Ville Kuvaja

Identification of Error Correction Codes in Signals Intelligence

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 7.5.2015

Thesis supervisor:

Prof. Patric Östergård

Thesis advisor:

D.Sc. (Tech.) Jussi Poikonen

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Ville Kuvaja

Title: Identification of Error Correction Codes in Signals Intelligence

Date: 7.5.2015 Language: English Number of pages: 8+62

Department of Communications and Networking

Professorship: Communications Engineering Code: S-72

Supervisor: Prof. Patric Östergård
Advisor: D.Sc. (Tech.) Jussi Poikonen

Error correction coding is an integral part of a digital communication system. In signals intelligence the aim is to recover the transmitted messages and part of this task is identifying the used error correction coding method.

The purpose of this study is to present a overview of different identification methods of forward error correcting codes and test the performance of these codes in a controlled setting.

The codes that are discussed in this work are block codes and convolutional codes with a main focus on low density parity check (LDPC) codes and turbo codes. Test cases for LDPC code identification are presented and remarks about the performance and limits are made.

Keywords: error correction codes, identification, signals intelligence

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Ville Kuvaja

Työn nimi: Virheenkorjauskoodien tunnistus signaalitiedustelussa

Päivämäärä: 7.5.2015 Kieli: Englanti Sivumäärä: 8+62

Tietoliikenne- ja -tietoverkkotekniikan laitos

Professuuri: Tietoliikennetekniikka Koodi: S-72

Valvoja: Prof. Patric Östergård

Ohjaaja: TkT Jussi Poikonen

Virheenkorjauskoodit ovat oleellinen osa digitaalista tietoliikennejärjestelmää. Signaalitiedustelussa tavoite on selvittää lähetetty viesti ja osa tätä tehtävää on käytetyn virheenkorjauskoodin selvittäminen.

Tämän työn tarkoituksena on esittää yleiskatsaus erilaisiin virheenkorjaukoodien tunnistusmenetelmiin ja testata näiden menetelmien suorituskykyä kontroloiduissa olosuhteissa.

Virheenkorjauskoodit, joita käsitellään tässä työssä ovat lohkokoodit ja konvoluutiokoodit ja pääpaino on low density parity check (LDPC) -koodeissa ja turbokoodeissa. LDPC-koodin tunnistamismenetelmien testitulokset esitetään ja menetelmien suorituskykyä ja rajoitteita tarkastellaan.

Avainsanat: virheenkorjauskoodit, tunnistus, signaalitiedustelu

Preface

This thesis work was part of a MATINE project on error correction code identification. The work started with general guidelines to direct the work. This thesis was written in Department of Communication and Networking.

I would like thank Professor Patric Östergård for supervision as well as for creating a good research atmosphere. I would also like to thank my instructor Jussi Poikonen for valuable comments and advice. Finally I would like to thank all the students and staff whom I have been privileged to work with and all those who have helped and supported me.

Otaniemi, 7.5.2015

Ville J. N. Kuvaja

Contents

Symbols and Abbreviations

Symbols

$\mathbf{C}$	code
$\mathbf{c}$	code word
$\mathbf{C}^{\perp}$	dual code
D	delay operator
d_{min}	minimum distance
d	synchronization
$\mathbf{e}$	error vector
$\mathbf{G}$	generator matrix
g	impulse response, encoder polynomial
$GF(q)$	Galois field of order q
$\mathbf{H}$	parity check matrix
$\mathbf{I}$	identity matrix
K	constraint length
k	information bits
M	number of rows in a matrix
m	degree of encoder polynomials
$\mathbf{m}$	information word
n	encoded bits
P	probability
p	crossover probability
r	code rate
w	weight
X	input alphabet
$\mathbf{x}$	input vector
Y	output alphabet
$\mathbf{y}$	output vector
z	syndrome
γ	fractional rate loss
λ	polynomial
$\mathbf{\Pi}$	interleaver
τ	bit error probability

Operators

$\sum$	sum
$*$	convolution

Abbreviations

ARQ	automatic repeat request
BSC	binary symmetric channel
DVB	Digital Video Broadcasting
ETSI	European Telecommunications Standards Institute
FEC	forward error correction
IEEE	Institute of Electrical and Electronics Engineers
LDPC	low density parity check
WiMAX	Worldwide Interoperability for Microwave Access
WRAN	Wireless Regional Area Network

Chapter 1

Introduction

Error correction coding is an integral part of a digital communication system. Increase in available computational power has made it possible to use more advanced error correction codes in practical systems. Reliable transmission rates are already close to the channel capacities, which give the theoretical upper bounds for transmission of information in a given channel.

In signals intelligence the aim is to detect transmitted signals and recover the messages encoded in these streams. Part of this task is to identify the used error correction coding and decode the message without prior knowledge of the used code. From a signals intelligence viewpoint, advanced error correction codes can also make it more difficult to identify used codes and their parameters. To make the the identification task even more complicated these codes can be concatenated and the coded data is typically interleaved as well. A practical identification scenario will also involve noisy communication channels and synchronization might be unavailable, in other words, the starting location of a code word is not known.

1.1 Objectives

The aim of this thesis is to present an overview of different identification methods of forward error correcting codes in a non-cooperative context and to implement a test bed for testing these methods and principles in a controlled laboratory setting. The main focus is on the methods that are designed for modern forward error codes such as turbo codes and low density parity check (LDPC) codes. Code identification can be closely related to identification of interleaver parameters so interleaving is also covered in this work.

The problem of identification of forward error correcting codes without prior

knowledge of the used coding scheme can be a challenging task. In fact even with knowledge about the used coding method the parameters used in a particular case can be too hard to find. A brute force approach can prove to be insufficient for most practical cases. Consequently there exists a need for more powerful techniques that can be used with modern error correction codes.

Potential application for these identification methods can be in organizations that are involved with signals intelligence. Research results might be used in signal identification or in design of secure communication systems. Identification methods can have applications outside of military or intelligence context as well. There could be possible applications for communication systems with adjustable or changing error correction codes according to changing channel conditions as in a software radio system.

1.2 Organization

This thesis is organized as follows. In Chapter 2 error correction codes are introduced with a theory of block codes and convolutional codes. The operating principle of low density parity check codes and turbo codes are presented. Considerations on interleaving and transmission channel are also included in Chapter 2. Chapter 3 is about code identification methods as they are treated in the literature. Special emphasis is on LDPC and turbo codes. Identification of interleaver structure is included in Chapter 3. In Chapter 4 simulations and test cases are presented. In Chapter 5 conclusions of this work are presented together with recommendations for future research.

Chapter 2

Background

In this chapter background information relevant to code identification is presented. First, error correcting codes are described in Section 2.1. Two main classes of codes, block codes and convolutional codes are considered with their respective features. From these broad classes of codes we proceed to more specific codes. Low density parity check codes are a form of block codes and Turbo codes are a form of convolutional codes. These codes are modern codes that are used in today's communication systems. We examine properties of these codes. In Section 2.2 we explain the principle and implementation of interleaving. Interleaving is considered because it is an important subject in relation to error correction code identification as will be discussed later. In the final Section 2.3 transmission channels and how they relate to error correction codes and identification of error correction codes are considered.

2.1 Error Codes

In a digital communication system information is transferred as symbols. This enables the use of error correction coding to increase the system's immunity to noise. Error correction codes work by inserting controlled redundancy into transmitted messages. This redundancy is then used to detect and correct errors. In a digital communication system error control coding can be regarded as an operation between encryption and modulation as can be seen in the Figure 2.1. Error control coding can work with different strategies to combat errors. When errors are detected, retransmission of the received word can be requested, a strategy referred as automatic repeat request (ARQ), or the received word can be passed forward and tagged as erroneous, a strategy called muting. In this work the main interest is in error codes that try to correct the errors in the received word. These codes are called forward

error correction codes or FEC codes. There are trade-offs in these forward error codes that have to be considered. More redundancy is needed to be able to correct errors than to just detect them. On the other hand these codes can be used in situations where requesting a retransmission is impractical or impossible. For example in broadcast systems there is typically no way to retransmit information to individual users.

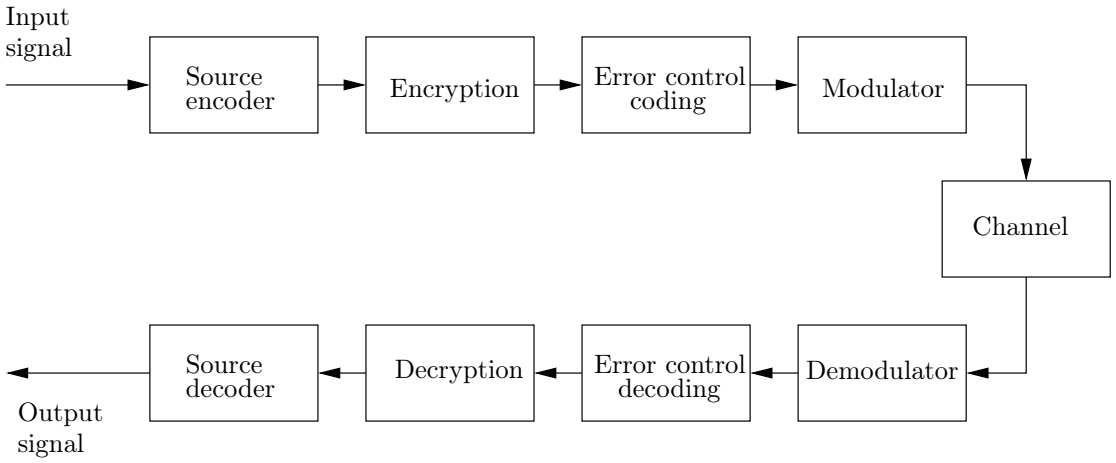

Figure 2.1: Communication system

2.1.1 Block Codes

A block code $\mathbf{C}$ is a set of M code words.

$$\mathbf{C} = \{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_{M-1}\}.$$

Each code word is a vector of the form $\mathbf{c} = (c_0, c_1, ..., c_{n-1})$. Vector coordinates, the c_i:s, will take values from the Galois field $GF(q)$. The code C is then called a q-ary code. In this work we will be working with codes that have $q = 2$, that is, binary codes.

Encoding of block codes is splitting the information stream into symbol blocks of length k and mapping these blocks onto length n symbol code blocks or code words, as depicted in Figure 2.2. When the symbols of the information stream can take any value in $GF(q)$, there will be $M = q^k$ possible information symbol blocks of length k and all these possible k-tuples $m = (m_0, m_1, ..., m_{k-1})$ will form a vector space over $GF(q)$.

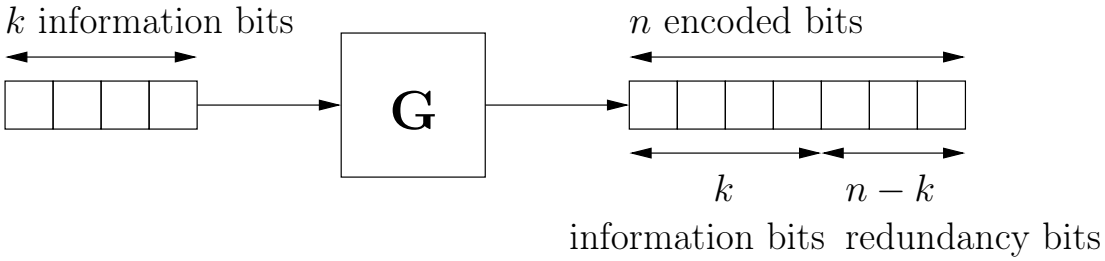

Figure 2.2: Linear block encoder

The length of a code word is n and the collection of all possible n-tuples forms a vector space over $GF(q)$ that contains q^n different vectors. As there are more possible code words than there are possible information words, the code C is said to contain redundancy. Redundancy can be expressed in logarithmic form as $r = n - \log_q M$. When $M = q^k$, this simplifies to $r = n - k$, that is difference between the lengths of the code block and the information block. Redundancy of a code $\mathbf{C}$ can be also expressed as a code rate. In logarithmic form code rate is expressed as $r = \frac{\log_q M}{n}$. And when $M = q^k$, this form simplifies to $r = \frac{k}{n}$.

Decoding of a block code starts with determining if a received word is a valid code word. If the decoder identifies the word to be invalid then it is assumed that errors have caused the code word to become invalid and an error is detected. The only way an erroneous code word can be undetected is if the errors in the received code word have converted the code word into an another valid code word. These type of errors are called undetectable errors. There are $M - 1$ undetectable error patterns for a size M code $\mathbf{C}$, that is, number of all the other valid code words. An error pattern is the error vector that can be expressed as

$$\mathbf{e} = (e_0, e_1, ..., e_{n-1})$$

where a nonzero coordinate indicates an error in that position.

The weight of a code word is the number of nonzero coordinates in the code word. The weight of the code word is expressed as $w(\mathbf{c})$. The weight can be calculated on error patterns as well, and again it is the number of nonzero coordinates of the error vector. The weight of an error pattern is relevant with error correcting codes as codes are designed to correct all error patterns of weight less than or equal to a certain limit.

Hamming distance is the number of coordinates in which two vectors differ. For

example, a hamming distance d between code words $\mathbf{v} = (00001010)$ and $\mathbf{w} = (11001010)$ is $d(\mathbf{v}, \mathbf{w}) = 2$. Minimum distance of a block code $d_{\min}$ is the minimum Hamming distance between all pairs of code words in code $\mathbf{C}$. In other words, the code words differ from each other at least in $d_{\min}$ coordinates. Minimum distance is an extensively used metric with block codes. A code with a minimum distance $d_{\min}$ can detect all error patterns of weight $d_{\min} - 1$ or less. This is because all code words differ from each other at least in $d_{\min}$ coordinates and in order to change a code word to an another code word, that is, to generate an undetectable error, code word must be changed in at least $d_{\min}$ coordinate positions. Minimum distance is also gives the number of errors a code $\mathbf{C}$ can correct. A code with minimum distance $d_{\min}$ can correct all error patterns of weight $\lfloor (d_{\min} - 1)/2 \rfloor$ or less.

A block code $\mathbf{C}$ that consists of n-tuples $\{(c_0, c_1, ..., c_{n-1})\}$ with symbols from $GF(q)$ is a q-ary linear code if and only if $\mathbf{C}$ is a vector subspace of $GF(q)^n$. The dimension of a linear block code $\mathbf{C}$ is the dimension of its vector space k. A linear code with length n and dimension k can be expressed as an (n, k) code. Linear codes have a structure that makes them easy to implement and analyze. The sum of linear code words is a code word and the product of a code word with a field element is a code word. The all-zero word is also a code word. For a linear code, minimum distance $d_{\min}$ is the weight of the code word that has the smallest weight $w_{\min}$, excluding the all-zero code word.

The theory of vector spaces can be used to analyze linear codes. A linear code is a subspace of $GF(q)^n$ and any set of basis vectors of the subspace can be used to form a generator matrix matrix $\mathbf{G}$ of the linear code. These basis vectors will be the rows of the k by n matrix $\mathbf{G}$. Any linear combination of the rows of matrix $\mathbf{G}$ is a code word. The rows of matrix $\mathbf{G}$ are linearly independent and the number of rows is the dimension of the code. The rank of the matrix $\mathbf{G}$ is k. There are q^k code words in $\mathbf{C}$ and these code words can be mapped with q^k different k-tuples.

Encoding of a linear block code is usually presented with an expression

$$\mathbf{c} = \mathbf{mG},$$

where $\mathbf{c}$ is the length n code word, $\mathbf{m}$ is the length k information word and matrix $\mathbf{G}$ is the generator matrix with basis vectors as rows.

For example, a simple $(3, 2)$ binary parity code with a generator matrix $\mathbf{G}$ that

will add a parity bit to each block of two information bits can be expressed as

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

The data block $\mathbf{m} = \begin{bmatrix} 0 & 1 \end{bmatrix}$ is encoded into a codeword as

$$\mathbf{mG} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}.$$

From the example above it can be seen how the resulting code word is a linear combination of basis vectors, that is the rows of the generator matrix $\mathbf{G}$, according to the data vector. The code $\mathbf{C}$ in this example is a subspace of $GF(2)^3$ and it has a dimension $k = 2$. The dimension is equal to the number of row vectors in the generator matrix $\mathbf{G}$.

Because the code $\mathbf{C}$ is a subspace it has an orthogonal component $\mathbf{C}^{\perp}$, that is the set of all vectors orthogonal to set of vectors $\mathbf{C}$. This orthogonal component is also a subspace of $GF(q)^n$ and is thus also a code. This code $\mathbf{C}^{\perp}$ is called the dual code of code $\mathbf{C}$. The dual code $\mathbf{C}^{\perp}$ has dimension $n-k$. Any basis for the dual code will therefore have $n-k$ vectors. The generator matrix $\mathbf{H}$ of the dual code $\mathbf{C}^{\perp}$ will be a matrix with any set of basis vectors as rows. Code word $\mathbf{c}$ will be a valid code word of code $\mathbf{C}$ if it is orthogonal to every row vector of $\mathbf{H}$. This can be expressed as a parity check equation,

$$\mathbf{cH}^{\top} = \mathbf{0}.$$

$\mathbf{H}$ is the parity check matrix of the code $\mathbf{C}$. The parity check matrix is an $(n-k)$ by n matrix. For the received code word $\mathbf{z} = \mathbf{x} + \mathbf{e}$, where the $\mathbf{x}$ is the valid code word and $\mathbf{e}$ is the error pattern, the vector that the equation produces is called the syndrome. The syndrome depends only on the error pattern as

$$\mathbf{zH}^{\top} = (\mathbf{x} + \mathbf{e})\mathbf{H}^{\top} = \mathbf{xH}^{\top} + \mathbf{eH}^{\top} = \mathbf{0} + \mathbf{eH}^{\top}.$$

In syndrome decoding, the decoder searches from a pre-computed table the error pattern matching the syndrome $\mathbf{eH}^{\top}$ and decodes the received word as $\mathbf{x} = \mathbf{z} - \mathbf{e}$.

For the example above, the parity check matrix is the 1 by 3 matrix $\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$. It can be seen that for every valid code word $\mathbf{c}$, the parity check equation

$\mathbf{cH}^\top$ will produce zero.

$$\begin{bmatrix}0 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}, \begin{bmatrix}0 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}, \begin{bmatrix}1 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}, \begin{bmatrix}1 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}.$$

For a received word $\begin{bmatrix}1 & 0 & 0\end{bmatrix}$ with an error pattern $\begin{bmatrix}1 & 0 & 0\end{bmatrix}$ and a transmitted code word $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$ the equation produces

$$\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}1\end{bmatrix}.$$

Here the syndrome only detects the error but can not correct it. This is because the code's minimum distance $d_{\min}$ is 2 and the code can detect $d_{\min} - 1 = 2 - 1 = 1$ errors and correct $\lfloor (d_{\min} - 1)/2 \rfloor = \lfloor (2-1)/2 \rfloor = 0$ errors.

From a linear code with parameters n, k and $d_{\min}$, it is possible to create a new code with the same parameters by taking two coordinates in a code word and changing the symbols in these coordinates with each other in every code word. This kind of permutation results in code that is said to be equivalent with the original code. For generator matrix this permutation operation means that we permute columns of the generator matrix. Generator matrices of equivalent codes are called equivalent generator matrices. A generator matrix can be converted to an equivalent generator matrix by permuting the columns or performing elementary row operations. Elementary row operations that can be performed are permuting a row, multiplication of a row by a non-zero constant and addition of a multiple of a row to another row. With these operations it is possible to convert any generator matrix into an equivalent form called the systematic form. The systematic form for a generator matrix is

$$\mathbf{G} = \begin{bmatrix}\mathbf{I} & \mathbf{P}\end{bmatrix},$$

where $\mathbf{I}$ is the k by k identity matrix and $\mathbf{P}$ is a k by $(n-k)$ matrix. This systematic generator matrix will encode length k data word into a length n code word where the first k symbols are the data word and the last $(n-k)$ symbols are the check symbols.

The above example's generator matrix $\mathbf{G}$ is already in a systematic form. The encoded code words are the data words with the parity check bit added as the last

bit.

The parity check matrix $\mathbf{G}$ for a generator matrix in a systematic form $\mathbf{G} = \begin{bmatrix} \mathbf{I} & \mathbf{P} \end{bmatrix}$ can be written as

$$\mathbf{H} = \begin{bmatrix} -\mathbf{P}^\top & \mathbf{I} \end{bmatrix}.$$

This follows from

$$\mathbf{G}\mathbf{H}^\top = \begin{bmatrix} \mathbf{I} & \mathbf{P} \end{bmatrix} \begin{bmatrix} -\mathbf{P} \\ \mathbf{I} \end{bmatrix} = -\mathbf{P} + \mathbf{P} = \mathbf{0}.$$

For the example's systematic generator matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \mathbf{P} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

the parity check matrix is

$$\mathbf{H} = \begin{bmatrix} -\mathbf{P}^\top & \mathbf{I} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

where $-\mathbf{P}^\top = \mathbf{P}^\top = \begin{bmatrix} 1 & 1 \end{bmatrix}$ in modulo 2 arithmetic.

2.1.2 Convolutional Codes

In this subsection convolutional codes are presented. This includes the basic structure of the convolutional encoder, different ways to evaluate convolutional codes and different ways to describe them. Further information about convolutional codes can be found in [48], [38], [28] and [8].

In a block code the encoder outputs a code word of length n from each length k data word whereas in a convolutional code the encoder outputs one coded stream from the input datastream. The encoder in Figure 2.3 has one input and two outputs, that is, it converts a single data stream into two coded streams. After the output these two streams are then combined into a single stream for transmission. The encoder is said to be a rate 1/2 encoder. The code rate for a convolutional encoder is generally $r = k/n$ as a fraction of inputs k and outputs n. This result is valid asymptotically, but for shorter input streams the length of the memory of the encoder and the time it takes to empty the registers after last information bit must be taken into account. For a length L data message it is possible to use fractional rate loss to describe the effect of register length on the encoder. Fractional rate loss is defined

as [48]:

$$\gamma = \frac{r - r_{\text{effective}}}{r} = \left(\frac{k}{n}\right)^{-1} \left(\left(\frac{k}{n}\right) - \left(\frac{L}{\left(\frac{n}{k}\right) L + nm} \right) \right) = \frac{km}{L + km},$$

where m is the shift register length.

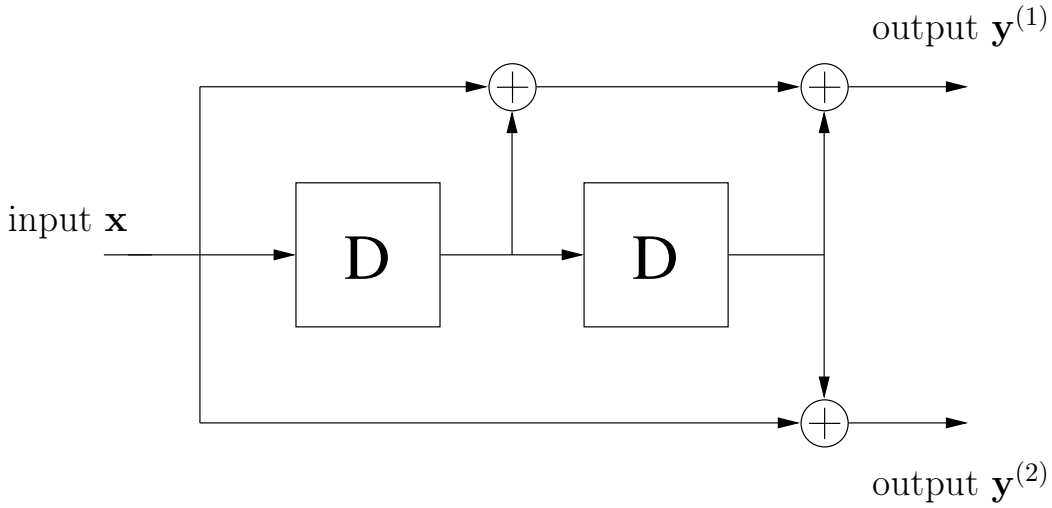

Figure 2.3: Convolutional encoder

As an example, input of the convolutional encoder in Figure 2.3 is fed with a binary data stream

$$\mathbf{x} = (x_0, x_1, x_2, ..).$$

This data stream goes into the shift registers and is read to outputs

$$\mathbf{y}^{(1)} = (y_0^{(1)}, y_1^{(1)}, y_2^{(1)}, ..),$$

$$\mathbf{y}^{(2)} = (y_0^{(2)}, y_1^{(2)}, y_2^{(2)}, ..)$$

of the encoder from the shift registers according to the connections in the encoder. These outputs are then interlaced into a single output stream

$$\mathbf{y} = (y_0^{(1)}, y_0^{(2)}, y_1^{(1)}, y_1^{(2)}, y_2^{(1)}, y_2^{(2)}, ..).$$

The shift registers in the encoder are assumed to be initialized to zero before the

encoding begins. For the encoder in Figure 2.3 output $\mathbf{y}^{(2)}$ is

$$\mathbf{y}_0^{(2)} = x_0 + 0$$

$$\mathbf{y}_1^{(2)} = x_1 + 0$$

$$\mathbf{y}_2^{(2)} = x_2 + x_0$$

$$\mathbf{y}_3^{(2)} = x_3 + x_1$$

$$\vdots$$

$$\mathbf{y}_j^{(2)} = x_j + x_{j-2}$$

From these equations the linearity of the encoder can be seen, as the output is a linear combination of the input. Linear structure of the code allows the use of techniques from linear systems theory. It is shown that all linear convolutional encoders have an equivalent minimal encoder without feed back loops in the encoder [48].

The impulse respose of a convolutional encoder is obtained by feeding a single one followed by a string zeros to the input. Impulse responses are frequently referred to as generator sequences. Impulse responses are given with reference to the particular input and output pair. For the encoder in Figure 2.3 the impulse responses are

$$\mathbf{g}^{(1)} = (111),$$

$$\mathbf{g}^{(2)} = (101).$$

These generator sequences have a length of 3 bits and this means that a single input bit can affect 3 output bits. This fact can also be seen from Figure 2.3, where there are two memory elements. The maximum number of bits a single input bit can affect is called the constraint length of the convolutional code. This number can be calculated as the longest input shift register plus one, $K = m + 1$, where m is the length of the shift register with longest length or memory.

The equation for the i:th output $y^{(i)}$ of the encoder can be expressed with generator sequences as [48]

$$y_j^{(i)} = \sum_{l=0}^{m} x_{j-l} g_l^{(i)}.$$

This equation is the discrete convolution of a pair of sequences. This equation is the reason for the name convolutional code. This can also be expressed with input

sequence $\mathbf{x}$, output sequence $\mathbf{y}^{(i)}$ and impulse response $\mathbf{g}^{(i)}$ as $\mathbf{y}^{(i)} = \mathbf{x} * \mathbf{g}^{(i)}$. For k-input encoders these equations have the following form.

$$y_j^{(i)} = \sum_{t=0}^{k-1} (\sum_{l=0}^{m} x_{j-l}^{(t)} g_{t,l}^{(i)})$$

$$\mathbf{y}^{(i)} = \sum_{t=0}^{k-1} (\mathbf{x}^{(t)} * \mathbf{g}_t^{(i)})$$

These equations can be expressed as a matrix multiplication operation. This provides a generator matrix that is similar to block codes. Because the input sequence is not bounded in length the corresponding generator matrix is not bounded either. This will result with a semi-infinite matrix. The generator matrix for a rate-1/2 code in a general form is created by interleaving the generator sequences $\mathbf{g}^{(1)}$ and $\mathbf{g}^{(2)}$ as follows [48].

$$\mathbf{G} = \begin{bmatrix} g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & \dots & g_m^{(1)} g_m^{(2)} & & & & \mathbf{0} \\ & g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & \dots & g_m^{(1)} g_m^{(2)} & & & \\ & & g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & \dots & g_m^{(1)} g_m^{(2)} & & \\ & & & g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & \dots & g_m^{(1)} g_m^{(2)} & \\ & & & & g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & \dots & g_m^{(1)} g_m^{(2)} \\ \mathbf{0} & & & & \ddots & \ddots & \ddots & \dots & \ddots \end{bmatrix}$$

This matrix is then used to encode the convolutional code word, or coded sequence

$$\mathbf{y} = \mathbf{x}\mathbf{G}.$$

For the example's encoder with generator sequences $\mathbf{g}^{(1)} = (111)$ and $\mathbf{g}^{(2)} = (101)$ the encoding of a information sequence $\mathbf{x} = (1011)$ is performed as

$$\mathbf{y} = \mathbf{x}\mathbf{G} = \begin{bmatrix} 1011 \end{bmatrix} \begin{bmatrix} 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 11 & 10 & 11 \end{bmatrix} = \begin{bmatrix} 11 & 10 & 00 & 01 & 01 & 11 \end{bmatrix}$$

.

Discrete convolution can be expressed in simpler form after the delay transform and the use of a delay operator D. After delay transform it is possible to express

the input, the output and the generator sequence as [48]

$$\mathbf{x}^{(i)} = (x_0^{(i)}, x_1^{(i)}, x_2^{(i)}, ...) \leftrightarrow \mathbf{X}^{(i)}(D) = x_0^{(i)} + x_1^{(i)}D + x_2^{(i)}D^2 + \dots$$

$$\mathbf{y}^{(i)} = (y_0^{(i)}, y_1^{(i)}, y_2^{(i)}, ...) \leftrightarrow \mathbf{Y}^{(i)}(D) = y_0^{(i)} + y_1^{(i)}D + y_2^{(i)}D^2 + \dots$$

$$\mathbf{g}_j^{(i)} = (g_{j0}^{(i)}, g_{j1}^{(i)}, g_{j2}^{(i)}, ...) \leftrightarrow \mathbf{G}_j^{(i)}(D) = g_{j0}^{(i)} + g_{j1}^{(i)}D + g_{j2}^{(i)}D^2 + \dots$$

The exponent of the delay operator D indicates the number of time units the coefficient is delayed against the coefficient with the D^0 term. Here in $g_{j2}^{(i)}$, j is the input index, i is the output index and 2 denotes the second bit of the sequence.

Using the delay operator we can perform the encoding operation for a single input and multiple output system as follows.

$$\mathbf{Y}^{(i)}(D) = \mathbf{X}(D)\mathbf{G}^{(i)}(D)$$

For a multiple input and multiple output system the equation can be expressed as

$$\mathbf{Y}^{(i)}(D) = \sum_{j=0}^{k-1} \mathbf{X}^{(j)}(D)\mathbf{G}_j^{(i)}(D).$$

The above equations can also be expressed as matrix operations. For a k input and n output system the matrix equation of the encoding operation will be in the following form.

$$\mathbf{Y}(D) = \mathbf{X}(D)\mathbf{G}(D) = \begin{bmatrix} \mathbf{X}^{(0)}(D) & \mathbf{X}^{(1)}(D) & \dots & \mathbf{X}^{(k-1)}(D) \end{bmatrix}$$
$$\cdot \begin{bmatrix} \mathbf{G}_0^{(0)}(D) & \mathbf{G}_0^{(1)}(D) & \dots & \mathbf{G}_0^{(n-1)}(D) \\ \mathbf{G}_1^{(0)}(D) & \mathbf{G}_1^{(1)}(D) & \dots & \mathbf{G}_1^{(n-1)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_{k-1}^{(0)}(D) & \mathbf{G}_{k-1}^{(1)}(D) & \dots & \mathbf{G}_{k-1}^{(n-1)}(D) \end{bmatrix}$$

Matrix representation of the previous encoding example with generator sequences

$$\mathbf{g}^{(1)} = (111) \leftrightarrow \mathbf{G}^{(1)}(D) = 1 + D + D^2,$$

$$\mathbf{g}^{(2)} = (101) \leftrightarrow \mathbf{G}^{(2)}(D) = 1 + D^2$$

and the information sequence

$$\mathbf{x} = (1011) \leftrightarrow \mathbf{X}(D) = 1 + D^2 + D^3$$

will be in the following matrix form.

$$\mathbf{Y}(D) = \left[1 + D^2 + D^3\right] \left[1 + D + D^2 \quad 1 + D^2\right] = \left[1 + D + D^5 \quad 1 + D^3 + D^4 + D^5\right]$$

Where $1 + D + D^5 \leftrightarrow (110001)$ and $1 + D^3 + D^4 + D^5 \leftrightarrow (100111)$ so the output string will be $(11, 10, 00, 01, 01, 11)$.

2.1.3 Low-Density Parity-Check Codes

In this subsection low density parity parity check (LDPC) codes are introduced. Their structure, description and construction are presented.

Low density parity check codes are linear block codes that are characterized by a sparse parity check matrix [27]. These codes were first introduced by Gallager in 1960. It has been shown that these codes have a comparable performance to turbo codes. This performance has resulted in inclusion of LDPC codes in several communication and broadcasting standards. [39]

LDPC codes are linear block codes that have very long codewords, from hundreds to even tens of thousands in broadcasting applications. The parity check matrices for these codes are very large and contain very few 1's compared to the number of 0's. Low density refers to this low density of 1's in the parity check matrix. The $m \times n$ parity check matrix of an LDPC code should satisfy that the number of 1's in a row, the row weight, will have $w_r \ll \min\{m, n\}$ and the number of ones in a column, the column weight, will have $w_c \ll \min\{m, n\}$. A regular low density parity check matrix is a sparse matrix that has the same weight w_c for every column and the same weight w_r for every row [32]. An irregular LDPC code is one with nonconstant row and column weights for rows and columns.

As an example, a block code with a parity check matrix

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

with size $n \times m, (n = 8, m = 4)$ is assumed to be a LDPC code. The dimensions of **H** are obviously smaller than in a practical LDPC code, but the key principles can still be illustrated. In the matrix the $(n - k)$ rows represent the parity check equations that must be fulfilled for every valid code word. That is every code word **c** must satisfy

$$c_1 + c_3 + c_4 + c_7 = 0$$

$$c_0 + c_1 + c_2 + c_5 = 0$$

$$c_2 + c_5 + c_6 + c_7 = 0$$

$$c_0 + c_3 + c_4 + c_6 = 0$$

where additions are performed modulo 2.

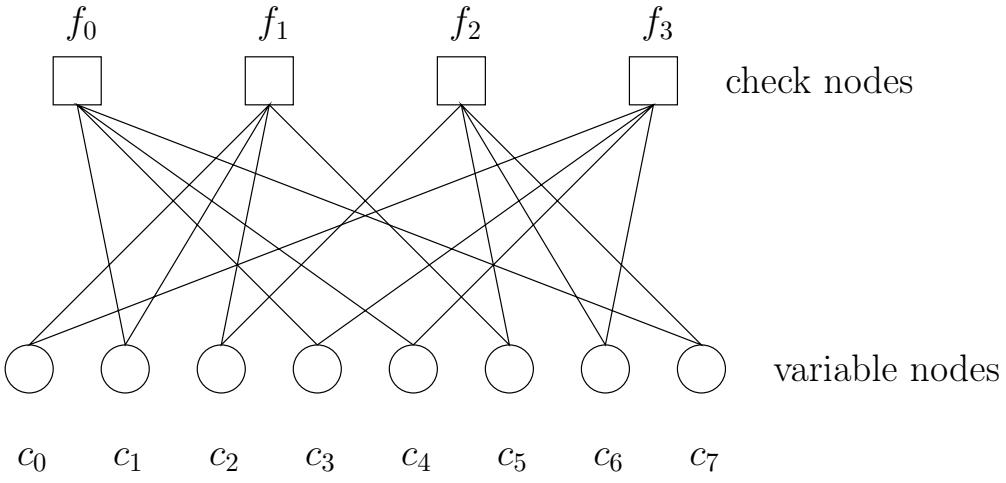

Figure 2.4: Tanner graph

LDPC codes are usually represented by a parity check matrix or a Tanner graph. A Tanner graph [44] for the parity check matrix **H** can be seen in Figure 2.4. A Tanner graph is a graphical representation of parity check equations of **H**. A graph is a data structure that is a collection of nodes and edges that connect the nodes. A degree of a node is the number of edges that are connected to the node [39]. A bipartite graph is a graph that divides the nodes into two subsets so that each edge connects two nodes from different subsets. A Tanner graph is a bipartite graph that divides the set of nodes to variable nodes and check nodes. The variable nodes represent the bits of the received word and the check nodes represent the parity checks, or the parity check bits, of the codeword. Comparing the matrix **H** and the

Tanner graph, correspondence between the check nodes and the rows of the matrix and between the variable nodes and the columns of the matrix can be observed.

A cycle in a graph is a path on the edges from a node back to the same node. A Tanner graph of a LDPC code usually has cycles in it. For example in the Tanner graph in Figure 2.4 there is a cycle $c_2 \rightarrow f_2 \rightarrow c_5 \rightarrow f_1 \rightarrow c_2$. The length of the shortest cycle is called the girth of the graph. The guaranteed error correction capability of an LDPC code is higher for codes with larger girth [12].

2.1.4 Turbo Codes

In this subsection turbo codes are introduced. The turbo code encoder structure is presented with a polynomial and a matrix description of the encoding process that is extracted mostly from [8].

A turbo code is a linear block code that is constructed from a terminated systematically encoded convolutional code using two different words of the convolutional code [8]. A turbo encoder encodes the data twice so that the two decoders for the two convolutional code words can use the decoding information from each other [8]. The decoding proceeds iteratively. The outcome of an iteration is an a posteriori probability that is used as an input for the next iteration [28]. For a large binary turbo code, ten to twenty iterations are usually sufficient for the outcomes of the iterations to converge between the decoders [8]. A turbo code can also use puncturing, a process where some of the output bits are removed after the encoding. Puncturing a code will result in a higher code rate.

A turbo code encoder is shown in Figure 2.5. In this figure there are two parallel convolutional encoders, $\mathbf{G}_1$ and $\mathbf{G}_2$ and the interleaver $\mathbf{\Pi}$ that precedes the second encoder. From the figure it can also be seen that the code is systematic, as the unmodified input stream is fed straight to the output. The two convolutional encoders are usually identical.

A systematic rate 1/2 convolutional encoder in the turbo encoder can be presented with polynomial generator matrix

$$\mathbf{G}(x) = \begin{bmatrix} 1 & g(x) \end{bmatrix}.$$

Here the $g(x)$ is the generator polynomial of the convolutional code. In a turbo code there can be recursive convolutional encoders that have feedback structure. Such a structure will be represented with rational functions of the form $\frac{g_1 x}{g_2(x)}$ in the generator matrix instead of polynomials. The $g_1(x)$ will represent the feedforward connections

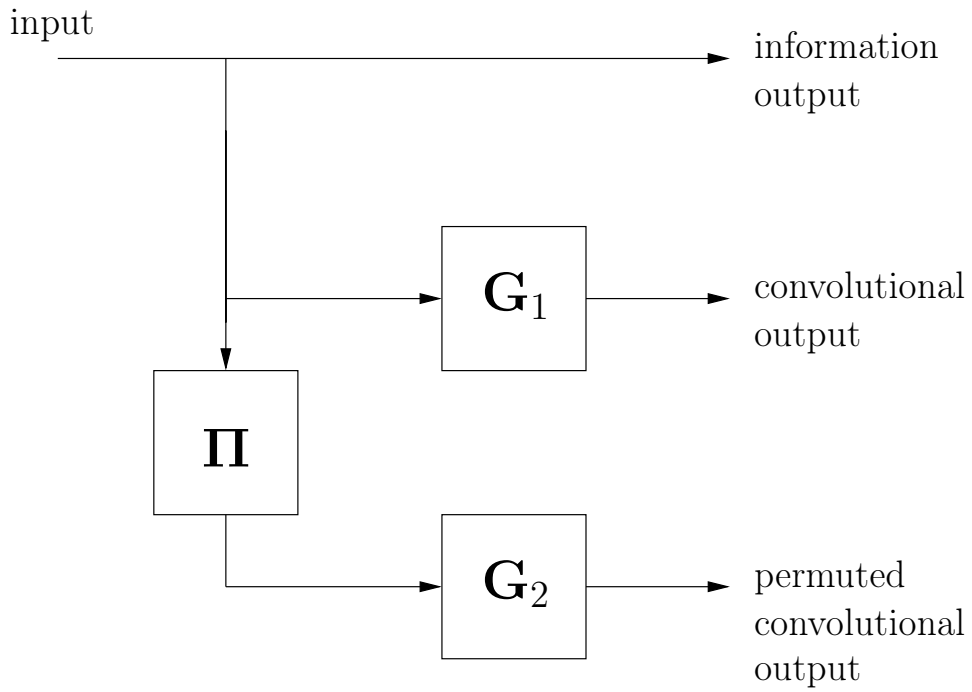

Figure 2.5: Turbo code encoder

and the $g_2(x)$ the feedback connections. A code word $\mathbf{c}$ of this convolutional code with a data word $a(x)$ will have the form

$$\mathbf{c}(x) = a(x)\mathbf{G}(x).$$

This code word can be written also as

$$\begin{bmatrix} c_0(x) & c_1(x) \end{bmatrix} = \begin{bmatrix} a(x) & a(x)g(x) \end{bmatrix}.$$

When the length of the data word is restricted, the data word will have a maximum degree of $k-1$ at most. The degree of $g(x)$ is defined to be r. This finite length convolutional code can then be described as a block code. A generator matrix for the terminated convolutional code in the block code representation will be

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \begin{bmatrix} g(x) \end{bmatrix} \end{bmatrix}.$$

Here the $\mathbf{I}$ is the k by k identity matrix and $\begin{bmatrix} g(x) \end{bmatrix}$ is the matrix of coefficients of $g(x)$ with rows shifted cyclically. The code word will be a pair $(a(x), a(x)g(x))$ with a block length of $n = r + 2k$.

The interleaver $\mathbf{\Pi}$ in the turbo encoder can be described with a permutation matrix of size k by k according to the maximum degree of the data word polynomial. A permutation matrix is an identity matrix where the rows have been permuted.

The turbo code can be presented with a generator matrix of the form

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \begin{bmatrix} g(x) \end{bmatrix} & \mathbf{\Pi} \begin{bmatrix} g(x) \end{bmatrix} \end{bmatrix}.$$

The generator matrix $\mathbf{G}$ has k rows and $2r + 3k$ columns. Therefore the turbo code is a $(2r + 3k, k)$ block code.

For example, in a small turbo code with $(n, k) = (13, 3)$, the convolutional generator polynomial $g(x) = g_2x^2 + g_1x + g_0$ and the permutation matrix

$$\mathbf{\Pi} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

the generator matrix of the turbo code will be

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 & g_2 & g_1 & g_0 \\ 0 & 1 & 0 & 0 & g_2 & g_1 & g_0 & 0 & g_2 & g_1 & g_0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & g_2 & g_1 & g_0 & 0 & g_2 & g_1 & g_0 & 0 \end{bmatrix}$$

and the code word will be

$$\mathbf{c} = \mathbf{aG}$$

where the data word $\mathbf{a}$ is a vector of length $k = 3$. The code word $\mathbf{c}$ can be divided into three parts, the systematic part, the convolutional code part and the permuted convolutional code part. The code word can then be presented as a vector of three polynomials as

$$c(x) = (a(x), g(x)a(x), g(x)a'(x)) = (c_0(x), c_1(x), c_2(x)).$$

These three polynomials correspond to the three output streams.

With a generator sequence $\mathbf{g} = (101)$, the generator polynomial is $x^2 + 1$ and the systematic generator matrix will be

$$\mathbf{G}(x) = \begin{bmatrix} 1 & x^2 + 1 \end{bmatrix}.$$

The terminated code words will have the form $(a(x), a(x)(x^2 + 1))$. The block code

representation of the generator matrix will have the following form

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

The permutation matrix is chosen to be

$$\mathbf{\Pi} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

The rows indicate the permutation order, so that in the permuted part of the turbo code generator matrix, first will come the third row of the non systematic part of the generator matrix, then the first and the second. The $(13, 3)$ turbo code thus corresponds to the block code with a generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Encoding the data word $\begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$ will then result in a code word

$$\mathbf{c} = \mathbf{aG} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

From the code word it is possible to see the three parts or the three output streams. The first three bits are the systematic data part, then comes the five bit convolutional check part and finally the five bit permuted convolutional check part. The use of two checks in parallel is referred to as parallel concatenation. Useful turbo codes will have a much larger block length than in the above example [8].

2.2 Interleaving

Most error correcting codes have been designed to correct random errors that are statistically independent. If errors occur in bursts then error correction capability is wasted as some words can be heavily corrupted while others can be error free. Interleaving the coded data is an effective method to combat burst errors so that the channel appears as having independent errors and error correction codes designed for independent errors can be used. An interleaver reorders the symbols from several code words so that the symbols in a given word are separated from each other. Transmitted data is then deinterleaved in the receiver by a deinterleaver and arranged in the correct order for the decoder.

Interleavers can be divided into two main types according to their structure, block or convolutional interleavers. Difference between the two types is related to the implementation of the interleaver circuits and both types can be used in a communication system in much the same way. A brief description of the operating principle of a block interleaver is presented next.

A block interleaver arranges the coded data into a matrix. Input data to the interleaver is read into rows of the matrix. Output from the interleaver is read out column wise. The length of the column vector is the interleaver period that defines the separation distance of bursty errors. In the receiver side a deinterleaver stores the data in the same matrix format as in the transmitter side but the data is read out row wise. A block interleaver with 3 rows and 3 columns is illustrated in Figure 2.6, where the input string is $1, 2, 3, 4, 5, 6, 7, 8$ and 9, and the output string is $1, 4, 7, 2, 5, 8, 3, 6$ and 9. The length of the column vector is 3 in this interleaver and so the separation distance is 3, which can be checked also from the output string, where consecutive input bits are are separated by at least three bit positions.

The interleaving operation can be described in matrix form with a permutation matrix. For a coded data vector $\mathbf{c} = [c_0, c_1, c_2, ..., c_{n-1}]$ the permutation matrix is an n by n square matrix $\mathbf{\Pi}$ of zeros and ones. $\mathbf{\Pi}$ has a single one in every row and in every column and the rest of the matrix entries are zeros. This permutation matrix can be constructed from a identity matrix by permuting the rows or the columns. The permutation matrix will permute the rows of a matrix when it multiplies it from the left and the columns when it multiplies it from the right.

For example the interleaving of a code vector $\mathbf{c} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^\top$ so that the output of the interleaver will be the third code bit, then the first and the second is done

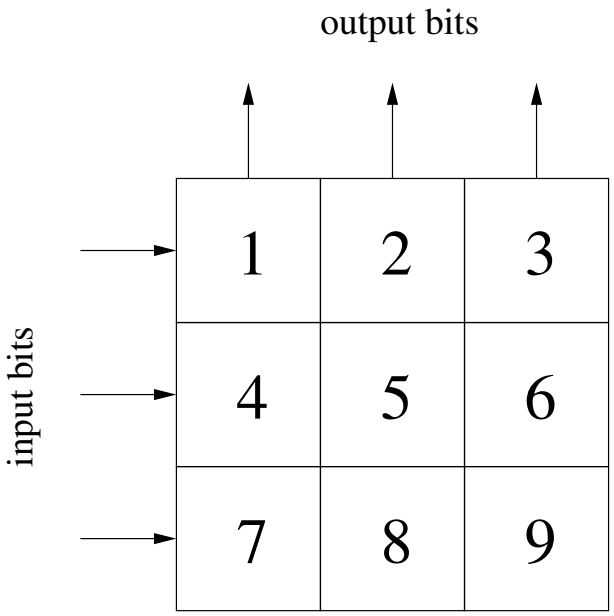

Figure 2.6: Block interleaver

with a permutation matrix $\mathbf{\Pi}$ by

$$\mathbf{\Pi c} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}.$$

Here the permutation matrix operates on the rows, that is the entries in the column vector $\mathbf{c}$. The same operation could be performed by transposing $\mathbf{\Pi}$ and multiplying a row vector from the right.

2.3 Transmission Channel

In this section the general communication channel model is presented. The specific channel model that is used in this work is also described.

In digital communications a general transmission channel is described by the possible inputs, the input alphabet X and the possible outputs, the output alphabet Y and the conditional probabilities $\mathrm{P}[\mathbf{y}|\mathbf{x}]$ that relate the input $\mathbf{x} = (x_1, x_2, ..., x_n)$ and the output sequences $\mathbf{y} = (y_1, y_2, ..., y_n)$ of length n. A channel is defined to be

a memoryless channel when

$$\mathrm{P}[\mathbf{y}|\mathbf{x}] = \prod_{i=1}^{n} \mathrm{P}[y_i|x_i] \qquad \text{for all } n.$$

The equation says that in a memoryless channel the output at time i depends only on the input at time i. When the memoryless channel has a input alphabet and a output alphabet that are the same with $X = Y = \{0, 1\}$, the channel is called the binary symmetric channel (BSC) [39]. This binary symmetric channel is the channel model that is used in this work.

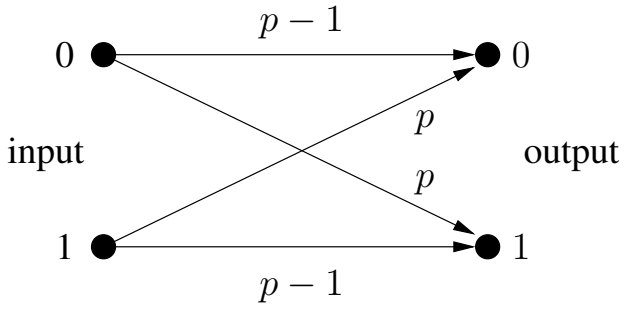

Figure 2.7: Binary symmetric channel

If channel noise causes statistically independent errors in the transmitted binary sequence with average probability p then we have the following probabilities.

$$\mathrm{P}[Y = 0|X = 1] = \mathrm{P}[Y = 1|X = 0] = p$$

$$\mathrm{P}[Y = 1|X = 1] = \mathrm{P}[Y = 0|X = 0] = 1 - p$$

The probability p is commonly referred to as the crossover probability, as it describes the probability that the bit will cross its value from 0 to 1 or vice versa in the graphical representation of a binary symmetric channel. The above probabilities can be represented in a graphical form as in Figure 2.7.

Chapter 3

Code Identification

Identification of a code is a problem that requires solving different subproblems. One needs to find out the interleaver parameters, synchronization and recognize the code length. For a non-binary code one also needs to recognize the used alphabet, synchronize the symbols and extract the symbols. After these steps comes the code identification and noise extraction. Then it is necessary to recognize the mapping between code words and non-encoded words. This list only includes those operations that are closely related to code identification. A complete retrieval of the transmitted message in a modern communication system would also require to solve additional problems, for example, identification of the used modulation method and possibly cryptanalysis. [46]

In this chapter different methods to obtain information about the used error correction code are presented. At first methods for identification of interleaver structure are presented. Then principles of algorithms for block codes and convolutional codes are illustrated. Identification methods as they are applied for LDPC codes and turbo codes conclude this chapter.

3.1 Interleavers

In this section interleaver parameter recovery is presented. When the structure of the interleaver is not known, there are different methods [9], [40], [41], [17] to find out the interleaver parameters.

A simple method [40] proposed for estimating the interleaver period is:

$$\rho_a = \frac{\text{the number of vectors that occur}}{2^{n_a}}$$

where the number of different vectors of size n_a that are found in the intercepted stream are counted and this number is divided by the number of all possible different vectors of size n_a. The interleaver period that minimizes ρ_s will be the estimate for the interleaver period. The idea of calculating different vectors from the stream is used in conjunction with convolutional codes in [5].

Identification of the interleaver parameters can be closely related to identification of the error correction codes. In fact, it will be shown next how the code rate of a block code can be estimated with an algorithm that is primarily designed to detect the interleaver parameters.

This method presented in [9] estimates the interleaver period, performs a blind synchronization on the interleaver blocks and finally estimates the code rate of the block code. Adaptation of the method to convolutional encoders is also possible [9].

The method proposed in [9] for solving interleaver structure with block codes is based on linear algebra. At first the interleaver period n_i is estimated. This period is assumed to be a multiple of the length of the encoded block. The intercepted stream is divided into analysis blocks of size n_a. These blocks are then used as columns in a matrix Z. Examining the ratio ρ, as defined in (3.1), for different block sizes n_a, one can note that ρ is equal to 1, except when n_a is a multiple of the correct interleaver period n_i.

$$\rho = \frac{rank(Z)}{n_a}, \tag{3.1}$$

The rank of a matrix is the largest number of linearly independent columns or rows in the matrix and this method is based on finding the size of the matrix that is not of full rank. The rank is full when analysis block size is not a multiple of interleaver period and columns of the matrix Z are linearly independent. Complexity of calculating rank by Gaussian elimination is $O(n^3)$.

To illustrate the functioning of this algorithm, a simple example is presented, as in [9]. Considering a simple parity code with a code word length $n = 3$, message length $k = 2$ and a random interleaver with period $n_i = 12$. A matrix Z is built from the interleaved stream of 2400 symbols and then ρ versus increasing analysis block sizes n_a is plotted. The plot can be seen in Figure 3.1.

Upper bound for ρ is derived as $\rho = r + \frac{1-r}{b_i}$, where integer b_i is the multiple of the interleaver period and r is the code rate $\frac{k}{n}$. Lower bound for ρ is the code rate, as will be seen later.

Once the interleaver period n_i has been estimated, the next step is to estimate the synchronization d. Synchronization is the number of symbols that the start

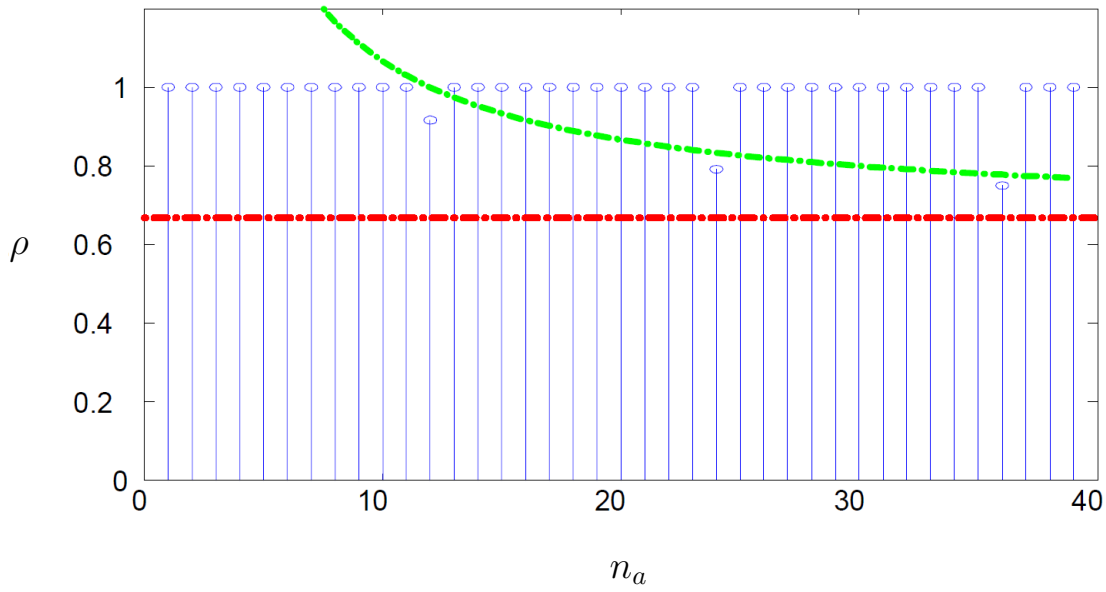

Figure 3.1: Estimation of interleaver period (the dashed curves are lower and upper bounds for ρ)[9]

of the intercepted stream is shifted from the start of the coded block. Interleaved stream is first divided into analysis blocks of size $n_a = n_i$ and first $\hat{d}$ symbols are skipped. This time ρ versus $\hat{d}$ is plotted. The plot for the example is shown in Figure 3.2. It can be noted that minimum of ρ and thus the synchronization is found at $\hat{d} = d$.

After the synchronization is found, the code rate is estimated. The first d symbols are skipped from the interleaved stream and the values of ρ for increasing values n_a are plotted. When n_a is a multiple of the interleaver period n_i, the value of ρ is the code rate. The plot is shown in Figure 3.3, where it can be seen that the code rate is $r = \frac{k}{n} = \frac{2}{3}$.

The method presented in [9] is developed for error-free transmission. If the transmission channel introduces errors to the intercepted stream, the matrix Z will be of full rank and the method will not be successful. A method for a scenario where the intercepted stream is corrupted with errors is presented in [41]. This method is based on the same concept as the earlier method, where the matrix rank was used to find interleaver parameters.

At first an analysis matrix $H(n_a, d)$ is built from the intercepted stream, where n_a is the row length of the analysis matrix or the interleaver size, and d is the number of skipped symbols from the beginning of the symbol stream or synchronization.

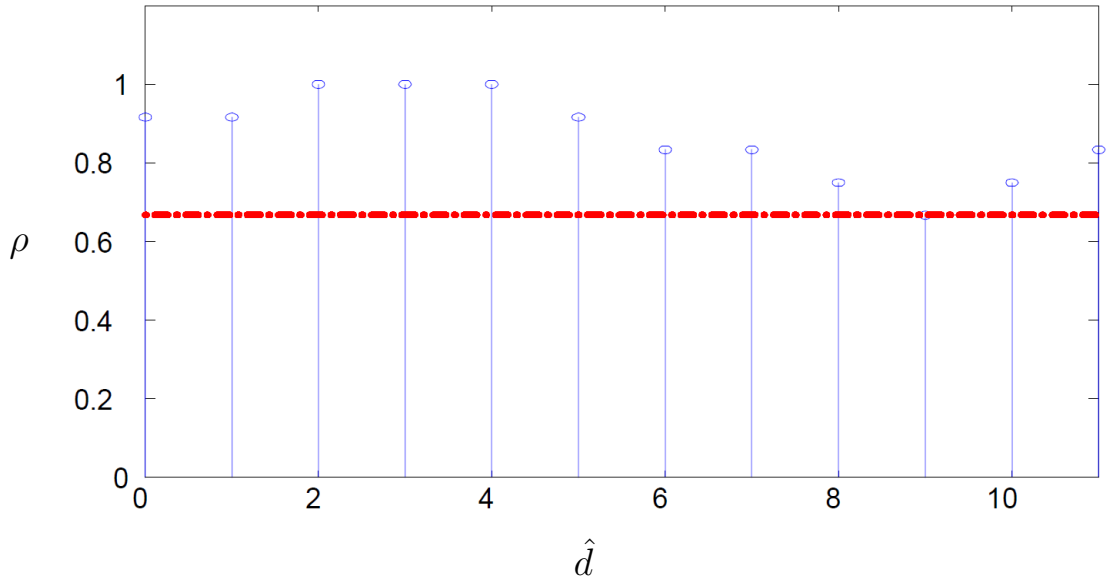

Figure 3.2: Estimation of synchronization (the dashed line shows the code rate) [9]

Matrix H is then converted into lower triangular matrix $L(n_a, d)$. Now B_i is the number of ones in the column i of $L(n_a, d)$ from row n_a to the last row. For an independent column i, B_i is binomially distributed with a mean m_B. Next $\phi(k)$ is defined as

$$\phi(k) = \frac{B_k}{m_B}.$$

For $n_a \neq \alpha S$, $\alpha \in \mathbb{N}$, where S is the interleaver size, it can be shown that $\forall k \in \{1, \ldots, n_a\}$, $\lim_{M\to\infty} \phi(k) \xrightarrow{p} 1$, where M is the number of rows in the matrix and $\xrightarrow{p}$ is the convergence in probability. Now if $n_a = \alpha S$, it can be shown that for some columns $\phi(k)$ will not converge to one. The maximum number of these columns is reached at $d = t_0$, that is when the synchronization is correct. The difference between $\phi(k)$ for different columns is estimated to be significant enough to estimate the interleaver parameters when the error probability of the channel is not too high and there are not too many dependent columns.

Also here it is possible to gain information about the used error correction code in addition to the information about the interleaver. Indeed, code rate can be estimated by dividing the number of dependent columns with the interleaver size. The estimate might not be perfect however, as the algorithm might have missed some dependent columns. In order to improve the performance of the algorithm, it is possible to make a so called new virtual realization of $H(n_a, d)$ by permuting the

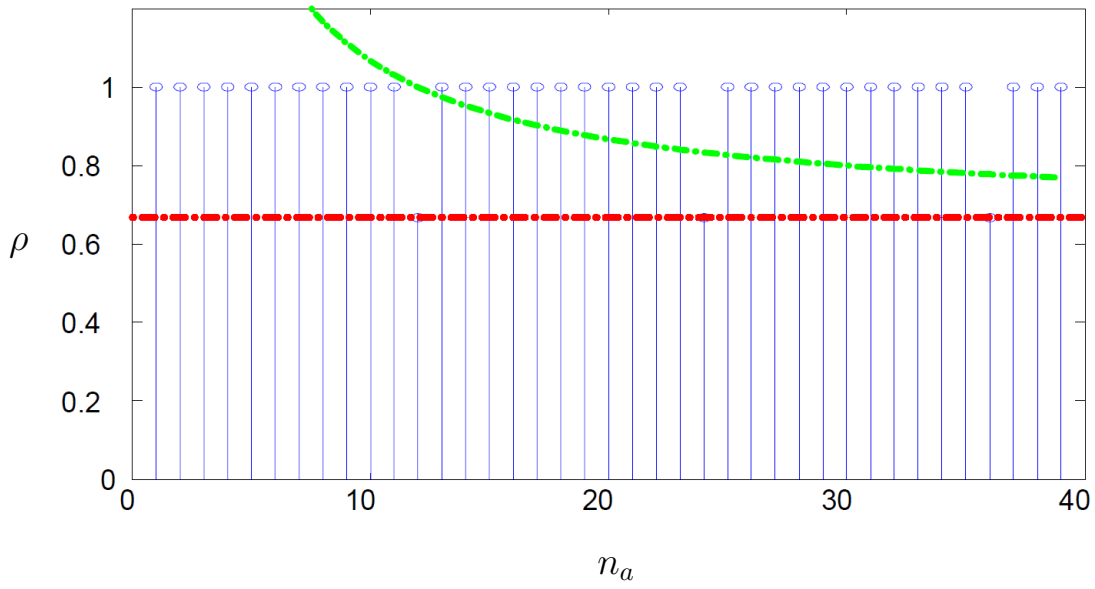

Figure 3.3: ρ versus n_a when analysis blocks are synchronized [9]

rows of the matrix. This way the estimate can be completed and made iteratively more accurate. In Figure 3.4 it is shown how with more iterations it is possible to increase the correct detection probability. This method gives more information about the interleaver than the method in [9] as the found dependent columns in the interleaved block belong to the same code word. In Figure 3.5 the proportion of found vectors in the basis $\hat{\mathcal{D}}_{S,d}$ is plotted versus the bit error rate. Here the basis $\hat{\mathcal{D}}_{S,d}$ is the set of vectors and a vector is a set of found dependent column positions.

Recovering the length and synchronization can also be based on searching for words in the dual code as is proposed in [17]. A q-ary code $\mathbf{C}$ of length n is a vector subspace embedded within the space of all n-tuples over $GF(q)$. The dual space of a linear code $\mathbf{C}$ is called the dual code of $\mathbf{C}$ and is denoted by $\mathbf{C}^{\perp}$. Words in the dual code will be orthogonal to the words in the coded stream. So if one can find such dual words h of the dual code $\mathbf{C}^{\perp}$ the product of h with words of the code $\mathbf{C}$ will be zero. Searching for words in the dual code can be sufficient to decide if a specific length and synchronization is correct.

In order to search for dual words, the intercepted stream is first split into words of test length n. Then a matrix $\mathbf{G}$ is constructed so that each row will be one word. Matrix $\mathbf{G}$ will be of size $M \times n$, where n is the word length and $M = \lfloor \frac{l-s}{n} \rfloor$, where l is the length of the intercepted bitstream and s is the synchronization. If h is a word of

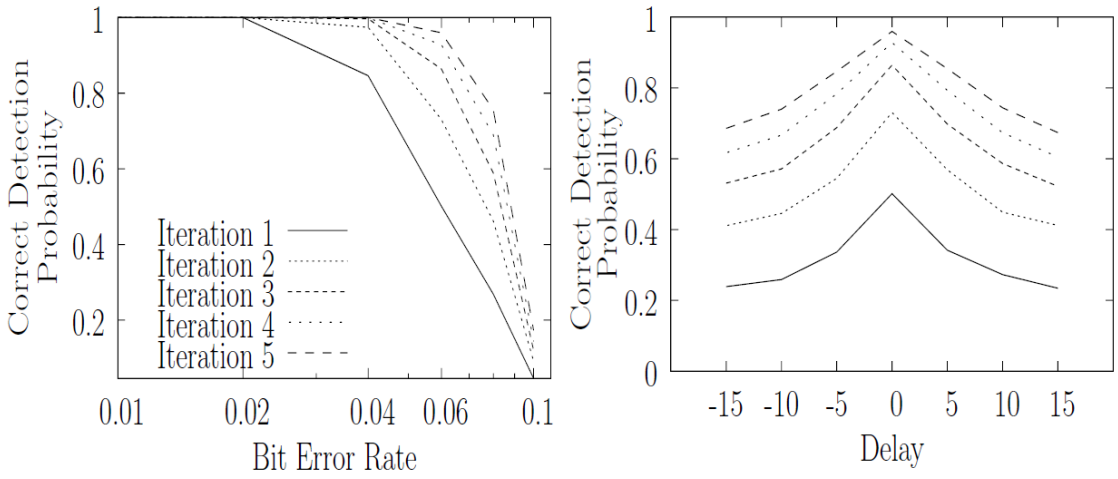

Figure 3.4: Correct detection probability of the interleaver size [41]

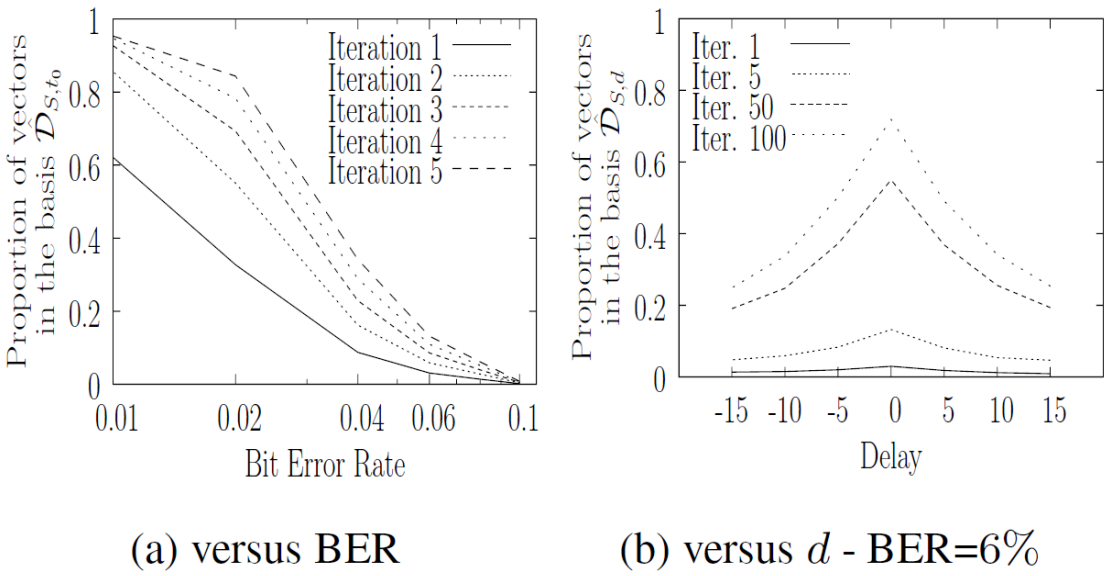

Figure 3.5: Proportion of vectors found in $\hat{\mathcal{D}}_{S,d}$ [41]

the dual of $\mathbf{C}$ the product $\mathbf{G} \times h$ will depend on the binary symmetric channel cross-over probability τ and follow binomial distribution centered at $\frac{M}{2}(1-(1-2\tau)^w)$, with a variance of $\rho^2 = \frac{M}{4}(1-(1-2\tau)^2 w)$. This distribution is shown in Figure 3.6. Distribution of the weight of product $\mathbf{G} \times h'$ when h' is not a word in the dual of $\mathbf{C}$ will follow the binomial distribution with mean $\frac{M}{2}$ and variance $\frac{M}{4}$.

If the length of the code word is chosen incorrectly it is unlikely to find any words so that the weight of the product $\mathbf{G} \times h$ would follow binomial distribution with center at $\frac{M}{2}(1-(1-2\tau)^w)$. If the synchronization is incorrect, then shifted dual words can be found if the support of the shifted dual word is in either of the two intervals $[0, d-1]$ or $[d, n-1]$, where d is the used synchronization. Probability of finding such words will be lower as the synchronization offset grows.

When searching for words in the dual, words with the lowest weight w are easier to find [17]. For low weight words the distributions of weight of the product $\mathbf{G} \times h$, as shown in Figure 3.6, are further away from one another and also the computing time will be less for exhaustively going through words of weight w and their product with $\mathbf{G}$. Authors use two different algorithms, [13] and [10], in the search for words in the dual of $\mathbf{C}$. This method is studied further in [42], where theoretical probability of detection is computed and optimal detection threshold is derived.

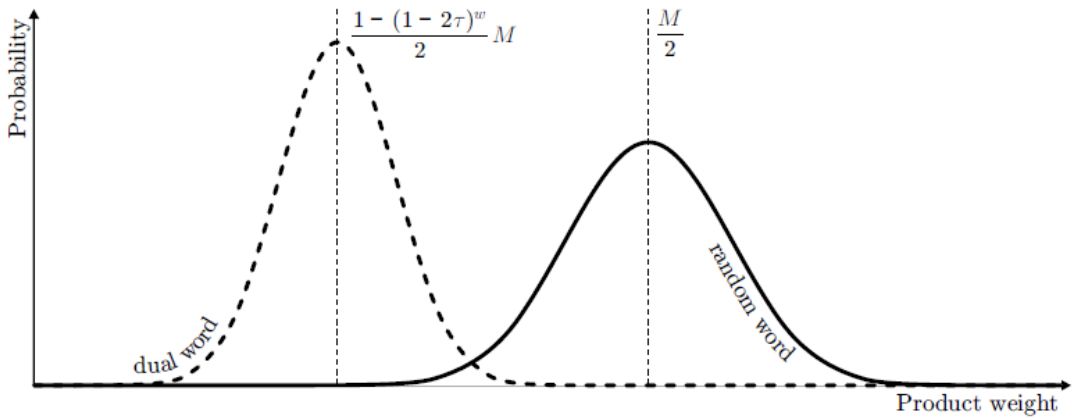

Figure 3.6: Distribution of weight of the Product $G \times h$. [17]

3.2 Block Codes

The problem of detection and recognition of a binary linear block code is examined and formalized in [46]. The problem is to find the nearest (n, k)-code in Hamming distance from the received binary stream [46].

The following hypotheses are used in the examination. The observed binary stream has been transmitted through a BSC channel with a bit error probability

$\tau < \frac{1}{2}$. From this stream the erroneous consecutive code words of length n are intercepted. All the code words are a priori equally likely to be transmitted.

Then from these assumptions is derived the probability that the code $\mathbf{C}$ was used knowing that the stream $\mathbf{X}$ was received. From this maximum likelihood problem is derived an associated decision problem that is proved to be NP-complete. After this general theoretical result practical situations are considered.

An algorithm based on dual words is suggested for recognition. This algorithm finds the length, the dimension and synchronization of a block code. The algorithm will be efficient for codes of length up to 512 with no more than 1.5 errors on average in one code word [46]. A table of limits on code length, dimension and bit error probability is also given in [46].

At first the algorithm constructs a matrix $\mathbf{X}$ with N rows from the received stream. This matrix hides the structure of a k-dimensional subspace of the original code $\mathbf{C}$. A sparse error matrix $\mathbf{E}$ is also added in $\mathbf{X}$. By finding the parity check equations from this received matrix it is possible to retrieve the original k-dimensional subspace.

A parity check equation correspond to a dual word h of the dual code $\mathbf{C}^{\perp}$. When a dual word $\mathbf{h}$ belongs to $\mathbf{C}^{\perp}$, the linear combination of columns $\mathbf{X}$ will be $\mathbf{h}\mathbf{X}^{\top}$. This vector of length N will have a low weight. The expectation of weight will depend on the bit error probability τ and the weight of the dual word $\mathbf{h}$. If $\mathbf{h}$ belongs to dual code $\mathbf{C}^{\perp}$, the expected weight will be

$$(1-(1-2\tau)^{w(\mathbf{h})})^{N/2}.$$

If $\mathbf{h}$ does not belong to $\mathbf{C}^{\perp}$, the expected weight will be $N/2$ with standard deviation $\sqrt{N/2}$. This difference between expected weights provides a way to recognize the dual words from the other words.

The algorithm will generate linear combinations of $\mathbf{X}$ that have a low weight. This is equivalent to generating low weight vectors $\mathbf{h}\mathbf{X}^{\top}$ in the code with generator matrix $\mathbf{X}^{\top}$. The algorithm has to perform two tasks. First it generates the dual words and then it eliminates the dual word candidates that do not satisfy detection criterion.

Detection criterion is a threshold for the probability $P(\mathbf{X}|H(\mathbf{h}))/P(\mathbf{X})$ where $H(\mathbf{h})$ is the hypothesis that the dual code contains the word $\mathbf{h}$. This is referred to as a first-order algorithm.

Higher-order algorithms consider the hypothesis that several words are in the dual code. For instance, second-order algorithm will consider the hypothesis $\mathbf{H}(\mathbf{h},\mathbf{h}')$.

In other words, the objective is to decide if a given two-dimensional subspace is a in included in the $\mathbf{C}^{\perp}$. Higher order algorithm should provide a better detection of words. This is because the 1's in $\mathbf{hX}^{\top}$ and $\mathbf{h}'\mathbf{X}^{\top}$ should be correlated.

The above algorithm is improved in [15]. Instead of performing statistical tests on the dual word candidates, the algorithm will try to decode using these dual words as parity checks. The principle of the used decoding is iterative decoding, as introduced in [27]. The feedback from the decoding will then be used to decide if a dual word candidate belongs to the dual code.

Both of the above algorithms, [46] and [15], consider primarily the case where the code's length and synchronization have already been recovered. In [17] it is shown that the same principles that can be used to recover the syncronization and length can also be used to reconstruct the code. Additionally in [17], two algorithms, [13] and [10], are used for optimizing the search for dual words in different scenarios. The problem of finding low weight words is also studied in a different context, for example in [29] or [43]. Finding a linear binary code word of a given weight is proven to be an NP-complete problem in [7].

These algorithms are designed to find the basis of the linear space defined by the code. That is, in a successful run of a algorithm, the used code is found. However, this does not mean finding the encoder. [15]

Different approach for block code identification than the one used in the algorithms above, is exhibited in [9]. Instead of searching for dual words, this algorithm is based on calculation of the rank of a matrix. This approach is further improved in [41], where the algorithm is made more resistant to noise. Both of these algorithms are detailed in the interleaver section of this chapter.

The decision problem whether a given sequence is coded by a linear block code is studied in [11]. Methods for identification of the code and classifying them is presented in [49], where also a detailed description of the implementation of the system is provided. The problem of how many code words are needed for the recovery of a code is examined in [19]. Generic approach for linear block code reconstruction is studied in [4]. Overview of different identification methods of block codes is presented in [16].

3.3 Convolutional Codes

Convolutional code reconstruction techniques for convolutional codes of any rate have been established in [23] and improved later in [3] and [20]. Using the identifi-

cation method presented in [23] for rate 1/2 convolutional code is considered here as an example. The highest degree of the encoder polynomial will be m and the message polynomial will have a degree $m + t$, where t is the time index and the constraint length is defined as $K = m + 1$. The identification method is based on solving the homogeneous linear system

$$\mathbf{Cf} = \mathbf{0},$$

where $\mathbf{C}$ is the matrix constructed from the known coded sequence, given by a $(t - m)$ by $2K$ matrix

$$\mathbf{C} = \begin{bmatrix} c_{2,m} & c_{2,m-1} & \cdots & c_{2,0} & c_{1,m} & c_{1,m-1} & \cdots & c_{1,0} \\ c_{2,m+1} & c_{2,m} & \cdots & c_{2,1} & c_{1,m+1} & c_{1,m} & \cdots & c_{1,1} \\ c_{2,m+2} & c_{2,m+1} & \cdots & c_{2,2} & c_{1,m+2} & c_{1,m+1} & \cdots & c_{1,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{2,t-1} & c_{2,t-2} & \cdots & c_{2,t-m-1} & c_{1,t-1} & c_{1,t-2} & \cdots & c_{1,t-m-1} \end{bmatrix}$$

and $\mathbf{f}$ is the length $2K$ coefficient vector of polynomials

$$\mathbf{f} = \Big[f_{1,0}, f_{1,1}, f_{1,2}, \ldots, f_{1,m}, f_{2,0}, f_{2,1}, f_{2,2}, \ldots, f_{2,m}\Big]^{\top}$$

that is to be recovered and $\mathbf{0}$ is the $(t - m)$ null vector.

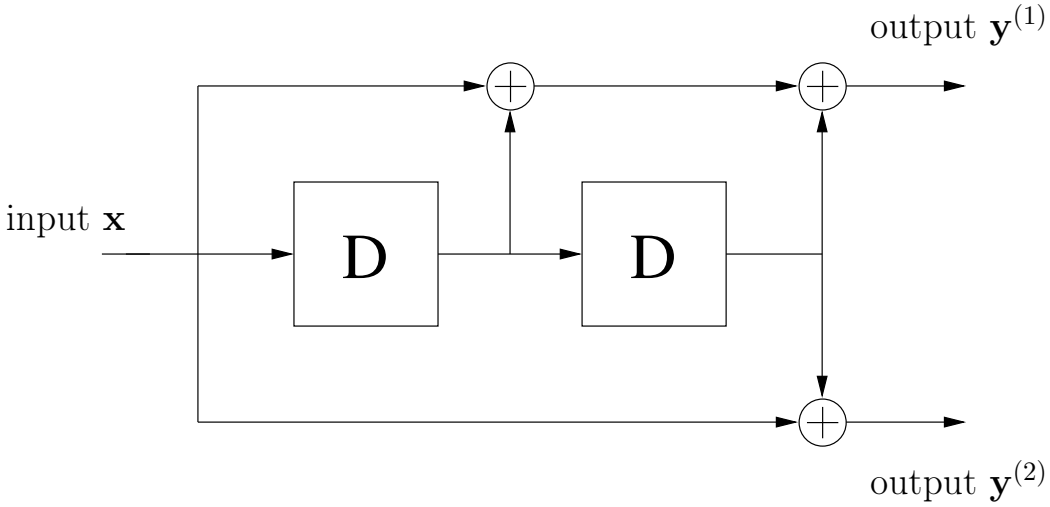

Figure 3.7: Convolutional encoder

For the encoder in Figure 3.7 with generator sequences $\mathbf{g}^{(1)} = (111)$ and $g^{(2)} =$

(101) and the received code message

$$\mathbf{c} = \mathbf{xG} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 11 & 10 & 00 & 01 & 01 & 00 & 10 & 11 \end{bmatrix}$$

$$= \begin{bmatrix} c_{1,0} & c_{2,0} & c_{1,1} & c_{2,1} & c_{1,3} & c_{2,3} & c_{1,4} & c_{2,4} & c_{1,5} & c_{2,5} & c_{1,6} & c_{2,6} & c_{1,7} & c_{2,7} \end{bmatrix},$$

that was generated with the unknown data message $\mathbf{x}$ and the unknown generator matrix $\mathbf{G}$, the equation will be

$$\begin{bmatrix} c_{2,2} & c_{2,1} & c_{2,0} & c_{1,2} & c_{1,1} & c_{1,0} \\ c_{2,3} & c_{2,2} & c_{2,1} & c_{1,3} & c_{1,2} & c_{1,1} \\ c_{2,4} & c_{2,3} & c_{2,2} & c_{1,4} & c_{1,3} & c_{1,2} \\ c_{2,5} & c_{2,4} & c_{2,3} & c_{1,5} & c_{1,4} & c_{1,3} \\ c_{2,6} & c_{2,5} & c_{2,4} & c_{1,6} & c_{1,5} & c_{1,4} \end{bmatrix} \begin{bmatrix} f_{1,0} \\ f_{1,1} \\ f_{1,2} \\ f_{2,0} \\ f_{2,1} \\ f_{2,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

After inserting the values this will be

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_{1,0} \\ f_{1,1} \\ f_{1,2} \\ f_{2,0} \\ f_{2,1} \\ f_{2,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

From this equation the null space can be obtained by Gaussian elimination and it

will be

$$\begin{bmatrix} f_{1,0} \\ f_{1,1} \\ f_{1,2} \\ f_{2,0} \\ f_{2,1} \\ f_{2,2} \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix},$$

which is the coefficient vector of the encoder that was wanted.

In a noiseless scenario the minimum number of intercepted bits in $1/n$ rate encoding is $L_{min} = n(3K - 2)$. This relation is a consequence of the requirement that the the rank of the interception matrix $\mathbf{C}$ must be lower than the number of coefficients in the encoder polynomials $\mathbf{f}$.

For noisy communications the reconstruction technique is essentially the same. When noise is present, the minimum required number of bits L_{min} is linked to the probability of noise per bit and is simply derived from the probability of finding one sequence of noiseless bits of length L_{min} as was required in the noiseless case.

Parameters (n, k, m) are searched for exhaustively. This is possible when these parameters are relatively small. Recovering the polynomials by solving the homogeneous linear system is done by Gaussian elimination, which is the most costly step. Complexity for this method is given as $O(K^4)$.

The reconstruction technique introduced in [23] is analyzed and improved further in [3]. This improved technique is then developed further by using algebraic description of convolutional codes in [20]. This enables the use of basic linear algebra and gets rid of polynomial operations. According to the authors the complexity of the reconstruction remains the same but the implementation of the algorithm is simplified. Algebraic description of convolutional codes is given in [38] and [25]. Identification of rate 1/2 convolutional codes and optimization of the identification process with Euclidean algorithm instead of Gaussian elimination is presented in [47]. Identification of convolutional code parameters in a noisy channel is considered in [22], where the identification process is done iteratively, based on the expectation maximation (EM) algorithm. Identification of $(n-1)/n$ encoders in a noisy environment is presented in [34], where the concept of dual codes of convolutional codes, studied in [26] and [36], is used in the iterative identification process. This method is developed further in [35] to include the case of punctured $(n-1)/n$ convolutional codes. Punctured convolutional codes are also studied in [24], [30] and [31]. A different way of detecting and reconstrucing convolutional code is suggested in [5], where the main

idea is based on counting collisions in the received sequence. A generic approach for convolutional code reconstruction is given in [4] and a more complete overview of identification of convolutional codes in [33].

3.4 Low-Density Parity-Check Codes

Identification of LDPC codes is essentially the same than identification of block codes. However, the long length and the low weight of a LDPC code word will require to select the used algorithms accordingly. The algorithms that are based on finding dual words, [46], [15] and [17], seem to work better for LDPC codes. This is due to the operation of these algorithms that generate dual words of small weight, matching the structure of LDPC code words. Identification of LDPC codes is described in more detail in the next chapter.

3.5 Turbo Codes

In this section different approaches to turbo code identification are presented. A description of three algorithms is given. The first one is based on finding the kernel of the polynomial matrix, the second one is based on finding dual words of the convolutional encoder and the third one is an iterative method based on using a decoder to recover the parameters of the turbo encoder. These three approaches cover most of the presented algorithms in literature.

A method presented in [2] is based on solving the kernel of the interception matrix and is founded on earlier work [23] on convolutional codes. The algorithm for reconstruction of convolutional encoder is developed further and two techniques for recovering the interleaver are proposed.

The asymptotic complexity of the whole process for a turbo encoder is $O(n^4)$ with convolutional encoders (n, k, m), where n is the code word length, k is the information word length and m is the degree of the encoder polynomials. The number of bits needed for reconstruction is stated as nk^2m. Complexity can be improved if the convolutional encoders are $(n, 1, m)$ encoders. In this case it is possible to use the Berlekamp-Massey algorithm [6], [37]. The achieved complexity will then be $O(m \log m^2)$. [2]

The method presented in [2] proceeds as follows. From a turbo encoder, as illustrated in Figure 3.8, parameters of the convolutional encoders C_1 and C_2 are recovered first. Then C_1' and C_2' are obtained which are $C_1 \cdot P_1$ and $C_2 \cdot P_2$ where

P_1 and P_2 are permutations that are introduced by the reconstruction algorithm. If the turbo encoder is systematic, it is possible to determine P_1 and C_1. A this point $\tilde{x_1}$ which is x_1 with noise and $\tilde{x_2}'$ which is $(P_2 \cdot P)(x_1)$ with noise are known. Finally $P' = P_2 \cdot P$ is computed and a turbo code encoder (C_1, C_2', P') is constructed and $\tilde{x_1}$ corrected into x_1 that was what was wanted in the first place. Next the whole process is presented in a detailed way.

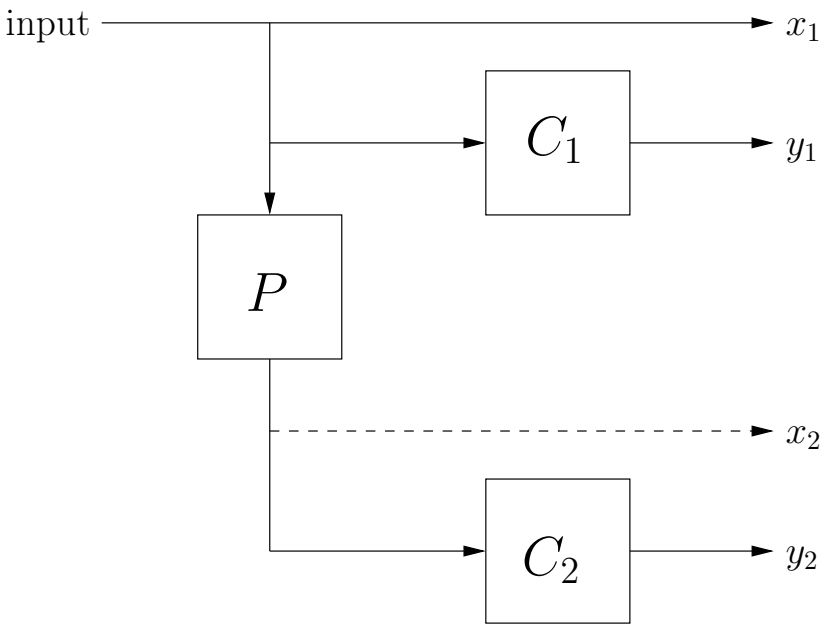

Figure 3.8: Turbo code encoder

Algebraic representation of convolutional encoders is used. Input message M is divided to k sub-messages M_i and output is divided into n sub-messages C_i. Each infinite sub-message is presented by formal series, $m_i(x)$ and $c_i(x)$, whose coefficients are values of the bits of the message. More details about the representation can be found in [38] and [25].

Following notation is used

$$\begin{array}{llrcl} \forall i = 1, ..., k, & & m_i(x) & = & \Sigma_{j \geq 0} m_i^j x^j, \\ \forall i = 1, ..., n, & & c_i(x) & = & \Sigma_{j \geq 0} c_i^j x^j, \\ \forall i = 1, ..., n & j = 1, ..., k, & f_{i,j}(x) & = & \Sigma_{l=0}^{m} f_{i,j}^l x^l. \end{array}$$

Algebraic representation of a (n, k, m)-encoder gives n equations describing the

encoder,

$$\begin{cases} m_1(x)f_{1,1}(x) & +\cdots+ & m_k(x)f_{1,k}(x) & = & c_1(x), & (E_1) \\ & \vdots & & & \vdots & \vdots \\ m_1(x)f_{n,1}(x) & +\cdots+ & m_k(x)f_{n,k}(x) & = & c_n(x). & (E_n) \end{cases}$$

From these equations can be built $(n-k)$ systems $(S_j)_{j=1,\ldots,n-k}$ of $k+1$ equations. Each (S_j) is equivalent to a $(k, k+1, m)$ sub-encoder. Next the reconstruction of (S_1) is presented. The technique for the other cases is the same.

$$(S_1) = \begin{cases} m_1(x)f_{1,1}(x) & +\cdots+ & m_k(x)f_{1,k}(x) & = & c_1(x) \\ & \vdots & & & \vdots \\ m_1(x)f_{k+1,1}(x) & +\cdots+ & m_k(x)f_{k+1,k}(x) & = & c_{k+1}(x) \end{cases}$$

Now $\mathbf{P}_1(x)$, the $(k+1)$-square matrix over the Laurent series field is defined by

$$\mathbf{P}_1(x) = \begin{bmatrix} f_{1,1} & \cdots & f_{1,k}(x) & c_1(x) \\ \vdots & & \vdots & \vdots \\ f_{k+1,1(x)} & \cdots & f_{k+1,k}(x) & c_{k+1}(x) \end{bmatrix}.$$

When $\Delta_i(x)$ is the minor of $\mathbf{P}_1(x)$ associated with $c_i(x)$, then

$$\sum_{i=1}^{k+1} c_i(x)\Delta_i(x) = 0,$$

which can be written in the following form

$$\forall j \geq km, \qquad \sum_{i=1}^{k+1}(\sum_{l=0}^{km} c_i^l \Delta_i^{km-l}) = 0. \tag{3.2}$$

By defining $\mathbf{\Delta}$ as the vector $(\Delta_1^0, \ldots, \Delta_1^{km}, \ldots, \Delta_{k+1}^0, \ldots, \Delta_{k+1}^{km})^\top$ and $\mathbf{G}$ as the $(k+1)(km+1)$-square matrix

$$\mathbf{G} = \begin{bmatrix} c_1^{km} & \cdots & c_1^0 & \cdots & c_{k+1}^{km} & \cdots & c_{k+1}^0 \\ c_1^{km+1} & \cdots & c_1^1 & \cdots & c_{k+1}^{km+1} & \cdots & c_{k+1}^1 \\ \vdots & & \vdots & \vdots & & \vdots & \\ c_1^{k(m(k+2)+1)} & \cdots & c_1^{k(m(k+1)+1)} & \cdots & c_{k+1}^{k(m(k+2)+1)} & \cdots & c_{k+1}^{k(m(k+1)+1)} \end{bmatrix}$$

it is possible to write (3.2) as $\mathbf{G}.\mathbf{\Delta} = \mathbf{0}$.

Then $\mathbf{\Delta}$ is computed by Gaussian elimination process. It can be noticed that $\Delta_{(k+j)}$ is common to all $\mathbf{P}_j(x)$. Now $\mathbf{\Delta}$ can be determined uniquely by comparing kernels for all S_j.

Next step in the reconstruction is to recover the coefficients of the encoder polynomials. This is done in a similar way as above. For all $j = 1, \ldots, k$ $\mathbf{Q}_1^j(x)$ is defined by

$$\mathbf{Q}_1^j(x) = \begin{bmatrix} f_{1,1}(x) & \ldots & f_{1,k}(x) & f_{1,j}(x) \\ \vdots & & \vdots & \vdots \\ f_{k+1,1}(x) & \ldots & f_{k+1,k}(x) & f_{k+1,j}(x) \end{bmatrix}.$$

Here $\det(\mathbf{Q}_1^j(x)) = 0$ and the equation can be developed as previously as

$$\forall j = 1, \ldots, k \qquad \sum_{i=1}^{k+1} \Delta_i(x) f_{i,j}(x) = 0.$$

Now defining the vector

$$\mathbf{f}_j = \left[f_{1,j}^0, \ldots, f_{1,j}^m, \ldots, f_{k+1,j}^0, \ldots, f_{k+1,j}^m \right]^\top$$

and the $(k+1)(m+1)$ square matrix

$$\mathbf{D} = \begin{bmatrix} \Delta_1^m & \ldots & \Delta_1^0 & \ldots & \Delta_{k+1}^m & \ldots & \Delta_{k+1}^0 \\ \Delta_1^{m+1} & \ldots & \Delta_1^1 & \ldots & \Delta_{k+1}^{m+1} & \ldots & \Delta_{k+1}^1 \\ \vdots & & \vdots & & \vdots & & \vdots \\ \Delta_1^{(k+1)(m+2)-1)} & \ldots & \Delta_1^{(k+1)(m+1)-1} & \ldots & \Delta_{k+1}^{(k+1)(m+2)-1} & \ldots & \Delta_{k+1}^{(k+1)(m+1)-1} \end{bmatrix},$$

the equation can be written as

$$\forall j = 1, \ldots, k \qquad \mathbf{D}.\mathbf{f}_j = \mathbf{0}.$$

It can be noticed that for all $i = 1, \ldots, k+1$ and $j > km$, $\Delta_i^j = 0$. Then, for all $j = 1, \ldots, k+1$, $\mathbf{f}_j$ is in the kernel of $\mathbf{D}$. Computation is done by Gaussian elimination. Now it can be noticed that for all $i, j = 1, \ldots, k$, $f_{i,j}(x)$ are common to all $\mathbf{Q}_i^j(x)$. By comparing the kernels for all (S_i) together, a set $\{(f_{1,j}(x), \ldots, f_{n,j}(x))$ so that $j =$

$1, \ldots, k\}$, can be determined uniquely. The $f_{i,j}(x)$ are then determined with random permutation of j. The permutation is introduced by the reconstruction process, but it can be determined if the convolutional code is systematic.

The method presented here can also be used with noisy channels. When there are errors in the intercepted stream it is proposed to detect a sequence of consecutive bits without noise, in the same manner as with convolutional codes in [23]. Result can be easily tested with a simple test. With a false result, only half of the equations produce zero over all of the transmission.

When the turbo code encoder is systematic, it is possible to determine the interleaver. After the structure of the convolutional encoders is known, the streams x_1 and x_2 can be obtained. These streams are the input and the output of the interleaver that is to be recovered.

In [2] two methods are proposed. The first one is based on a trie structure and the second one is based on a vote algorithm. Both methods are presented next.

A trie is a tree structure that distinguishes words from their prefixes. Each leaf is labeled with a word that is different from every other word. In the level L depth, each node is labeled with words that have the same prefix. Trie with depth D means that every word at level D is distinguished from other words. For example a trie for binary words a,b,c and d, as in Table 3.1, would be constructed like in Figure 3.9. Further information about tries can be found in [14].

Table 3.1: Binary words.

a	b	c	d
1	1	0	1
1	0	1	0
1	0	0	1

Tries can be used to recover the interleaver structure in the following way. For example, in Table 3.2 can be seen the sequence of words $x_1(0) = \begin{bmatrix}0 & 1 & 1\end{bmatrix}$, $x_1(1) = \begin{bmatrix}0 & 0 & 1\end{bmatrix}$, $x_1(2) = \begin{bmatrix}1 & 0 & 1\end{bmatrix}$ and so forth and the interleaved words $x_2(t)$ when an interleaver

$$\mathbf{\Pi} = \begin{bmatrix}0 & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$$

was used. The associated trie can be seen in Figure 3.10. By comparing the leaves of both tries, the permutation can be determined to be $x_1^1(t) = x_2^2(t)$, $x_1^2(t) = x_2^3(t)$

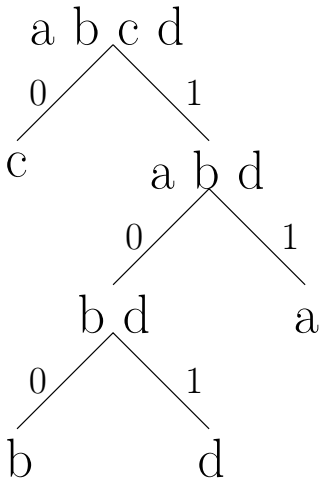

Figure 3.9: Trie structure

Table 3.2: Words and interleaved words

t	$x_1^1(t)$	$x_1^2(t)$	$x_1^3(t)$		$x_2^1(t)$	$x_2^2(t)$	$x_2^3(t)$
0	0	1	1		1	0	1
1	0	0	1		1	0	0
2	1	0	1		1	1	0
⋮	⋮	⋮	⋮		⋮	⋮	⋮

and $x_1^3(t) = x_2^1(t)$. This method is intended for a noiseless channel.

A second method is proposed for a noisy channel. it is based on a voting algorithm. An event E_{ij} is defined as "$x_1^i = x_2^j$". If the tested bit permutation is not correct, so that $x_1^i \neq x_2^j$, then $P(E_{ij}) = \frac{1}{2}$, and if $x_1^i = x_2^j$ then $P(E_{ij}) = (1 - \tau)$ where τ is the probability of a bit error. The vote V_{ij} will be such that for every event E_{ij} there will be vote $V_{ij} = V_{ij} + 1$ or in the other case $V_{ij} = V_{ij} - 1$.

Each V_{ij} is an independent random variable that will follow binomial probability distribution with parameter $\frac{1}{2}$ or τ. After N observations the mean will be either $\mu(V_{ij}) = (1 - 2\tau)N$, if the permutation is correct, or $\mu(V_{ij}) = 0$ if it is not. Experimental results from test cases with this algorithm are also presented in [2].

A different approach to the reconstruction of turbo codes is presented in [18], where two different algorithms are introduced. These algorithms are able to recover the permutation of a turbo code without puncturing in channels with high noise. The first algorithm is based on the idea of dual words for turbo codes and the second algorithm is an iterative method based on using a decoder for reconstruction.

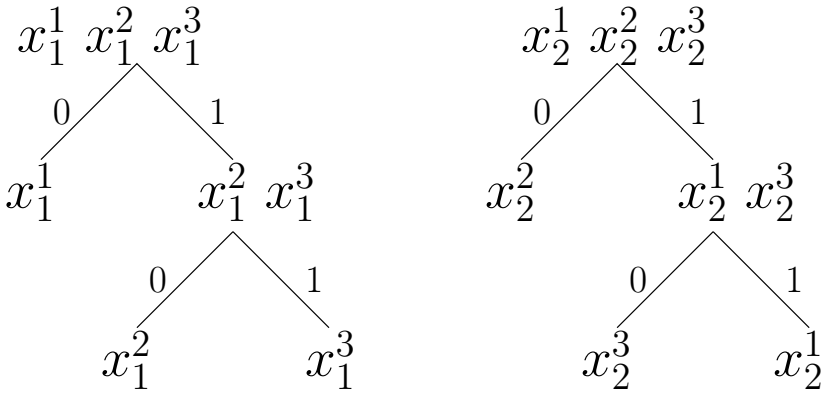

Figure 3.10: Comparison of tries

The basic idea of reconstruction that is based on the dual words of the convolutional code is similar to block code reconstruction. A convolutional code is defined by the generator matrix of polynomials or rational fractions and the dual code is the vector space spanned by the vectors orthogonal to the generator matrix. A dual word is an element of this vector space. A dual word will also correspond to a parity check equation, as in the case with linear block codes. An analysis of dual codes of convolutional codes is developed in [26] and studied with a view on code identification in [36].

The reconstruction of a turbo encoder, as shown in Figure 3.11, is achieved after recovering the parameters P', Q', P, Q and Π. The reconstruction begins with isolating the outputs X, Y and Z. This is done by searching the parity check equations that are valid for any multiple of n, where $n = 3$ in standard interleaving. These found parity check equations will only involve bits from X and Y, so it is possible to isolate Z. The P' and Q' can be recovered with convolutional code identification methods, or they can be solved by exhaustive search because of the low degree of the polynomials in turbo codes.

The low degree of polynomials P and Q in a conventional turbo encoder limits the number of possible encoders and consequently permits these polynomials to be recovered in an exhaustive search. On the other hand, the size N of the permutation Π can be very large. Therefore the most difficult task can be recovering the permutation.

For a turbo code with a systematic convolutional encoder with parameter P/Q, the dual words will have a form $(\lambda P, \lambda Q)$, where λ is any polynomial. For example, turbo code with $P/Q = 1+D^2+D^3/1+D+D^2$ will have a dual word $(1+D^2+D^3, 1+$

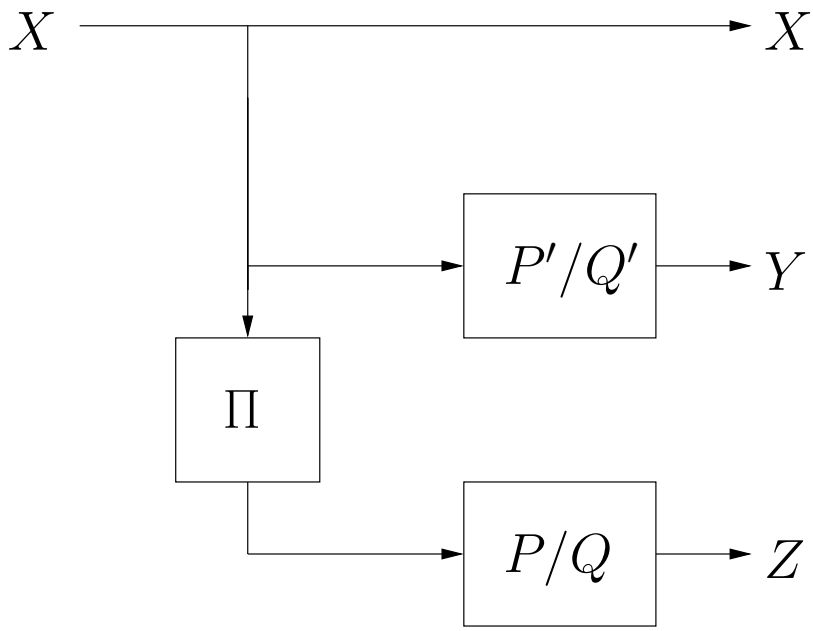

Figure 3.11: A systematic parallel turbo code

$D + D^2$). In this example the dual word has $\lambda = 1$. As the dual word corresponds to a parity check, for all binary code words the sum of these bit positions will be zero modulo 2.

The actual search for dual words is made in the same manner as the search for dual words in block codes. A matrix of M rows and $2N$ columns is constructed. M is the number of intercepted code blocks and the $2N$ corresponds to N columns from the information output X and N columns from the turbo coded output Z. Dual words of very low weight are then searched from this matrix. Methods presented in [17] can be used for optimization of the search.

Each parity check equation $(\lambda P, \lambda Q)$ that is found will have a form λQ on the turbo coded output Z and λP on the permuted information output X_Π. Only information output X is available, so only weight w of the λP is known. Each found parity check will therefore give a information pair $(w, \lambda Q)$. This pair can only coincide with some encoders P/Q.

To find the correct encoder P/Q, all polynomials λ up to a degree are listed and all multiples $(\lambda P, \lambda Q)$ are calculated. From these multiples the pairs $(w, \lambda Q)$ with low weight are stored. It is then possible to compare the found dual word with the list of $(w, \lambda Q)$ pairs. Every found dual word will limit the number of possible encoders P/Q until the resolution on the correct P/Q is reached. When P and Q

are of degree 3 or less, multiples of weight up to 6 can uniquely classify all pairs (P, Q) [18].

Each pair $(w, \lambda Q)$ will also give out part of the permutation Π. There is a possibility that some permutation position belongs to a dual word that is not found in the search. Then parts of the permutation are left unknown. The probability of this happening will increase with higher channel noise.

This method has a time complexity $O(N^{\lceil \frac{w}{2} \rceil})$ and memory complexity $O(N^{\lceil \frac{w}{4} \rceil})$. These results are achieved when the search of dual words is optimized using the search technique given in [13]. The same optimization technique is also presented in the error correction code identification context in [17].

The second algorithm in [18] is based on the possibility to distinguish entropies of coded message and random message. This algorithm takes noisy outputs X and Z as a input and outputs the positions of permutation Π. The fraction P/Q is needed for running this algorithm and the algorithm will go through all possible encoders P/Q to find the correct one. The number of different possibilities for P and Q is quite small as degree of polynomials P and Q is usually quite low for turbo codes so exhaustive search will be possible.

The algorithm proceeds iteratively from the first permutation position to the last. Previous steps are assumed successful and only the next needed position is considered in reconstruction process. The BCJR decoding algorithm [1], which is used for decoding turbo codes, is used to distinguish the most probable permutation position $\Pi(x)$.

Forward probabilities are used in the iteration step of the algorithm. F_i is the forward probability on the state of the encoder when i first couples of noisy information bits and redundancy bits have been used. F_i' is the random variable that corresponds to the forward probability of BCJR algorithm after $i-1$ couples of noisy information and redundancy bits and then a couple with a random bit and i-th noisy redundancy bit. The objective is to distinguish the F_i from F_i' and entropy is used for that task. Entropy of the distribution is $H(F_i) = \Sigma_a - F_i(a) \log F_i(a)$, where $F_i(a)$ is the forward probability that the i-th state of the encoder is a. When the channel is not too noisy, the two distributions, $H(F_i)$ and $H(F_i')$, will be sufficiently distinct from each other. Distributions of entropies for fraction $P/Q = \frac{1+D^2}{1+D+D^2}$ is plotted in Figure 3.12.

Iterative method for recovering the permutation is also used in [21]. The demonstrated technique works with noisy code words for small interleaver block size and is limited to rate 1/3 turbo codes with unpunctured rate 1/2 systematic convolutional

encoders. Iterative approach for recovering the turbo code interleaver is developed further in [45], where more efficient use of the available information results in significantly shorter execution time and fewer code words needed for reconstruction.

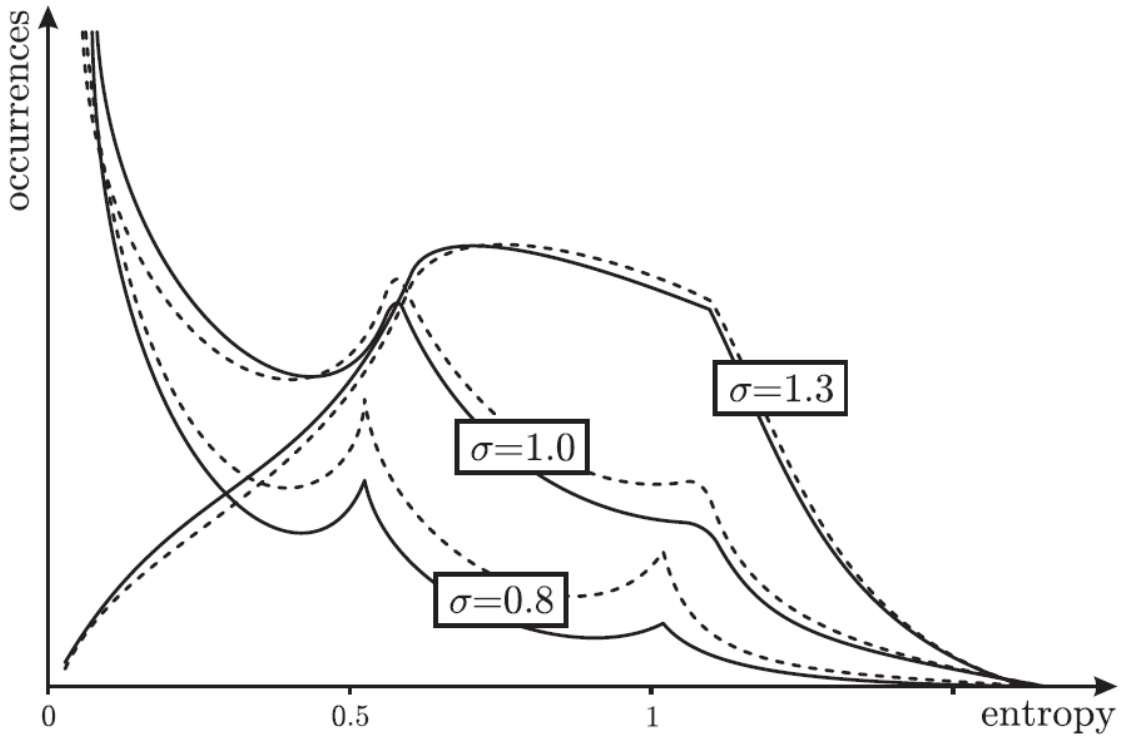

Figure 3.12: Distributions of entropies $H(F_i)$ (plain line) and $H(F_i')$ (dashed line) for gaussian channels with different standard deviations. [18]

Chapter 4

Simulations and Case Studies

The purpose of this chapter is to show what kind of simulations were made and report the results of these simulations. This chapter also includes discussion on the reason for implementing the chosen algorithms for the chosen codes.

4.1 Simulation Setup

Error correction codes that were chosen to the simulation were LDPC codes. These codes were chosen because they are frequently used in modern telecommunications applications.

For example, LDPC codes are currently used in ETSI standards DVB-S2X and DVB-S2 for satellite communicution, DVB-T2 and DVB-T2-Lite for terrestrial television broadcasting and DVB-C2 for cable transmission. LDPC codes are also used in IEEE standards 802.3 (10 GBASE-T), 802.11 (Wifi), 802.15.3c (60 Ghz PHY), 802.16 (WiMAX) and 802.22 (WRAN), near earth and deep space communications, and Chinese broadcasting standards. Furthermore, LDPC codes are also included in proposals for future standards.

4.1.1 Identification of LDPC codes

This section presents the most important steps of simulation of identification of error correcting codes. Identification algorithms can have various parameters, variables, premises and presumptions. This section will explain how the simulation of LDPC code identification was performed. The identification algorithm was implemented according to the method proposed in [17].

Identification algorithm and simulations were done in the MATLAB environ-

ment. In these simulations pseudorandom Gallager-type LDPC parity check matrices were created with selected parameters. These parameters are information word length, code word length and column weight of the LDPC matrix. In these simulations the code word's length is two times the information word's length, that is to say that the LDPC code rate is 1/2.

Information words were randomly generated, and this random data is then encoded with the LDPC encoder. The code will be systematic, so the code words will always have the information word with the parity check bits concatenated in the end. The number of generated code words can vary and this will have an impact on the result of the identification algorithm, as could be expected. The coded sequence is sent through a binary symmetric channel with a cross over probability τ. The channel condition will have an effect on the identification as well.

The identification algorithm will go through all parity checks of a given weight. LDPC codes typically have low weight parity checks, so the amount of parity checks generated can be manageable. These generated parity checks of a given weight will then be checked against all the received code words to see which of them are dual words of the received code words. Dual words will have a zero scalar product with the code words. Without channel noise dual words will be the ones with zero scalar product against all the received words. If there is channel noise, a threshold can be used to decide which of the generated parity checks are dual words. This threshold will be tested against the sum of the generated parity checks' product with all the received words.

The weight of the product of received words with generated parity checks, $G \times h$, will follow the binomial distribution centered on $\frac{1-(1-2\tau)^w}{2}M$, where τ is the bit error probability, w is the word weight and M is the number of received words as is explained in [17], when the correct word length and synchronization is found. Otherwise the weight of the product $G \times h$ will be centered on $M/2$.

Received code words do not necessarily define the original code's vector space uniquely. This will have an effect on the number of dual words found. If there are too few received code words, a large number of dual words to the received code words will be found. The number of received code words to get stable results for dual words is quite low, but is affected also by the channel crossover probability.

4.2 Results

This section describes the obtained results from the simulations. Different scenarios are presented in order to describe the behaviour of the algorithm.

The average number of found dual words against different number of received code words is plotted in Figure 4.1. The parity checks have a length of 40 bits and weight of 4. The average is calculated from ten different computations with different codes in each computation. The number of found dual words will stabilize after about 20 received code words as can be seen in the Figure 4.2. In this figure the found dual words are plotted against the number of received words from 14 to 40. The lower line at 20 marks the number of rows in the LDPC matrix of the code, that is the number of parity checks of the LDPC code. The number of found dual words rises exponentially when there is too few received code words available. If less words are received, the number of found dual words will be high as there will be less constraints on the dual words.

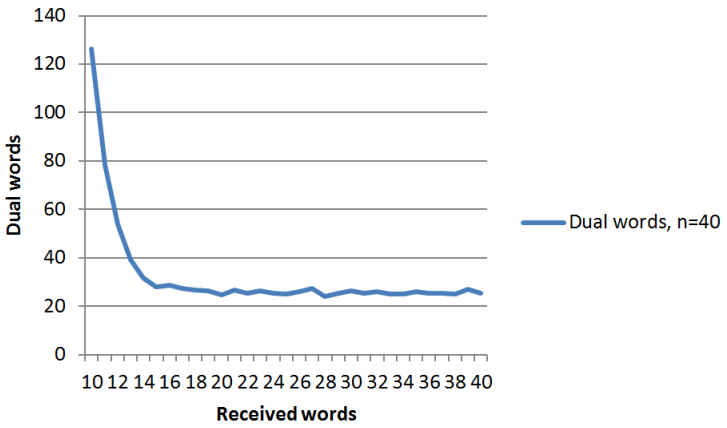

Figure 4.1: Dual words, word length=40, weight=4

The number of found dual words against the number of received code words follows a similar pattern for longer code words as well. The average number of found dual words with a length 80 and a weight 4 code words is depicted in Figure 4.3. The stabilization of found dual words happens after about 21 received code words. The number of found dual words after stabilization for a range of received code words can be seen in Figure 4.4, where the lower line at 40 is the number of parity checks in the original LDPC matrix. Similar graphs are plotted for code words of length 160 and weight 4 in Figure 4.5 and Figure 4.6. From these graphs it can be seen that the stabilization happens a little bit later, at about 23 received words.

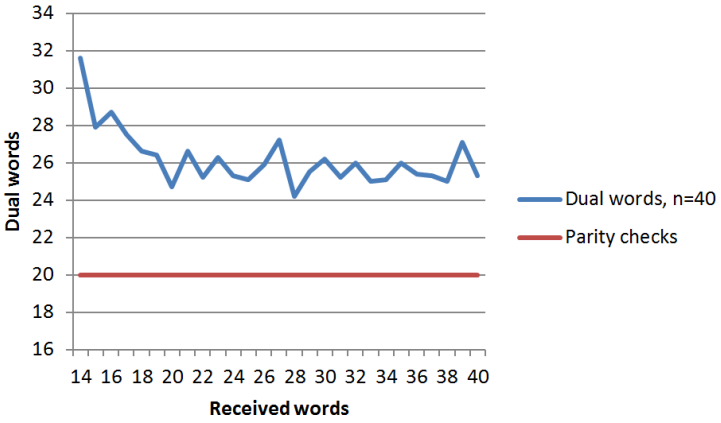

Figure 4.2: Dual words, wordlength=40, weight=4

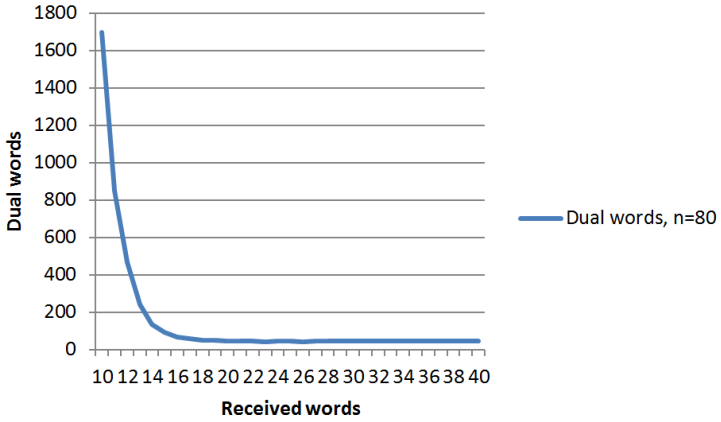

Figure 4.3: Dual words, word length=80, weight=4

Channel noise will cause some bits of the received words to be erroneous. Dual word for such an erroneous word will not necessarily be in the dual of the original code. A product of a dual word with every received code word will produce zero in an error free transmission, but when there is channel noise, the product of a dual word with a received erroneous code word might not be zero.

The algorithm computes the product of a dual word candidate with every received code word. These products are then added together. When there is no channel noise, the sum will be zero. In the presence of channel noise, there will be errors in the received code words and the sum might not be zero anymore. This happens when the received code word has an odd number of errors in the positions of the dual word's support. Support of a word is the positions of ones in a binary

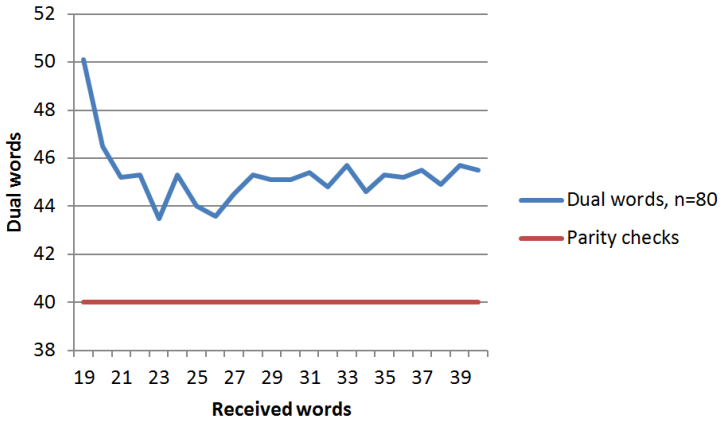

Figure 4.4: Dual words, word length=80, weight=4

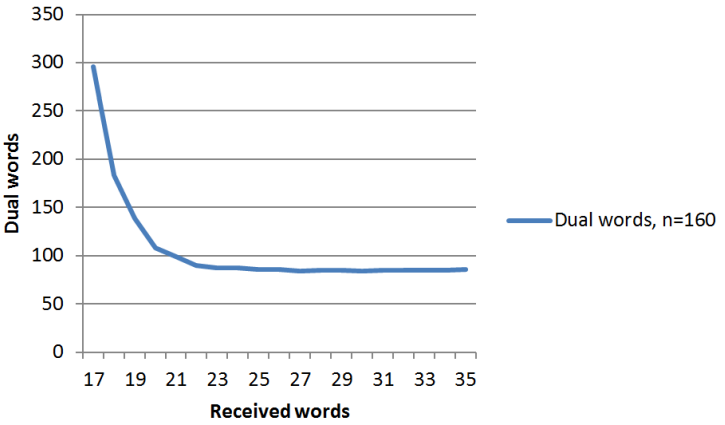

Figure 4.5: Dual words, word length=160, weight=4

code word.

With channel noise the algorithm has to have a higher selection threshold for the dual words than the zero sum. The threshold has to be selected so that the algorithm misses as few dual words as possible. The threshold must also leave out the words that are not in the dual but still have a small sum.

One proposed method for setting the threshold is to select the threshold more than three standard deviations away from the center of distribution of both the sum of products for a random word and the dual word [17]. The recommended separation

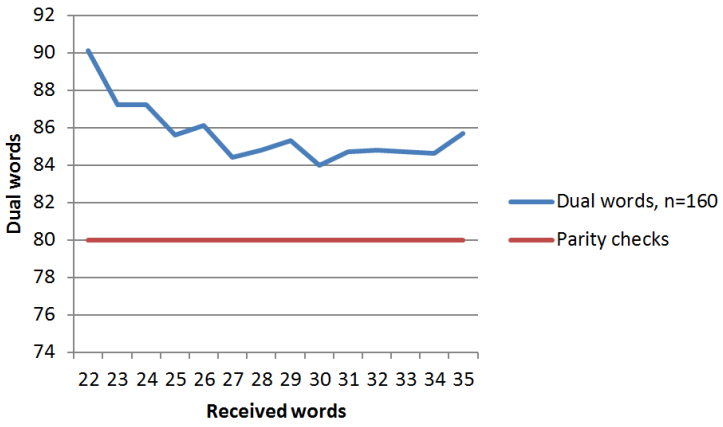

Figure 4.6: Dual words, word length=160, weight=4

of distributions is achieved when the number of received code words M is [17]

$$M > \left(\frac{3\sqrt{1-(1-2\tau)^{2w}}+1}{(1-2\tau)^{w}} \right)^{2},$$

where τ is the error rate and w is the weight of the word. In this case it is advised to select the threshold T in the middle of the interval as

$$T = \frac{M}{2}\left(1 - \frac{(1-2\tau)^{w}}{2}\right) + 3\frac{\sqrt{M}}{4}(\sqrt{1-(1-2\tau)^{2w}} - 1).$$

The selection of used threshold can have some effect on the performance of the algorithm and different thresholds could be selected according to the situation.

Channel noise's effect on the algorithm is illustrated in Figure 4.7. There are 300 received code words with a length of 40 and a weight of 4 and a bit error rate $\tau = 0.001$. The upper line is the number of found dual words and the lower line is the number of parity checks that are also in the LDPC parity check matrix that was used for the encoding. When the channel noise is higher, more received words are needed for the number of parity checks to stabilize. This stabilization can be seen in Figure 4.8 with $\tau = 0.01$ and in Figure 4.9 with $\tau = 0.1$.

In the presence of noise, more received code words are required for longer code words for identification. This can be seen in Figure 4.10 where stabilization of found dual words happens significantly later for longer code words. Without noise the difference was only few words.

The algorithm will usually find more parity checks than there are in the original

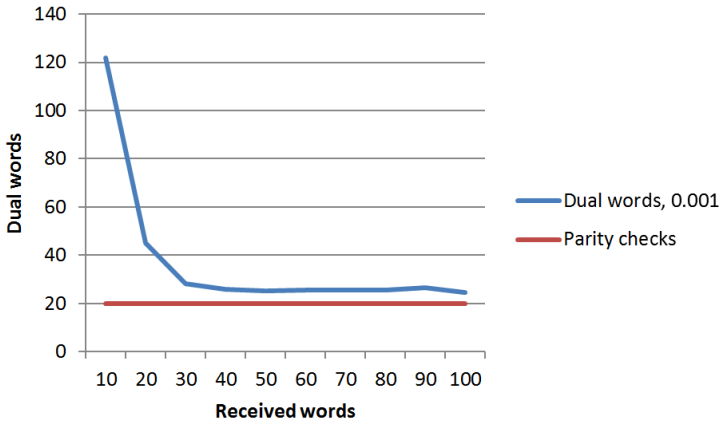

Figure 4.7: Effect of channel noise, $\tau = 0.001$

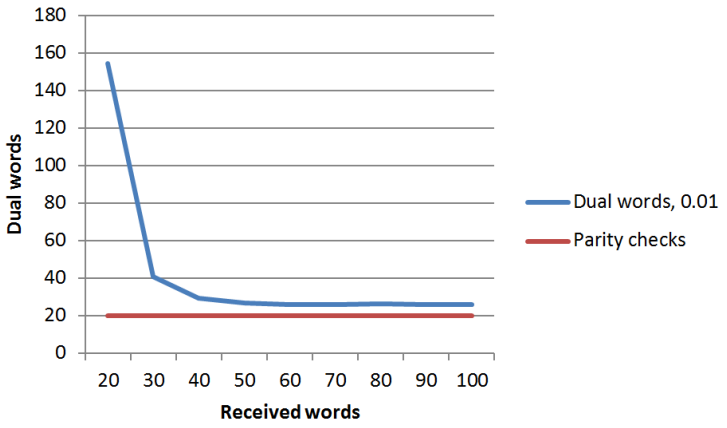

Figure 4.8: Effect of channel noise, $\tau = 0.01$

LDPC matrix. Even with more received words some parity checks will remain. This can be seen in Figure 4.11.

A LDPC matrix will have the same column sum over all columns. However, when the algorithm finds parity checks that are not in the original matrix, the column sum will be different for some columns. Using this information it is possible to remove these extra parity checks.

An algorithm that was used for removing extra parity checks will be such that it goes through all found parity checks and includes only those that have at least one correct column sum in the support of the parity check. This method will leave out those parity checks that have too high column sum for every position of ones in the parity check vector. This operating principle is illustrated in Figure 4.12, where the

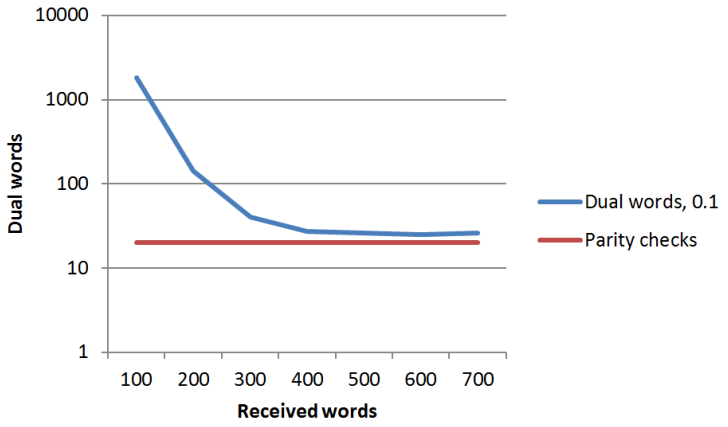

Figure 4.9: Effect of channel noise, $\tau = 0.1$

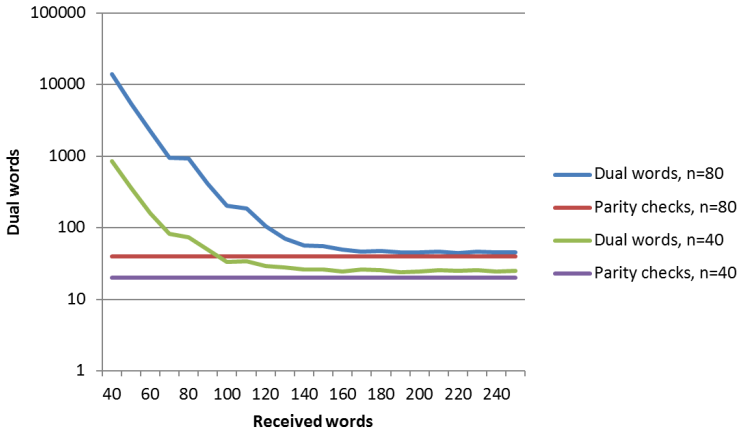

Figure 4.10: Dual words, different word lengths, $\tau = 0.05$

squares mark the ones in the parity checks and in the bottom is the column sums that should add to two in this example. Here it can be seen that the extra parity check can be spotted as the only one that has too high column sum in the whole support.

In some cases there is still a possibility that the extra parity check has its support positioned so, that it is impossible to distinguish the original parity check. This situation is depicted in Figure 4.13. Here it happens that there are two parity checks that have too high column sum for the whole support of the parity check.

In simulations it was found that the number of found extra parity check seems to remain constant even when the code word length grows. In realistic LDPC codes the word length is usually very high compared to the weight of the code word, so

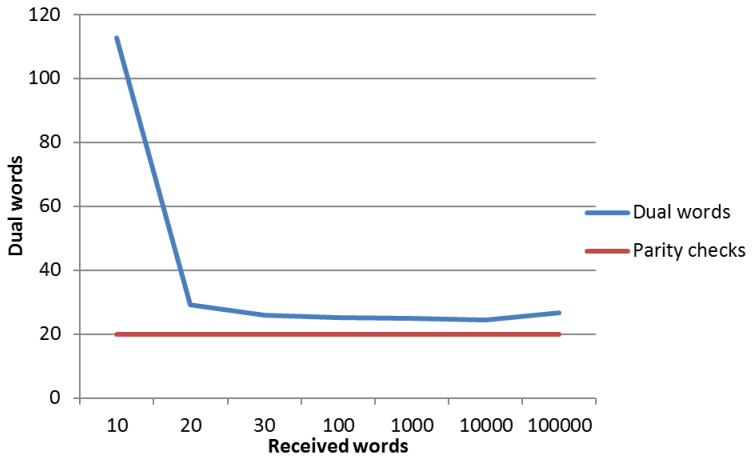

Figure 4.11: Effect of number of received words, $\tau = 0$

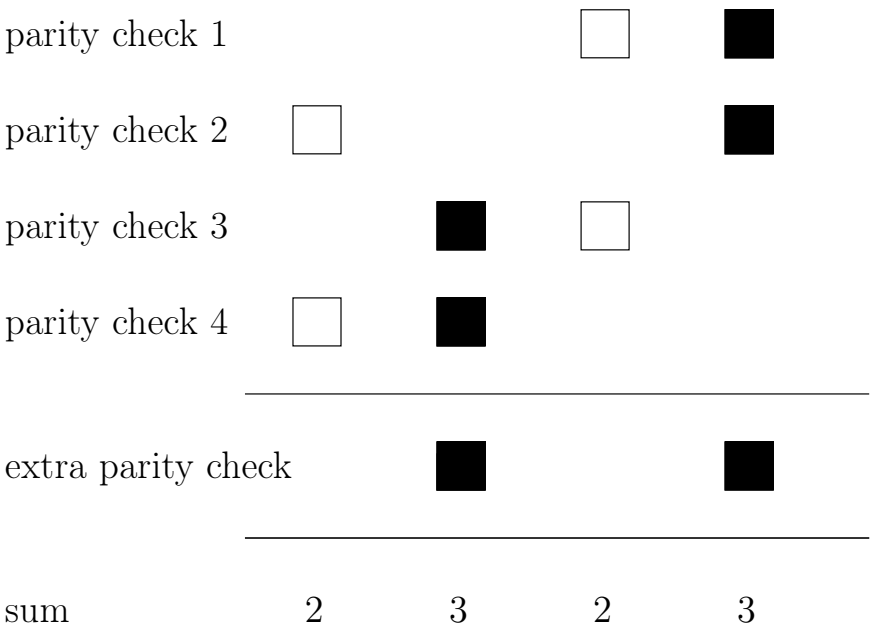

Figure 4.12: Parity check can be removed.

there should be smaller chance of finding extra parity checks like in the situation in Figure 4.13. Also the weight of the code word can be higher than two, so in this way the probability of finding undistinguishable extra parity checks is also lowered. In 20 runs of the algorithm with a code word length 40, the average number of dual words that were found after removing all the uncertain ones, was 16.7 out of 20. With a longer code word of 80 bits the average number of found parity checks that were in the LDPC matrix was 38.1 out of 40. In this case 15% of the runs found all

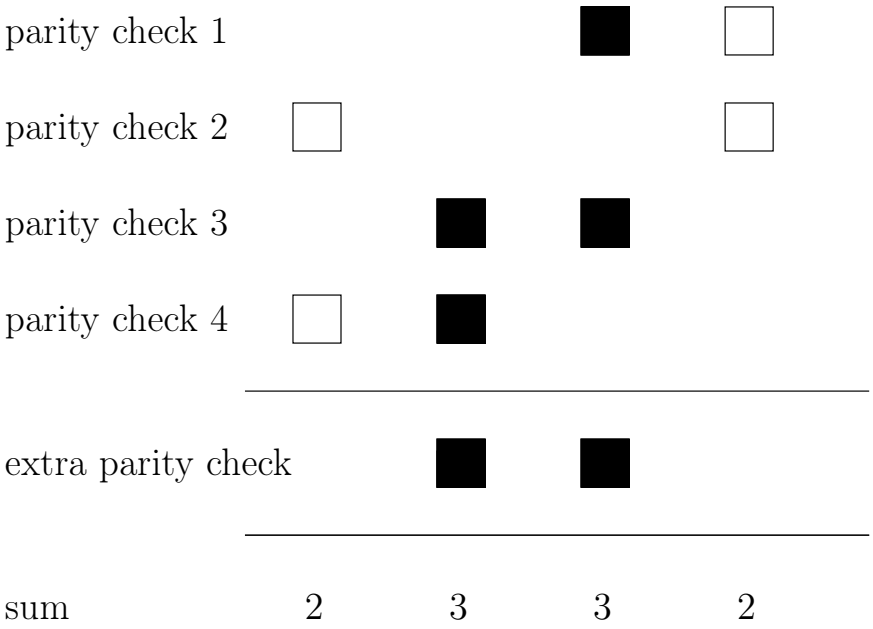

Figure 4.13: Parity check can not be removed.

parity checks.

Chapter 5

Conclusions

In this chapter a summary of the work is presented. Results that were presented in the previous chapter are evaluated, limits of the work are pointed out and areas of future research are discussed.

The objective of this thesis was to present an overview of different identification methods of forward error correcting codes in a non-cooperative context and to implement a test bed for testing these methods and principles in a controlled laboratory setting. The main focus was set on methods that are designed for modern forward error correcting codes.

In the first chapter the objectives for this study were stated and the focus and scope of the thesis were considered. Background and the need for this type of work were discussed. Also some possible applications were mentioned.

In the second chapter operating principles of error correction codes were explained for block codes and convolutional codes. LDPC and turbo codes were given as examples of those two types of currently used codes. Interleavers and a channel model were also presented as they apply to this work.

In the third chapter the identification methods were studied. These methods are presented as they are laid out in literature. Methods for block codes and convolutional were examined. with an emphasis on LDPC and turbo codes. Identification of interleaver structure is also presented as it is a important element of code identification especially with turbo codes. The block code identification methods that seem to work better for LDPC codes were those that were based on finding dual words. For turbo codes different identification methods include finding the dual words, retrieving the matrix kernel or iterative methods.

In the fourth chapter simulation and test case results were presented. In different exhibited scenarios the behaviour of LDPC code identification is explained. Param-

eters that are the most crucial in identification of LDPC code identification, code word length, number of intercepted code words, channel condition, were examined. For situations where too many parity checks are found, an algorithm was developed to retrieve the original parity checks.

In this work it has become clear that error correction coding is not the optimal way to achieve information security. The error correction code can be made more difficult to identify but encryption methods will provide a higher information security without making the encoding and decoding procedures too demanding.

The identification methods for other codes than presented here could be studied further. The methods presented in his study are designed for situation where only the type of code is know, for example turbo code. In a practical application there might be more information available about the used code. This information could be used to optimize the identification algorithm. Other relevant area for future research could be the application of these methods in a software defined radio system.

References

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (correspondence). *IEEE Transactions on Information Theory*, 20(2):284–287, March 1974.

[2] J. Barbier. Reconstruction of turbo-code encoders. In S. A. Zoltowski M. D. Singh R. Miller S. P. Rao, R. M. Dianat, editor, *Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE) Digital Wireless Communications VII and Space Communication Technologies*, volume 5819, pages 463–473, June 2005.

[3] J. Barbier. *Analyse de canaux de communication dans un contexte non coopératif.* PhD thesis, École polytechnique, Palaiseau, Essonne, France, November 2007.

[4] J. Barbier, G. Sicot, and S. Houcke. Algebraic Approach for the Reconstruction of Linear and Convolutional Error Correcting Codes. *International Journal of Applied Mathematics and Computer Sciences*, 2(3):113 – 118, 2006.

[5] M. Bellard and J.-P. Tillich. Detecting and reconstructing an unknown convolutional code by counting collisions. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2014)*, pages 2967–2971, June 2014.

[6] E. Berlekamp. Nonbinary BCH Decoding (Abstract). *IEEE Transactions on Information Theory*, 14(2):242, March 1968.

[7] E. Berlekamp, R. J. McEliece, and H. C. A. Van Tilborg. On the Inherent Intractability of Certain Coding Problems. *IEEE Transactions on Information Theory*, 24(3):384–386, May 1978.

[8] R. E. Blahut. *Algebraic Codes for Data Transmission.* Cambridge University Press, Cambridge, UK, 2003.

[9] G. Burel and R. Gautier. Blind Estimation of Encoder and Interleaver Characteristics in a Non Cooperative Context. In *Proceedings of the Second IASTED International Conference on Communications, Internet, and Information Technology*, pages 275–280, 2003.

[10] A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, January 1998.

[11] C. Chabot. Recognition of a code in a noisy environment. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2007)*, pages 2211–2215, June 2007.

[12] S. K. Chilappagari, D. V. Nguyen, B. Vasicž, and M. W. Marcellin. Girth of the Tanner graph and error correction capability of LDPC codes. In *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 1238–1245, September 2008.

[13] P. Chose, A. Joux, and M. Mitton. Fast Correlation Attacks: An Algorithmic Point of View. In L. R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 209–221. Springer Berlin Heidelberg, 2002.

[14] J. Clément, P. Flajolet, and B. Vallée. Dynamical Sources in Information Theory: A General Analysis of Trie Structures. *Algorithmica*, 29:307–369, 1999.

[15] M. Cluzeau. Block code reconstruction using iterative decoding techniques. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2006)*, pages 2269–2273, July 2006.

[16] M. Cluzeau. *Reconstruction d'un schéma de codage*. PhD thesis, École Polytechnique, Palaiseau, Essonne, France, October 2006.

[17] M. Cluzeau and M. Finiasz. Recovering a code's length and synchronization from a noisy intercepted bitstream. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2009)*, pages 2737–2741, June 2009.

[18] M. Cluzeau, M. Finiasz, and J.-P. Tillich. Methods for the Reconstruction of Parallel Turbo Codes. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2010)*, pages 2008–2012, June 2010.

[19] M. Cluzeau and J.-P. Tillich. On the code reverse engineering problem. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2008)*, pages 634–638, July 2008.

[20] M. Côte and N. Sendrier. Reconstruction of convolutional codes from noisy observation. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2009)*, pages 546–550, June 2009.

[21] M. Côte and N. Sendrier. Reconstruction of a turbo-code interleaver from noisy observation. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2010)*, pages 2003–2007, June 2010.

[22] J. Dingel and J. Hagenauer. Parameter Estimation of a Convolutional Encoder from Noisy Observations. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2007)*, pages 1776–1780, June 2007.

[23] E. Filiol. Reconstruction of Convolutional Encoders over GF(q). In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, pages 101–109, London, UK, 1997. Springer-Verlag.

[24] E. Filiol. *Techniques de reconstruction en cryptologie et théorie des codes*. PhD thesis, École Polytechnique, Palaiseau, Essonne, France, March 2001.

[25] G. D. Forney Jr. Convolutional Codes I: Algebraic Structure. *IEEE Transactions on Information Theory*, 16(6):720–738, November 1970.

[26] G. D. Forney Jr. Structural Analysis of Convolutional Codes via Dual Codes. *IEEE Transactions on Information Theory*, 19(4):512–518, July 1973.

[27] R. G. Gallager. Low-Density Parity-Check Codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962.

[28] R. Johannesson and K. S. Zigangirov. *Fundamentals of Convolutional Coding*. Wiley-IEEE Press, Piscataway, New Jersey, USA, 1999.

[29] J. Leon. A Probabilistic Algorithm for Computing Minimum Weights of Large Error-Correcting Codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, September 1988.

[30] P. Lu, S. Li, X. Luo, and Y. Zou. Blind Recognition of Punctured Convolutional Codes. In *Proceedings of the International Symposium on Information Theory (ISIT 2004)*, page 457, June 2004.

[31] P. Lu, S. Li, Y. Zou, and X. Luo. Blind Recognition of Punctured Convolutional Codes. *Science in China Series F: Information Sciences*, 48(4):484–498, 2005.

[32] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, New York, USA, 2002.

[33] M. Marazin. *Reconnaissance en aveugle de codeur à base de code convolutif : Contribution à la mise en oeuvre d'un récepteur intelligent.* PhD thesis, Université de Bretagne Occidentale, Brest, France, December 2009.

[34] M. Marazin, R. Gautier, and G. Burel. Dual code method for blind identification of convolutional encoder for cognitive radio receiver design. In *2009 IEEE GLOBECOM Workshops*, pages 1–6, November 2009.

[35] M. Marazin, R. Gautier, and G. Burel. Algebraic method for blind recovery of punctured convolutional encoders from an erroneous bitstream. *IET Signal Processing*, 6(2):122–131, April 2012.

[36] M. Marazin, R. Gautier, and G. Burel. Some interesting dual-code properties of convolutional encoder for standards self-recognition. *IET Communications*, 6(8):931–935, May 2012.

[37] J. L. Massey. Shift-Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, January 1969.

[38] R. J. McEliece. The algebraic theory of convolutional codes. *Handbook of Coding Theory*, 1:1065–1138, 1998.

[39] J. G. Proakis. *Digital Communications 5th Edition.* McGraw Hill, New York, New York, USA, 2008.

[40] H. Ryu, J. Lee, H. Hong, and D. Yoon. Estimation of interleaver period for unknown signals. In *2nd IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC 2010)*, pages 678–680, September 2010.

[41] G. Sicot and S. Houcke. Blind detection of interleaver parameters. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, volume 3, pages 829–832, March 2005.

[42] G. Sicot, S. Houcke, and J. Barbier. Blind detection of interleaver parameters. *Signal Processing*, 89(4):450 – 462, 2009.

[43] J. Stern. A Method for Finding Codewords of Small Weight. In *Proceedings of the 3rd International Colloquium on Coding Theory and Applications*, pages 106–113, London, UK, 1989. Springer-Verlag.

[44] R. M. Tanner. A Recursive Approach to Low Complexity Codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981.

[45] J.-P. Tillich, A. Tixier, and N. Sendrier. Recovering the interleaver of an unknown Turbo-Code. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2014)*, pages 2784–2788, June 2014.

[46] A. Valembois. Detection and recognition of a binary linear code. *Discrete Applied Mathematics*, 111(1–2):199–218, 2001.

[47] F. Wang, Z. Huang, and Y. Zhou. A Method for Blind Recognition of Convolution Code Based on Euclidean Algorithm. In *International Conference on Wireless Communications, Networking and Mobile Computing (WiCom 2007)*, pages 1414–1417, September 2007.

[48] S. B. Wicker. *Error Control Systems for Digital Communication and Storage*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA, 1995.

[49] J. Ziegler. Automatic Recognition and Classification of Forward Error Correcting Codes. Master's thesis, George Mason University, Fairfax, Virginia, USA, 2000.