boldsymbol{\sigma}'; \boldsymbol{r}_{[t,T)} | S_t = \boldsymbol{\sigma}) \\
&= \frac{P(\boldsymbol{r}_{[0,t)})}{P(\boldsymbol{r}_{[0,T)})} \alpha_t(\boldsymbol{\sigma}) P(S_{t+1} = \boldsymbol{\sigma}'; \boldsymbol{r}_t | S_t = \boldsymbol{\sigma}) P(\boldsymbol{r}_{[t+1,T)} | S_{t+1} = \boldsymbol{\sigma}') \\
&= k_t^\delta \alpha_t(\boldsymbol{\sigma}) \gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \beta_{t+1}(\boldsymbol{\sigma}'),
\end{aligned}$$

where the constant term

$$k_t^\delta = |\mathcal{S}| \frac{P(\boldsymbol{r}_{[0,t)}) P(\boldsymbol{r}_{[t+1,T)})}{P(\boldsymbol{r}_{[0,T)})} \tag{6.16}$$

can be obtained by the relation

$$\sum_{(\boldsymbol{\sigma}, \boldsymbol{\sigma}') : \boldsymbol{\sigma} \to \boldsymbol{\sigma}'} \delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') = 1. \tag{6.17}$$

In the second equality of (6.15) we used that the past is represented by the state at time t, and in the third equality that the future is represented by the state at time $t+1$.

In Algorithm 6.1 we summarize the above calculations.

So far we have only considered the algorithm for truncated convolutional codes. If the frames are long it is desirable to have a windowed version. This would also permit us to use infinite sequences, which is a natural way of viewing convolutional codes.

In Figure 6.2 we have a sketch of a section of a trellis where the times T_1, T_2, T_3, and T_4 are marked. Assume that the *a posteriori* probabilities are calculated up to time T_1 and the α-values to time T_2. Continue the forward recursion from time T_2 to time T_4. Now we would like to start the backward recursion from time T_4 to time T_1, but we do not know the distribution of the β-values that starts the recursion. However, the distribution of the α-values and the distribution of the β-values are related. Thus, initialize the beta values at time T_4 by the corresponding alpha values,

$$\beta_{T_4}(\boldsymbol{\sigma}) = \alpha_{T_4}(\boldsymbol{\sigma}), \quad \forall \boldsymbol{\sigma} \in \mathcal{S}, \tag{6.18}$$

and start the backward recursion. After a couple of steps the β-values will approach the values we would have had if the recursion started at the end of

Initialization:

$\alpha_0(\mathbf{0}) \leftarrow 1$; $\beta_T(\mathbf{0}) \leftarrow 1$

$\gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \leftarrow P(\boldsymbol{r}_t | v(\boldsymbol{\sigma}, \boldsymbol{\sigma}')) P(S_{t+1} = \boldsymbol{\sigma}' | S_t = \boldsymbol{\sigma})$,
$0 \leqslant t < T$; $(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \in \mathcal{S} \times \mathcal{S}$

Forward recursion: t from 0 to $T-1$; $\boldsymbol{\sigma} \in \mathcal{S}$;

$\alpha_{t+1}(\boldsymbol{\sigma}') \leftarrow k_{t+1}^{\alpha} \sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma} \to \boldsymbol{\sigma}'} \alpha_t(\boldsymbol{\sigma}) \gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$

Backward recursion: t from $T-1$ to 0; $\boldsymbol{\sigma} \in \mathcal{S}$;

$\beta_t(\boldsymbol{\sigma}) \leftarrow k_t^{\beta} \sum_{\boldsymbol{\sigma}':\boldsymbol{\sigma} \to \boldsymbol{\sigma}'} \beta_{t+1}(\boldsymbol{\sigma}') \gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$

Gather information: $0 \leqslant t < T$; $(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \in \mathcal{S} \times \mathcal{S}$;

$\delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \leftarrow k_t^{\sigma} \alpha_t(\boldsymbol{\sigma}) \gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \beta_{t+1}(\boldsymbol{\sigma}')$

Get *a posteriori* information $0 \leqslant t < T$

$P(u_t^{(j)} = u | \boldsymbol{r}_{[0,T)}) \leftarrow \sum_{(\boldsymbol{\sigma},\boldsymbol{\sigma}'):\boldsymbol{\sigma} \xrightarrow{u} \boldsymbol{\sigma}'} \delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$, $1 \leqslant j \leqslant b$

$P(v_t^{(j)} = v | \boldsymbol{r}_{[0,T)}) \leftarrow \sum_{(\boldsymbol{\sigma},\boldsymbol{\sigma}'):\boldsymbol{\sigma} \xrightarrow{v} \boldsymbol{\sigma}'} \delta_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$, $1 \leqslant j \leqslant c$

Algorithm 6.1: The BCJR algorithm.

the trellis. How many steps it takes for the recursion to converge depends on a number of factors, e.g., how close the distributions should be, the rate and memory of the code, and the channel. For relatively good channels and the codes considered in this chapter it seems like six or seven times the memory is enough. In Figure 6.2 the interval between T_3 and T_4 lets the β-vales converge. After the backward recursion we can calculate the δ-values and the corresponding *a posteriori* probabilities from time T_1 to time T_3. To continue add $T_3 - T_1$ to the values T_1, T_2, T_3, and T_4 and start all over.

Let W_D be the *decision window*, the length of the part where decisions are made, i.e., $T_3 - T_1$, and W_B the *backtrack window*, the length needed for the β-values to converge, i.e., $T_4 - T_3$. Then we can write the windowed BCJR algorithm as in Algorithm 6.2.

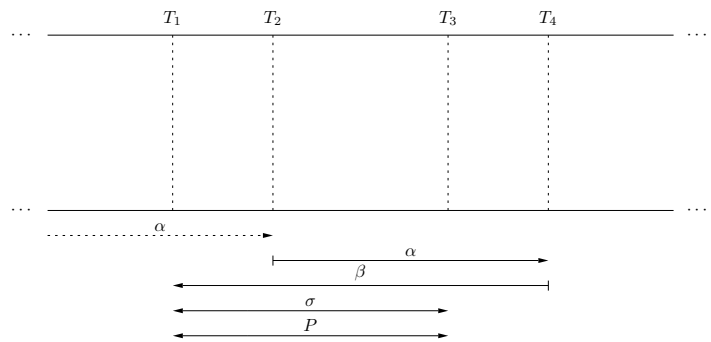

Figure 6.2: The section of a trellis in which the windowed BCJR algorithm works.

Initialization:
- $T_1 \leftarrow 0$; $T_2 \leftarrow 0$
- $T_3 \leftarrow W_D$
- $T_3 \leftarrow W_D + W_B$

while (not end of trellis)
- **APP-algorithm:**
  - $\gamma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$: $T_2 \leqslant t \leqslant T_4$
  - $\alpha_t(\boldsymbol{\sigma})$: for t from T_2 to T_4
  - $\beta_{T_4}(\boldsymbol{\sigma}) \leftarrow \alpha_{T_4}(\boldsymbol{\sigma})$: $\boldsymbol{\sigma} \in \mathcal{S}$
  - $\beta_t(\boldsymbol{\sigma})$: for t from T_4 to T_1
  - $\sigma_t(\boldsymbol{\sigma}, \boldsymbol{\sigma}')$: $T_1 \leqslant t \leqslant T_3$
  - $P(v_t^{(i)}|\boldsymbol{r})$: $T_1 \leqslant t \leqslant T_3$
- **Update T:**
  - $T_1^{(\text{new})} \leftarrow T_3$
  - $T_2^{(\text{new})} \leftarrow T_3 + W_B$
  - $T_3^{(\text{new})} \leftarrow T_3 + W_D$
  - $T_4^{(\text{new})} \leftarrow T_4 + W_D$

end while

Algorithm 6.2: A windowed version of the BCJR algorithm.

6.2. Iterative Decoding

The idea behind iterative decoding is that information retrieved by one decoder can be used by the next in a chain of successive decoders [20]. The decoder for a woven convolutional code consists of two blocks of decoders, one for the inner warp and one for the outer warp. The two warps have the intermediate sequence in common, i.e., the code sequence for the outer warp is used as information sequence for the inner warp. Therefore, the *a posteriori* information about the inner information sequence from the decoder of the inner warp should be used as *a priori* information about the the outer code sequence for the decoder of the outer warp. Then we get an estimate of the *a posteriori* probability for the outer information sequence.

From the decoder for the outer warp we can also get *a posteriori* information about the outer code sequence. This can be used as *a priori* information about the inner information sequence by the decoder for the inner warp. That is the start of the second iteration. Continuing, each new iteration will give an estimate, hopefully improved, of the *a posteriori* probability for the information sequence for the woven convolutional code.

Denote by $\boldsymbol{r}_{\mathcal{I}\setminus_t^{(j)}}$ the received sequence where the ith channel symbol of the c-tuple at time t is excluded, i.e.,

$$\boldsymbol{r}_{\mathcal{I}\setminus_t^{(j)}} = \left(\boldsymbol{r}_0 \ldots \boldsymbol{r}_{t-1}\ (r_t^{(1)} \ldots r_t^{(j-1)}\ r_t^{(j+1)} \ldots r_t^{(c)})\ \boldsymbol{r}_{t+1} \ldots\right). \tag{6.19}$$

Then the *a posteriori* probability can be split as

$$\begin{aligned} P(v_t^{(j)}|\boldsymbol{r}) &= \frac{1}{P(\boldsymbol{r})} P(v_t^{(j)}; r_t^{(j)}; \boldsymbol{r}_{\mathcal{I}\setminus_t^{(j)}}) \\ &= \frac{1}{P(\boldsymbol{r})} P(r_t^{(j)}; v_t^{(j)}) P(\boldsymbol{r}_{\mathcal{I}\setminus_t^{(j)}} | r_t^{(j)}; v_t^{(j)}) \\ &= \frac{1}{P(\boldsymbol{r})} P(v_t^{(j)}) P(r_t^{(j)}|v_t^{(j)}) P(\boldsymbol{r}_{\mathcal{I}\setminus_t^{(j)}} | v_t^{(j)}). \end{aligned} \tag{6.20}$$

Both the *a priori* probability, $P(v_t^{(j)})$, and the channel transition probability, $P(r_t^{(j)}|v_t^{(j)})$, serve as inputs for the algorithm. The last factor of (6.20) is called the *extrinsic information* and can be derived from the *a posteriori* probability as

$$P_{\text{ext}}(v_t^{(j)}) = P(\boldsymbol{r}_{\mathcal{I}\setminus_t^{(j)}} | v_t^{(j)}) = P(\boldsymbol{r}) \cdot \frac{P(v_t^{(j)}|\boldsymbol{r})}{P(v_t^{(j)}) P(r_t^{(j)}|v_t^{(j)})}, \tag{6.21}$$

where $P(\boldsymbol{r})$ can be derived by normalization. It is the extrinsic information that should be sent as *a priori* information to the next decoder in the chain. It represents the change in *a posteriori* information obtained by the decoder.

In Figure 6.3 we show the APP-module that is used in an iterative decoding scheme.

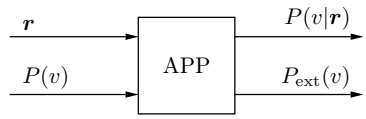

Figure 6.3: The APP-module with outputs for both *a posteriori* probability and extrinsic information.

Assume that all encoders in the two warps are systematic. Then from the received sequence we can easily extract the part that corresponds to the received sequence for the outer warp. That means that both the decoder for the outer warp and the decoder for the inner warp will have access to the corresponding received sequences and *a priori* information. Since the encoders of a warp are independent, the corresponding decoder can be built as a warp of APP-modules. In Figure 6.4 we show a block diagram of a decoder for the twill. To start the iterations initialize the *a priori* probabilities to be equally distributed, i.e., $P(u_t^{i(j)} = 0) = P(u_t^{i(j)} = 1) = 1/2$.

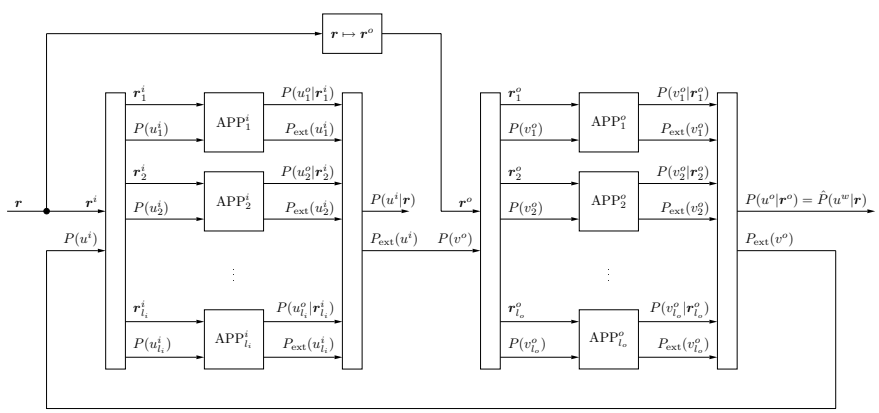

Figure 6.4: Block diagram for a decoder of a twill.

The decoder in Figure 6.4 only works when the transmitted data are divided into frames. If the data are transmitted as an infinite sequence (or in very long frames) we cannot wait until the decoders are finished before starting the next iteration. Instead we can use the windowed BCJR algorithm and send the extrinsic information from the decoders for the outer warp to a new, identical, set of decoders as the one in the figure. These decoders will, of course, work on the same received sequence, only a bit delayed compared with the first decoders. In such pipelined manner we get a decoder like the one in Figure 6.4 for each level of iterations.

In the simulations the data have been divided into frames. However, we still need the windowed algorithm. Consider a woven convolutional code with outer warp, where the warp consist of 100 parallel encoders. We will use outer encoders with rate $R_o = 2/3$ and inner encoders with rate $R_i = 1/2$. If the frame length for each of the outer encoders is 300 information symbols, the frame length for the inner encoder will be about $300 \cdot 3 \cdot 100 = 90000$ information symbols. Such frames are too long to be handled by the BCJR algorithm for truncated convolutional codes.

6.3. Simulation Results

In the simulated communication system we have used an eight level (three bit) quantized AWGN channel. The quantization is such that the cut-off rate, R_0, is maximized [40]. For such channels the capacity equals 1/3 for the signal to noise ratio $[E_b/N_0]_C = -0.37$dB, and the cut-off rate equals 1/3 for $[E_b/N_0]_{R_0} = 2.2$dB. In the results presented here the outer encoders are of rate $R_o = 2/3$ and have the generator matrix

$$G^o(D) = \begin{pmatrix} 1 & 0 & \frac{1+D^2+D^3}{1+D^3} \\ 0 & 1 & \frac{1+D+D^3}{1+D^3} \end{pmatrix} \tag{6.22}$$

with overall constraint length $\nu_o = 6$ and memory $m_o = 3$. The corresponding code has free distance $d^o_{\text{free}} = 4$. The generator matrix for the rate $R_i = 1/2$ inner encoders,

$$G^i(D) = \begin{pmatrix} 1 & \frac{1+D+D^2+D^3}{1+D+D^3} \end{pmatrix}, \tag{6.23}$$

has the overall constraint length and memory $\nu_i = m_i = 3$ and generates free distance $d^i_{\text{free}} = 6$. The woven convolutional code is of rate $R_w = 1/3$ and if l_o or l_i are large enough its free distance satisfies $d^{tw}_{\text{free}} \geqslant d^o_{\text{free}} d^i_{\text{free}} = 24$.

Since the generator matrices are systematic they are minimal. Furthermore, the inner generator matrix is canonical and, therefore, the realization on controller canonical form with three delay elements is a minimal encoder.

A realization on observer canonical form of the outer generator matrix is a minimal encoder with three delay elements, see Figure 2.3.

The length of the frames used in the outer encoders is about 300 symbols, including the zero state driving sequence. This, together with the numbers of encoders, l_o and l_i, determines the frame length for the inner encoders. Given the frame length for the outer encoders, FL_o, the frame length for the inner encoders is

$$\mathrm{FL}_i = \mathrm{FL}_o \frac{l_o c_o}{l_i b_i} + m_i. \tag{6.24}$$

Likewise, the outer frame length can be derived from the inner frame length as

$$\mathrm{FL}_o = (\mathrm{FL}_i - m_i) \frac{l_i b_i}{l_o c_o}. \tag{6.25}$$

Since FL_o and FL_i are integers we have to fulfill the following conditions

$$\begin{cases} l_o c_o \mid (\mathrm{FL}_i - m_i) l_i b_i, \\ l_i b_i \mid \mathrm{FL}_o l_o c_o. \end{cases} \tag{6.26}$$

Our first simulation is for a woven convolutional code with outer warp. For the chosen generator matrices we have $l_i b_i = 1$ and the outer frame length can be set to $\mathrm{FL}_o = 300$. Let the number of encoders in the warp be $l_o = 30$, then the inner frame length is $\mathrm{FL}_i = 27003$. In Figure 6.5 we show the bit error rate (BER) versus the signal to noise ratio (SNR), E_b/N_0, in dB for the first ten iterations. The curves tend to flatten out after a while, i.e., there is a bend on the curves at about 1.6dB, and the improvement for each iteration decrease. The bend is probably due to the relatively low free distance, $d^{ow}_{\text{free}} = d^{o}_{\text{free}} d^{i}_{\text{free}} = 24$ (equality since the encoders of the warp are identical and that there is only one warp).

In Figure 6.6 the number of encoders in the warp is varied. The curves show the BER after ten iterations when $l_o = 1$, $l_o = 10$, $l_o = 30$, and $l_o = 100$. The result for the cascaded convolutional code ($l_o = 1$) is, as expected, poor, but already with a warp of ten encoders the curve is significantly better. The improvements for each added encoder diminish fast though, and the small difference between the curves for $l_o = 30$ and $l_o = 100$ is probably due to inaccuracies in the results.

To improve the performance we include an extra interleaver in the scheme. Consider again Figure 5.1 where we have the encoder for a woven convolutional code with outer warp. The code sequences from the outer encoders are, as before, written row-wise into a buffer. Before reading a column its bits are permuted, i.e., the bits in the buffer are permuted within each column before the buffer is read column-wise to form the information sequence

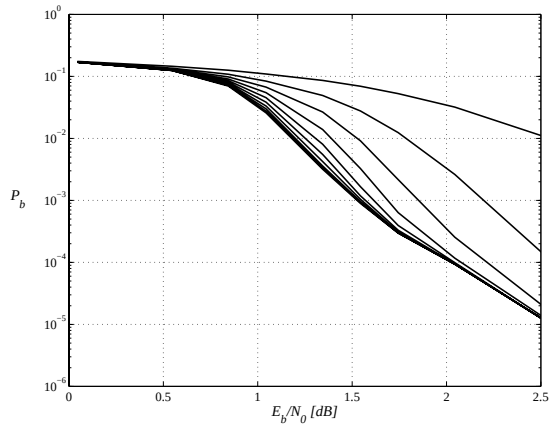

Figure 6.5: BER versus SNR for a woven convolutional code with outer warp, where $l_o = 30$ and the generator matrices are defined by (6.22) and (6.23).

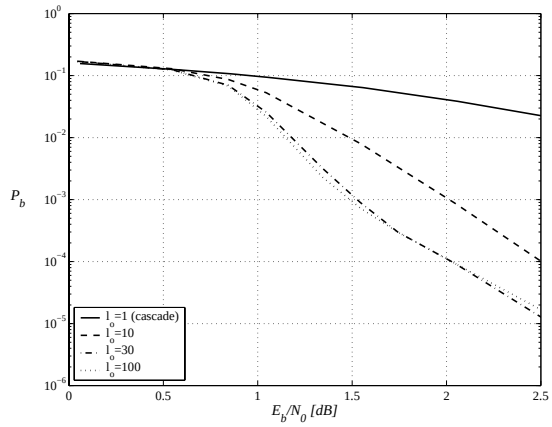

Figure 6.6: BER for a woven convolutional code with outer warp, when l_o is varied.

for the inner encoder. It is important that not all columns have the same permutations.

There are a number of interleavers to choose from. The one used here is taken from [1]. Let the number of outer encoders be such that $l_o + 1$ is a prime number, and let α be a primitive element in the field $\mathbb{F}_{l_o+1}$. The positions of the kth column are then permuted according to the rule

$$i^+ = i\alpha^k \qquad \bmod l_o + 1, \tag{6.27}$$

where i, $1 \leqslant i \leqslant l_o$ are the positions and i^+ the permuted positions. In Figure 6.7 we show an encoder for a woven convolutional code with outer warp and column-wise permutations.

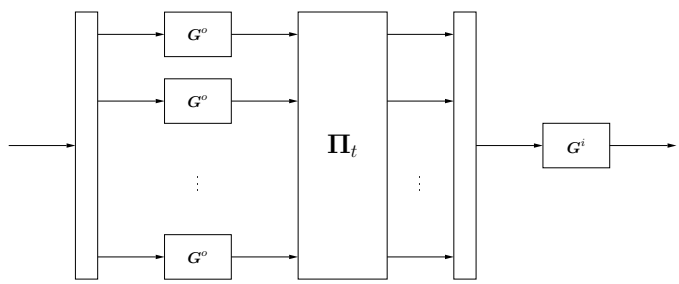

Figure 6.7: An encoder for a woven convolutional code with outer warp and column-wise permutations.

For $l_o = 10$ with $\alpha = 6$, $l_o = 30$ with $\alpha = 11$, and $l_o = 100$ with $\alpha = 50$ the BER for the scheme with column-wise permutations are shown in Figure 6.8. For $l_o = 10$ there is not much improvement compared to the scheme without permutations (Figure 6.6), but for $l_o = 30$ and $l_o = 100$, the results are considerably better. We do not have the bend of the curves anymore, at least not in the plotted intervals, which means that we have increased the free distance.

Next we will consider the twill with the same number of encoders in the outer warp, viz., $l_o = 10$, $l_o = 30$, and $l_o = 100$. To have the number of outer and inner encoders relatively prime let $l_i = l_o - 1$. From (6.24) and (6.25) we get

$$\begin{cases} \mathrm{FL}_o = 306 \text{ and } \mathrm{FL}_i = 1023, & \text{if } l_o = 10 \\ \mathrm{FL}_o = 319 \text{ and } \mathrm{FL}_i = 993, & \text{if } l_o = 30 \\ \mathrm{FL}_o = 297 \text{ and } \mathrm{FL}_i = 903, & \text{if } l_o = 100. \end{cases} \tag{6.28}$$

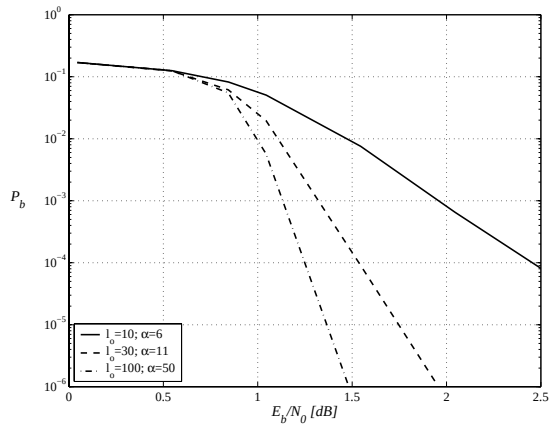

Figure 6.8: BER for a woven convolutional code with outer warp and column-wise permutations for different l_o.

The BER after ten iterations for various l_o are shown in Figure 6.9. We still have a flattening due to the relatively low free distance. Compared to the curves for the woven convolutional code with outer warp in Figure 6.6 the bend is lowered. An explanation could be that the free distance for the twill is greater than the product of the outer and inner free distances. It could also be that the number of paths corresponding to the free distance has decreased due to the extra interleaving effect in the twill.

For woven convolutional codes with outer warp the results were improved significantly when an interleaver was inserted. To obtain the same for the twill the permutation scheme has to be modified a bit. Viewed over l_i columns of the buffer, the input bits for each of the inner encoders are taken one from each row, i.e., one from each of the l_o output sequences of the outer encoders. Thus, the l_i sequences of l_o bits read from an $l_o \times l_i$ block of the buffer correspond to the columns in a buffer for a woven convolutional code with outer warp. For the block that starts at column k and for the sequence corresponding to the lth inner encoder apply the permutation rule

$$i^+ = i\alpha^{k+l-1} \qquad \bmod l_o + 1. \tag{6.29}$$

We will still call it column-wise permutations since for $l_i = 1$ it will be equivalent to (6.27). With the extra permutations we get the results shown in Figure 6.10. Again we see that the bend has disappeared within the considered intervals, and by the permutations we have increased the free distance.

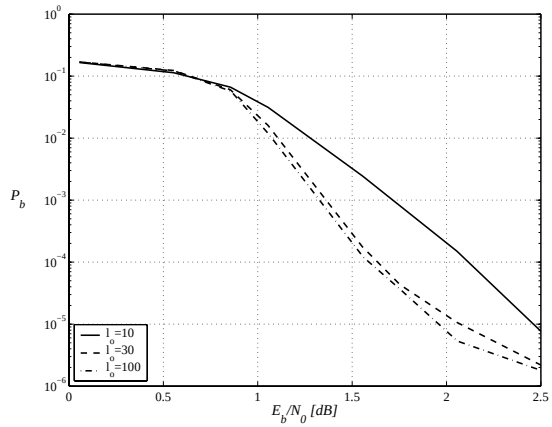

Figure 6.9: BER for a twill with $l_i = l_o - 1$ and different l_o.

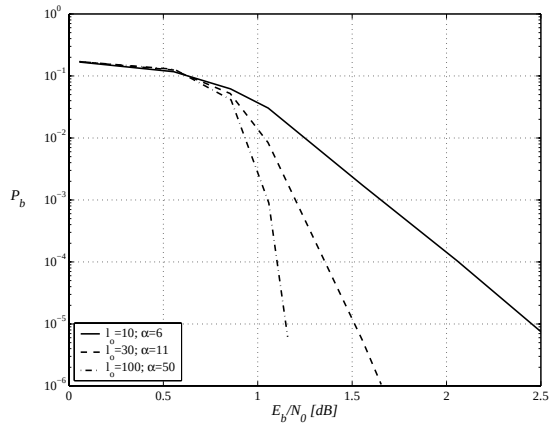

Figure 6.10: BER for a twill with column-wise permutations. The number of encoders in the outer warp l_o varies while $l_i = l_o - 1$.

In Figure 6.11 we have collected the four different cases (woven convolutional codes with outer warp with and without permutations and the twill with and without permutations) when $l_o = 30$.

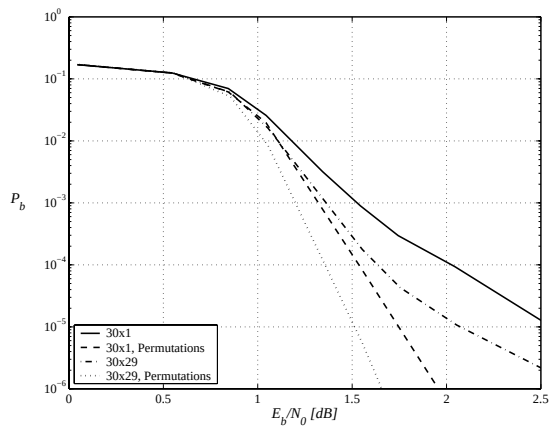

Figure 6.11: BER for the four schemes of woven convolutional codes when $l_o = 30$.

We conclude this section with a plot, Figure 6.12, showing the results for a twill with column-wise permutations and $l_o = 30$, where the memories of the constituent encoders are varied. The new generator matrices are selected from the same tables as (6.22) and (6.23), i.e., from [33, 42]. These tables list polynomial generator matrices for convolutional codes with maximum free distance. We use systematic generator matrices equivalent to the ones listed. For $m_o = m_i = 2$ we have the generator matrices

$$G^o(D) = \begin{pmatrix} 1 & 0 & \frac{1}{1+D+D^2} \\ 0 & 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix} \tag{6.30}$$

and

$$G^i(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}, \tag{6.31}$$

and for $m_o = m_i = 4$ we have

$$G^o(D) = \begin{pmatrix} 1 & 0 & \frac{1+D^3+D^4}{1+D+D^4} \\ 0 & 1 & \frac{1+D+D^2+D^4}{1+D+D^4} \end{pmatrix} \tag{6.32}$$

and

$$G^i(D) = \begin{pmatrix} 1 & \frac{1+D^2+D^3+D^4}{1+D+D^4} \end{pmatrix}. \tag{6.33}$$

We still get minimal encoders by realizing the outer generator matrices on observer canonical form and the inner on controller canonical form.

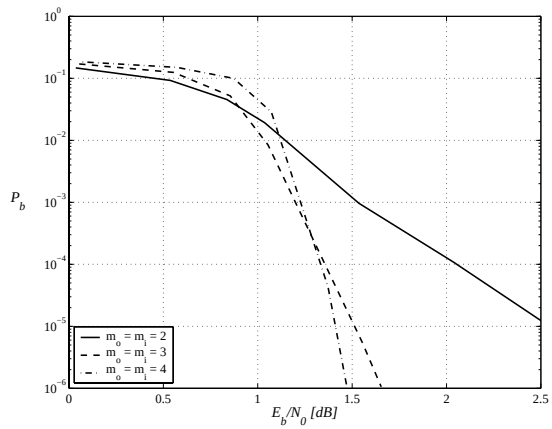

Figure 6.12: The resulting BER for twills with different constituent memories.

7

Bounds on Woven Convolutional Codes

In the previous chapters we have defined the woven convolutional codes. First the encoder properties were examined and then an iterative scheme for decoding was defined and simulated. The structure of woven convolutional codes with outer and inner warp, respectively, are well suited for analytical examination. In the papers [28, 48, 49] the error exponents and bounds on the active distances were derived for woven convolutional codes. This chapter gives a survey over these bounds. The bounds are given without proofs.

The first two sections are devoted to error exponents. In Section 7.1 a compact review of the error exponents for block and convolutional codes is given, while in Section 7.2 we give the error exponents for woven convolutional code first with outer and then with inner warp. In Section 7.3 lower bounds on the active distances for woven convolutional codes with both outer and inner warp are given.

7.1. Error Exponents

To formulate the theorems on error exponents for block and convolutional codes we first need to define three parameters from information theory. Firstly, the cut-off rate, R_0, for the binary symmetric channel is

$$R_0 \triangleq 1 - \log\left(1 + 2\sqrt{p(1-p)}\right), \tag{7.1}$$

where p is the channel crossover probability. Similarly, the expurgation rate and critical rate are defined as

$$R_{\text{ex}} \triangleq 1 - h\left(\frac{2\sqrt{p(1-p)}}{1 + 2\sqrt{p(1-p)}}\right) \tag{7.2}$$

and

$$R_{\text{cr}} \triangleq 1 - h\left(\frac{\sqrt{p}}{\sqrt{p} + \sqrt{1-p}}\right), \tag{7.3}$$

respectively, where $h(\cdot)$ is the binary entropy function.

We are now ready to state a well-known theorem about the error probability for block codes [18].

Theorem 7.1 [Gallager bound]: There exists a binary block code $\mathcal{B}$ with rate R and block length N such that when used to communicate over a binary symmetric channel together with maximum likelihood decoding, the error probability is upper bounded by

$$P_E \leqslant 2^{-(E_{\mathcal{B}}(R)+o(1))N}, \tag{7.4}$$

where the error exponent for block codes is defined as

$$E_{\mathcal{B}}(r) \triangleq \begin{cases} -\delta_{GV} \log\left(2\sqrt{p(1-p)}\right), & 0 \leqslant R < R_{ex}, \\ R_0 - R, & R_{ex} \leqslant R < R_{cr}, \\ \delta_{GV} \log \frac{\delta_{GV}}{p} + (1-\delta_{GV}) \log \frac{1-\delta_{GV}}{1-p}, & R_{cr} \leqslant R < C, \end{cases} \tag{7.5}$$

where $\delta_{GV} = h^{-1}(1-R)$ is the normalized Gilbert-Varshamov bound on the minimum distance for block codes and C the channel capacity. □

From the error exponent for block codes we can derive the error exponent for convolutional codes [35].

Theorem 7.2 [Yudkin bound]: There exists a convolutional code of rate $R = b/c$, encoded by a polynomial, periodically time-varying generator matrix with memory m and period T, such that when used to communicate over a binary symmetric channel together with maximum likelihood decoding the burst error probability is upper bounded by

$$P_B \leqslant 2^{-(E_{\mathcal{C}}(R)+o(1))mc}, \tag{7.6}$$

where the error exponent for convolutional codes is defined as

$$E_{\mathcal{C}}(R) \triangleq -\delta_C(R)\log(2\sqrt{p(1-p)}), \quad 0 < R < R_0 \tag{7.7a}$$

and

$$\begin{cases} E_{\mathcal{C}}(R) = G(s), & 0 \leqslant s \leqslant 1 \\ R = \frac{G(s)}{s}, & R_0 \leqslant R < C, \end{cases} \tag{7.7b}$$

where $\delta_C(R)$ is the main term of the normalized Costello bound on the free distance (2.38) and

$$G(s) = s - (1+s)\log\left(p^{\frac{1}{1+s}} + (1-p)^{\frac{1}{1+s}}\right) \tag{7.8}$$

is the so-called Gallager function. □

7.2. Error Exponents of Woven Convolutional Codes

Consider the case when the codewords of a woven convolutional code with outer warp are transmitted over a binary symmetric channel (BSC). The decoder first carry out the hard decisions Viterbi decoding of the inner code. Then, the estimated information symbols are fed into the corresponding l_o parallel outer decoders, see Figure 7.1, where we again use hard decisions Viterbi decoding. The estimated information symbols of all outer codes are delivered as the output of the woven communication system.

In [49] the error exponent for this construction was derived.

Theorem 7.3: There exists a woven convolutional code with outer warp of rate $R_{ow} = R_o R_i$, encoded by l_o outer rate $R_o = b_o/c_o$ periodically time-varying generator matrices with memory m_o and period T and one inner rate $R_i = b_i/c_i$ periodically time-varying generator matrix with memory m_i and period T, such that when used to communicate over a binary symmetric channel together with maximum likelihood decoding the burst error probability is upper bounded by

$$P_B \leqslant 2^{-(E_{ow}(R_{ow})+o(1))\ell_o m_o m_i c_o c_i}, \tag{7.9}$$

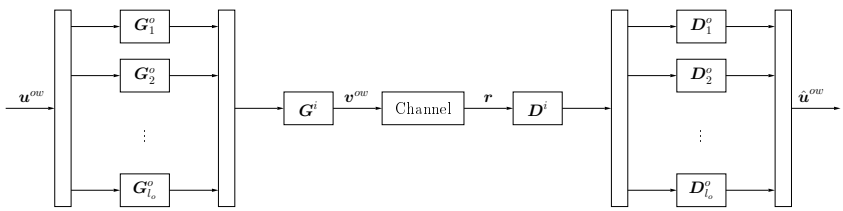

Figure 7.1: Block diagram of a communication system using woven convolutional coded with outer warp.

where the error exponent for woven convolutional codes with outer warp is defined as

$$E_{ow}(R_{ow}) \triangleq \max_{R_i} \left\{ \frac{1}{2\ell_o} E_{\mathcal{C}}(R_i) \delta_C \left(\frac{R_{ow}}{R_i} \right) \right\}, \tag{7.10}$$

and where $\ell_o = l_o/m_i b_i$ is the normalized number of outer encoders and $\delta_C(R)$ the main term of the normalized Costello bound on the free distance (2.38). □

The error exponents $E_{\mathcal{B}}(R)$, $E_{\mathcal{C}}(R)$, and $E_{ow}(R)$ are all given in Figure 7.2 for the binary symmetric channel with crossover probability $p = 0.01$.

The error probability for woven convolutional codes with outer warp decreases exponentially with $\mu c_o c_i \ell_o \triangleq m_o m_i c_o c_i \ell_o$ and the error exponent defined by (7.10). If the decoder of the woven convolutional code with outer warp consists of Viterbi decoders the decoding complexity is proportional to

$$\Gamma \approx 2^{m_i} + l_o 2^{m_o}. \tag{7.11}$$

Choose $m_i = m_o = \sqrt{\mu}$. Then, asymptotically, the decoding complexity increases only as $2^{\sqrt{\mu}}$.

For woven convolutional codes with inner warp, Zyablov, Shavgulidze, and Johannessson showed [48] that the corresponding error exponent is defined as

$$E_{iw}(R_{iw}) \triangleq \max_{R_i} \left\{ \frac{1}{2} E_{\mathcal{C}}(R_i) \delta_{GV} \left(\frac{R_{iw}}{R_i} \right) \right\}, \tag{7.12}$$

where $\delta_{GV}(R) = h^{-1}(1-R)$ is the normalized Gilbert-Varshamov bound on the minimum distance for block codes. Thus, there exists a woven convolutional code with inner warp such that the burst error probability is bounded

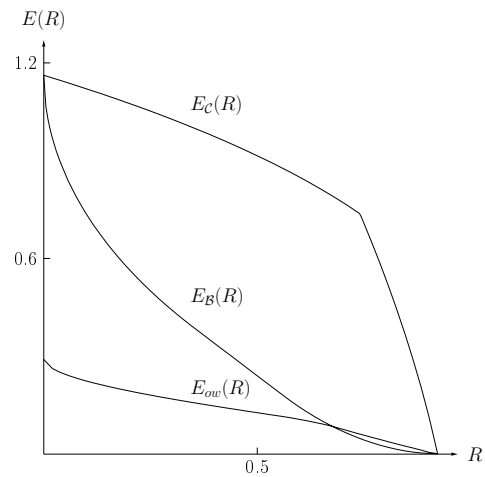

Figure 7.2: Error exponents for block codes, $E_{\mathcal{B}}(R)$, convolutional codes, $E_{\mathcal{C}}(R)$, and woven convolutional codes with outer warp, $E_{ow}(R)$.

by

$$P_B \leqslant 2^{-(E_{iw}(R_{iw})+o(1))\ell_i m_o m_i c_o c_i}, \tag{7.13}$$

where $\ell_i = l_i/m_o c_o$.

When $\ell_o = 1$, the error exponent for woven convolutional codes with inner warp is slightly better than that of woven convolutional codes with outer warp.

7.3. Bounds on the Active Distances

In [28] lower bounds on the active distances for woven convolutional codes with outer and inner warps were presented. Those are derived from the bounds in Theorem 3.19 together with Theorem 5.9 and Theorem 5.12.

Introduce the normalized length

$$\ell = \frac{j+1}{m_o}. \tag{7.14}$$

Then, the asymptotic ($m_o \to \infty$ and $m_i \to \infty$ and, hence, $m_{ow} \to \infty$) lower bounds on the active distances normalized by $m_{ow}c_{ow}$ is given by the following theorem.

Theorem 7.4: In the ensemble of binary, periodically time-varying, rate $R_{ow} = R_o R_i$, woven convolutional codes with outer warp encoded by polynomial generator matrices of memory $m_{ow} \leqslant m_o + 1$,

(*i*) there exists a code whose normalized active row distance asymptotically satisfies

(7.15a)
$$\delta_\ell^{row} \triangleq \frac{a_j^{row}}{c_{ow} m_{ow}} \geqslant \max_{R_i} \left\{ \frac{\delta_C(R_i)}{l_o(R_i)} h^{-1} \left(1 - \frac{\ell}{\ell+1} \frac{R_{ow}}{R_i} \right) (\ell+1) \right\}$$

for $\ell \geqslant 0$,

(*ii*) there exists a code whose normalized active burst distance asymptotically satisfies

$$\delta_\ell^{bow} \triangleq \frac{a_j^{bow}}{c_{ow} m_{ow}} \geqslant \max_{R_i} \left\{ \frac{\delta_C(R_i)}{l_o(R_i)} h^{-1} \left(1 - \frac{\ell-1}{\ell} \frac{R_{ow}}{R_i} \right) \ell \right\} \tag{7.15b}$$

for $\ell \geqslant 1$,

(*iii*) there exists a code whose normalized active column (reverse column) distance asymptotically satisfies

$$\left. \begin{aligned} \delta_\ell^{cow} &\triangleq \frac{a_j^{cow}}{c_{ow} m_{ow}} \\ \delta_\ell^{rcow} &\triangleq \frac{a_j^{rcow}}{c_{ow} m_{ow}} \end{aligned} \right\} \geqslant \max_{R_i} \left\{ \frac{\delta_C(R_i)}{l_o(R_i)} h^{-1} \left(1 - \frac{R_{ow}}{R_i} \right) \ell \right\} \tag{7.15c}$$

for $\ell \geqslant O\left(\frac{\log m_o}{m_o}\right)$,

(*iv*) there exists a code whose normalized active segment distance asymptotically satisfies

$$\delta_\ell^{sow} \triangleq \frac{a_j^{sow}}{c_{ow} m_{ow}} \geqslant \max_{R_i} \left\{ \frac{\delta_C(R_i)}{l_o(R_i)} h^{-1} \left(1 - \frac{\ell+1}{\ell} \frac{R_{ow}}{R_i} \right) \ell \right\} \tag{7.15d}$$

for $\ell \geqslant O\left(\frac{\log m_o}{m_o}\right)$,

where $\delta_C(R)$ is the main term of the normalized Costello bound on the free distance (2.38) and $l_o(R_i)$ the solution of

$$h^{-1} \left(1 - \frac{l_o(R_i)+1}{l_o(R_i)} R_i \right) l_o(R_i) = \delta_C(R_i) \tag{7.16}$$

for $l_o(R_i) \geqslant \frac{R_i}{1-R_i}$. □

In Figure 7.3 the typical behavior of the bounds in Theorem 7.4 is shown. By minimizing the active row distance (7.15a) we get the following lower bound on the normalized free distance for woven convolutional codes with outer warp

$$\delta_{\text{free}}^{ow} \triangleq \frac{d_{\text{free}}^{ow}}{c_{ow} m_{ow}} \geqslant \max_{R_i} \left\{ \frac{1}{l_o(R_i)} \delta_C(R_i) \delta_C \left(\frac{R_{ow}}{R_i} \right) \right\}. \tag{7.17}$$

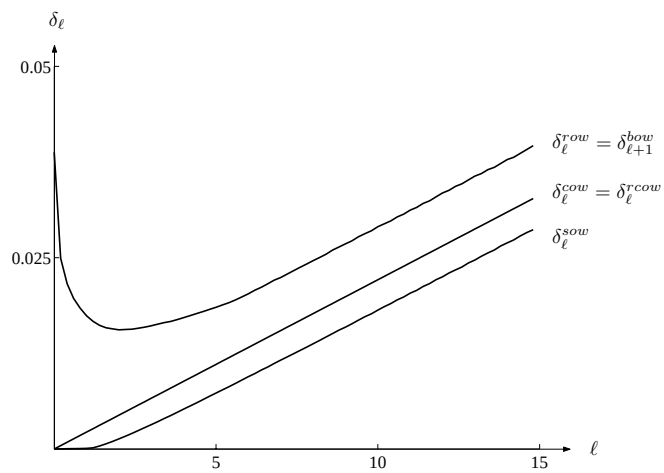

Figure 7.3: The behavior of the lower bounds on the normalized active distances of the woven convolutional codes with outer warp ($R_{ow} = 0.5$).

To write the corresponding bounds for active distances of woven convolutional codes with inner warp let

$$\ell = \frac{j+1}{m_i}. \tag{7.18}$$

Theorem 7.5: In the ensemble of binary, periodically time-varying, rate $R_{iw} = R_o R_i$, woven convolutional codes with inner warp encoded by polynomial generator matrices of memory $m_{ow} \leqslant 1 + m_i$,

(i) there exists a code whose normalized active row distance asymptotically

satisfies

(7.19a)
$$\delta_\ell^{riw} \triangleq \frac{a_j^{riw}}{c_{iw}m_{iw}} \geqslant \max_{R_o}\left\{\frac{\delta_C(R_o)}{l_i(R_o)}h^{-1}\left(1-\frac{\ell}{\ell+1}\frac{R_{iw}}{R_o}\right)(\ell+1)\right\}$$

for $\ell \geqslant 0$,

(*ii*) there exists a code whose normalized active burst distance asymptotically satisfies

$$\delta_\ell^{biw} \triangleq \frac{a_j^{biw}}{c_{iw}m_{iw}} \geqslant \max_{R_o}\left\{\frac{\delta_C(R_o)}{l_i(R_o)}h^{-1}\left(1-\frac{\ell-1}{\ell}\frac{R_{iw}}{R_o}\right)\ell\right\} \tag{7.19b}$$

for $\ell \geqslant 1$,

(*iii*) there exists a code whose normalized active column (reverse column) distance asymptotically satisfies

$$\left.\begin{aligned}\delta_\ell^{ciw} &\triangleq \frac{a_j^{ciw}}{c_{iw}m_{iw}}\\ \delta_\ell^{rciw} &\triangleq \frac{a_j^{rciw}}{c_{iw}m_{iw}}\end{aligned}\right\} \geqslant \max_{R_o}\left\{\frac{\delta_C(R_o)}{l_i(R_o)}h^{-1}\left(1-\frac{R_{iw}}{R_o}\right)\ell\right\} \tag{7.19c}$$

for $\ell \geqslant O\left(\frac{\log m_i}{m_i}\right)$,

(*iv*) there exists a code whose normalized active segment distance asymptotically satisfies

$$\delta_\ell^{siw} \triangleq \frac{a_j^{siw}}{c_{iw}m_{iw}} \geqslant \max_{R_o}\left\{\frac{\delta_C(R_o)}{l_i(R_o)}h^{-1}\left(1-\frac{\ell+1}{\ell}\frac{R_{iw}}{R_o}\right)\ell\right\} \tag{7.19d}$$

for $\ell \geqslant O\left(\frac{\log m_i}{m_i}\right)$,

where $\delta_C(R)$ is the main term of the normalized Costello bound on the free distance (2.38) and $l_i(R_o)$ the solution of

$$h^{-1}(1-R_o)\,l_i(R_o) = \delta_C(R_o) \tag{7.20}$$

for $l_i(R_o) \geqslant 0$. □

By minimizing the active row distance (7.19a) we get the following lower bound on the normalized free distance for woven convolutional codes with outer warp

$$\delta_{\text{free}}^{iw} \triangleq \frac{d_{\text{free}}^{iw}}{c_{iw}m_{iw}} \geqslant \max_{R_o}\left\{\frac{1}{l_i(R_o)}\delta_C(R_o)\delta_C\left(\frac{R_{iw}}{R_o}\right)\right\}. \tag{7.21}$$

8

Concluding Remarks

In this thesis the development of woven convolutional codes have been described. After a short introduction to convolutional codes the active distances were introduced. Those are distance measures that are well suited to handle concatenation of convolutional codes. Then, cascaded convolutional codes were thoroughly examined. Since that is the simplest construction of concatenated convolutional codes it helped us to develop the necessary intuition and understanding of the problems concerning concatenation of convolutional codes.

The examination of cascaded convolutional codes leads to the development of woven convolutional codes. Here the encoder properties were first examined, and then an iterative decoding scheme was designed and simulated.

In the next section some ideas for further work will be discussed.

8.1. Future Investigations

Most of the work in this thesis concerns the analyzes of the encoder properties. This is natural since the thesis is devoted to the development and understanding of woven convolutional codes. What is needed now is an exhaustive examination of the properties of the decoder. The purpose of Chapter 6 was to show that woven convolutional codes have potential of being realistic alternatives to other concatenated schemes, e.g., parallel or serial concatenation of convolutional codes or concatenation of Reed-Solomon codes and convolutional codes. It only covers a small fraction of the directions we can take in the investigations. There are numerous questions like 'which interleavers are good?' and 'how good is it for sub-optimal soft-in-soft-out decoding algorithms?'. Many of those are still unanswered.

We must also look closer into how the woven constructions can be compared with other concatenated systems. It is hard because Turbo codes or serially concatenated codes [4] are block codes created from convolutional codes. For those we compare the interleaver size or, equivalently, the block size. A change of this size will directly give a change in the results for these codes. Woven convolutional codes, on the other hand, are convolutional codes and in the simulations they have been truncated. If we double or halve the frame length the results will not be affected very much. To get a comparable system the frame lengths of the constituent encoders should be chosen as small as possible. This, however, results in a huge rate-loss due to the truncation of the frames. One way to create short effective woven convolutional codes would be to use short tail-biting convolutional codes as constituent codes. Such constructions can very well be compared with for example Turbo codes.

Bibliography

[1] J. D. Andersen and V. V. Zyablov. Interleaver design for turbo coding. In *Intern. Symposium on Turbo Codes*, pages 154–156, Brest, France, 1997.

[2] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. on Inform. Theory*, IT-20:284–287, March 1974.

[3] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. A soft-input soft-output maximum a posteriori (map) module to decode parallel and serial concatenated codes. Technical Report 42-127, TDA progress report, Nov. 15 1996.

[4] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding. *IEEE Trans. on Inform. Theory*, IT-44:909–926, May 1998.

[5] C. Berrou and A. Glavieux. Near optimum error-correcting coding and decoding: Turbo codes. *IEEE Trans. on Communications*, COM-44:1261–1271, Oct. 1996.

[6] C. Berrou, G. Glavieux, and P. Thitimashima. Near shannon limit coding and decoding: Turbo-codes. In *Proceedings of ICC'93*, pages 1064–1070, Geneva, Switzerland, May 1993.

[7] E. L. Blokh and V. V. Zyablov. Coding of generalized concatenated codes. *Problemy Peredachi Informatsii*, Vol. 10(3):218–222, 1974.

[8] D. J. Costello, Jr. A construction technique for random error-correcting convolutional codes. *IEEE Trans. on Inform. Theory*, IT-19:631–636, Sept. 1969.

[9] D. J. Costello, Jr. Free distance bounds for convolutional codes. *IEEE Trans. on Inform. Theory*, IT-20:356–365, July 1974.

[10] P. Elias. Coding for noisy channels. *IRE Conv. Rec.*, pages 37–46, 1955. Also in *Key Papers in Development of Coding Theory*, IEEE press, New York, NY, 1974.

[11] G. D. Forney, Jr. *Concatenated Codes*. MIT Press, Cambridge, Mass., 1966.

[12] G. D. Forney, Jr. Review of random tree codes. NASA Ames. Res. Con., Contract NAS2-3637, NASA CR 73176, Final Rep. Appendix A, Dec. 1967.

[13] G. D. Forney, Jr. Convolutional codes I: Algebraic structure. *IEEE Trans. on Inform. Theory*, IT-16:720–738, Nov. 1970.

[14] G. D. Forney, Jr. The Viterbi algorithm. In *Proceedings of the IEEE*, volume Vol. 61, pages 268–278, March 1973.

[15] G. D. Forney, Jr. Convolutional codes II: Maximum-likelihood decoding. *Information and Control*, Vol. 25:222–250, 1974.

[16] G. D. Forney, Jr., R. Johannesson, and Z.-x. Wan. Minimal and canonical rational generator matrices for convolutional codes. *IEEE Trans. on Inform. Theory*, IT-42:1865–1880, Nov. 1996.

[17] R. G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, Mass., 1963.

[18] R. G. Gallager. *Information Theory and Reliable Communication*. Wiley, New York, 1968.

[19] J. Hagenauer and P. Hoeher. A Viterbi algorithm with soft-decision outputs and its applications. In *Proceedings of IEEE Globecom conf.*, pages 1680–1686, Dallas, TX, Nov. 1989.

[20] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. on Inform. Theory*, IT-42:429–445, March 1996.

[21] R. W. Hamming. Error-detecting and error-correcting codes. *Bell System Technical Journal*, Vol. 29:147–160, 1950.

[22] C. Heegard and S. B. Wicker. *Turbo Coding.* Kluwer Academic Publisher, Dordrecht, the Netherlands, 1999.

[23] S. Höst, R. Johannesson, V. R. Sidorenko, K. Sh. Zigangirov, and V. V. Zyablov. *Communication and Coding*, chapter Cascaded Convolutional Codes, pages 10–29. In honor of Paddy G. Farrell on the occasion of his 60th birthday. Research Studies Press Ltd. and John Wiley & Sons, 1998.

[24] S. Höst, R. Johannesson, D. K. Zigangirov, K. Sh. Zigangirov, and V. V. Zyablov. On the distribution of the output error burst lengths for Viterbi decoding of convolutional codes. In *Proceedings of 1997 IEEE Intern. Symposium on Inform. Theory*, page 108, Ulm, Germany, June 24 - July 4 1997.

[25] S. Höst, R. Johannesson, and V. V. Zyablov. A first encounter with binary woven convolutional codes. In *Proceedings of the 4th Intern. Symposium on Comm. Theory and Appl.*, pages 13–18, Lake Districts, UK, July 1997.

[26] S. Höst, R. Johannesson, and V. V. Zyablov. Woven convolutional codes: Encoder properties. *In preparation*, 1999.

[27] S. Höst, R. Johannesson, V. V. Zyablov, and O. Skopintsev. Generator matrices for binary woven convolutional codes. In *Proceedings of 6th Intern. Workshop on Algebraic and Comb. Coding Theory*, pages 142–146, Pskov, Russia, Sept. 6–12 1998.

[28] S. Höst, R. Johannessson, O. D. Skopintsev, and V. V. Zyablov. Asymptotic distance properties of binary woven convolutional codes. *To appear in Problemy Peredachi Informatsii*, 1999.

[29] S. Höst, R. Johannessson, K. Sh. Zigangirov, and V. V. Zyablov. Active distances for convolutional codes. *IEEE Trans. on Inform. Theory*, IT-45:658–269, March 1999.

[30] S. Höst, R. Johannessson, and V. V. Zyablov. Nonequivalent cascaded convolutional codes obtained from equivalent constituent encoders. *Problemy Peredachi Informatsii*, Vol. 34(4):3–12, 1998.

[31] S. Höst, R. Johannessson, and V. V. Zyablov. Nonequivalent cascaded convolutional codes obtained from equivalent constituent convolutional encoders. In *Proceedings of 1998 IEEE Intern. Symposium on Inform. Theory*, page 338, Cambridge, Mass., Aug. 1998.

[32] S. Höst and V. R. Sidorenko. Some structural properties of cascaded convolutional codes. In *Proceedings of the 5th Intern. Workshop on Algebraic and Comb. Coding Theory*, pages 146–150, Sozopol, Bulgaria, June 1–7 1996.

[33] R. Johannesson and E. Paaske. Further results on binary convolutional codes with an optimum distance profile. *IEEE Trans. on Inform. Theory*, IT-24:264–268, March 1978.

[34] R. Johannesson and Z.-x. Wan. A linear algebra approach to minimal convolutional encoders. *IEEE Trans. on Inform. Theory*, IT-39:1219–1233, July 1993.

[35] R. Johannesson and K. Sh. Zigangirov. *Fundamentals of Convolutional Coding*. IEEE Press, Piscataway, N.J., 1999.

[36] J. Justesen, C. Thommesen, and V. V. Zyablov. Concatenated codes with convolutional inner codes. *IEEE Trans. on Inform. Theory*, IT-34:1217–1225, Sept. 1988.

[37] L.-N. Lee. Short unit-memory byte-oriented binary convolutional codes having maximal free distance. *IEEE Trans. on Inform. Theory*, IT-22:349–352, May 1976.

[38] S. Lin and D. J. Costello, Jr. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, Engleword Cliffs, N. J., 1983.

[39] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.

[40] J. L. Massey. Coding and modulation in digital communications. In *Proceedings of Intern. Zurich Seminar on Digital Communications*, pages E2(1)–E2(4), 1974.

[41] J. L. Massey and M. K. Sain. Codes, automata, and continous systems: Explicit interconnections. *IEEE Trans. on Automatic Control*, AC-12:544–650, 1967.

[42] E. Paaske. Short binary convolutional codes with maximal free distance for rates 2/3 and 3/4. *IEEE Trans. on Inform. Theory*, IT-20:683–689, Sept. 1974.

[43] L. Perez and D. J. Costello, Jr. Cascaded convolutional codes. In *Proceedings of 1995 IEEE Intern. Symposium on Inform. Theory*, page 160, Sept. 1995.

[44] C. Schlegel. *Trellis Coding*. IEEE Press, New York, NY, 1997.

[45] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, Vol. 27:379–423 (Part I) and 623–656 (Part II), 1948.

[46] C. Thommesen and J. Justesen. Bounds on distances and error exponents of unit-memory codes. *IEEE Trans. on Inform. Theory*, IT-29:637–649, Sept. 1983.

[47] A. J. Viterbi and J. K. Omura. *Principles of Digital Communication and Coding.* McGraw-Hill, New York, NY, 1979.

[48] V. V. Zyablov, S. Shavgulidze, and R. Johannessson. On the error exponent for woven convolutional codes with inner warp. *Submitted to IEEE Trans. on Inform. Theory*, 1999.

[49] V. V. Zyablov, S. Shavgulidze, O. Skopintsev, S. Höst, and R. Johannessson. On the error exponent for woven convolutional codes with outer warp. *IEEE Trans. on Inform. Theory*, IT-45:1649–1653, July 1999.

[50] V. V. Zyablov and S. A. Shavgulidze. Generalized convolutional concatenated codes with unit memory. *Problemy Peredachi Informatsii*, Vol. 22(4):9–28, 1986.