COMPOUND CODES BASED ON IRREGULAR GRAPHS AND THEIR ITERATIVE DECODING

by

Telex Magloire Ngatched Nkouatchah
University of KwaZulu-Natal

2004

Submitted in fulfilment of the academic requirements for the degree of Doctor of Philosophy in the School of Electrical, Electronic, and Computer Engineering, University of KwaZulu-Natal, 2004.

Abstract

Low-density parity-check (LDPC) codes form a Shannon limit approaching class of linear block codes. With iterative decoding based on their Tanner graphs, they can achieve outstanding performance. Since their rediscovery in late 1990's, the design, construction, and decoding of LDPC codes as well as their generalization have become one of the focal research points. This thesis takes a few more steps in these directions.

The first significant contribution of this thesis is the introduction of a new class of codes called Generalized Irregular Low-Density (GILD) parity-check codes, which are adapted from the previously known class of Generalized Low-Density (GLD) codes. GILD codes are generalization of irregular LDPC codes, and are shown to outperform GLD codes. In addition, GILD codes have a significant advantage over GLD codes in terms of encoding and decoding complexity. They are also able to match and even beat LDPC codes for small block lengths.

The second significant contribution of this thesis is the proposition of several decoding algorithms. Two new decoding algorithms for LDPC codes are introduced. In principle and complexity these algorithms can be grouped with bit flipping algorithms. Two soft-input soft-output (SISO) decoding algorithms for linear block codes are also proposed. The first algorithm is based on Maximum a Posteriori Probability (MAP) decoding of low-weight subtrellis centered around a generated candidate codeword. The second algorithm modifies and utilizes the improved Kaneko's decoding algorithm for soft-input hard-output decoding. These hard outputs are converted to soft-decisions using reliability calculations. Simulation results indicate that the proposed algorithms provide a significant improvement in error performance over Chase-based algorithm and achieve practically optimal performance with a significant reduction in decoding complexity.

An analytical expression for the union bound on the bit error probability of linear codes on the Gilbert-Elliott (GE) channel model is also derived. This analytical result is shown to be accurate in establishing the decoder performance in the range where obtaining sufficient data from simulation is impractical.

TO GOD BE THE GLORY

Dedication

This thesis is dedicated to my father, Elias Nkouatchah, who passed away on November 5, 2002. I thank you for my life and all the love and supported sacrifices you have made for me. The sense of discipline, independence, hard work, integrity, humility, love and compassion for all people you awakened and instilled within me at an early age has left an indelible impression on my life. As a young elementary boy, I remember promising that I would take my studies all the way and receive a Ph.D. Though I am glad that, despite the many adversities, I have kept the promise, it is a bitter disappointment that you did not live long enough to see its completion. No collection of words can express the deep sense of loss and anger I feel at having lost you prematurely. I am, however, relieve to know that you are now at peace and will never suffer again.

Preface

The research work discussed in this thesis was performed by Mr. Telex Magloire Ngatched Nkouatchah, under the supervision of Professor Fambirai Takawira, at the University of KwaZuluNatal's School of Electrical, Electronic and Computer Engineering Centre for Radio Access Technologies, which is sponsored by Alcatel and Telkom South Africa Ltd through the Centre of Excellence Programme.

Parts of this thesis have been presented by the author at the SATNAC 2002 conference in Drakensberg, South Africa, at the IEEE AFRICON 2002 conference in George, South Africa, at the IEEE WCNC 2003 conference in New Orleans, Louisiana, U.S.A, at the SATNAC 2003 conference in George, South Africa, at the SATNAC 2004 conference in Spier Wine Estate, Western Cape, South Africa, at the IEEE AFRICON 2004 in Gaborone, Botswana, and at the IEEE Globecom 2004 conference in Dallas, Texas, U. S. A. A paper has also been accepted for presentation at the IEEE ICC 2005 in Seoul, Korea. A paper has been published in SAIEEE, and three other papers have been submitted for review to the journals, "IEEE Transactions on Communications," and "IEEE Transactions on Vehicular Technology."

The whole thesis, unless specifically indicated to the contrary in the text, is the author's work, and has not been submitted in part, or in whole to any other University.

As the candidate's supervisor, I have approved this thesis for submission.

Name: ....

Date: ....

Signature: ....

Acknowledgements

It is impossible to express in words of any length my gratefulness and appreciation to my supervisor Professor Fambirai Takawira who introduced me to the area of information and coding theory. Working with him is a true privilege, and I have benefited tremendously from his knowledge of science, both in depth and broadness, and his enthusiasm of doing research. While being a constant source of valuable insights and advice, he has always allowed me high degree of freedom and independence to pursue my own research and study. It was, however, his unlimited encouragement, patience, and availability that kept me focused and made all the accomplishments in this thesis possible. He is a role model that I can always look up to.

I would like to express my deepest gratitude to Professor Martin Bossert for his invitation at the department of Telecommunications and Applied Information Theory of the University of Ulm in Germany, and his readiness to take over the supervision of my work. I deeply appreciate his insightful ideas and his many hours spent with me discussing my research.

I would like to thank Dr. Hongju Xu, who has been given me invaluable suggestions and comments on my research.

My special thanks go to Alcatel and Telkom South Africa for their valued financial support, to the University of KwaZulu-Natal for the "Doctoral research scholarship", and to the German Academic Exchange Service (DAAD) for supporting my stay at the University of Ulm.

I am grateful to all my postgraduate friends for their assistance and for making our time together enjoyable. Special thanks to Tom Walingo and Dr. Emmanuel Bejide. I thank our secretaries for their professional help and for their care and kindness. Bruce Harrison, our system administrator, who devoted great amount of time to maintain the computers, has been an indispensable resource for my work.

Above all, I would like to express my greatest gratitude and sincere appreciation to my family and friends for their support, understanding, love and trust.

Table of Content

List of Tables

List of Figures

Acronyms and Notations

The main acronyms and notations used in this document are presented here. We try to keep consistent and homogeneous notations in the entire report. However, in some rare places, notations do not have their general meaning summarized here. Normally, acronyms are recalled the first time they appear in any chapter.

Acronyms

APP	A Posteriori Probability
AWGN	Additive White Gaussian Noise (channel)
bps	bit per second
BCH	Bose-Chaudhuri-Hocquenghem (code)
BCJR	Bahl Cocke Jelinek Raviv
BER	Bit Error Rate
BF	Bit Flipping
BIAWGN	Binary Input Additive White Gaussian Noise
BIBD	Balanced Incomplete Block Design
BPSK	Binary Phase Shift Keying (modulation)
BSC	Binary Symmetric Channel
DMC	Discrete Memoryless Channel
EG	Euclidean Geometry
FEC	Forward Error Correction
GE	Gilbert-Elliott (channel)
GF	Galois Field
GILD	Generalized Irregular Low Density (code)
GLD	Generalized Low density (code)
GV	Gilbert-Varshamov (bound)
HAD	Hard Decision Aided
IDBP	Iterative Decoding Based on Belief Propagation
IHAD	Improved Hard Decision Aided
iid	independent and identically distributed

IKA	Improved Kaneko Algorithm
IOWEF	Input-Output Weight Enumerator Function
IWBF	Improved Weighted Bit Flipping (decoding)
KA	Kaneko Algorithm
LDPC	Low Density Parity Check (code)
LLR	Log Likelihood Ratio
MAP	Maximum *a posteriori* (decoding)
MDS	Maximum Distance Separable
ML	Maximum Likelihood (decoding)
MLG	Majority Logic (decoding)
pce	parity check equation
pdf	probability density function
PEG	Progressive Edge Growth
RS	Reed Solomon
SDR	Sign Difference Ratio
SNR	Signal to Noise Power Ratio
SPA	Sum Product Algorithm
spc	single parity check (equation)
SISO	Soft-Input Soft-Output (decoder)
SOVA	Soft-Output Viterbi Algorithm
SNR	Signal-to-Noise Ratio
UMP	Uniformly Most Powerful
VA	Viterbi Algorithm
WBF	Weighted Bit Flipping (decoding)
WMLG	Weighted Majority Logic (decoding)

Notations

$\underline{c}_i$	If not specified, the i-th bit of the considered codeword
$\underline{c}_j$	If not specified, the j-th codeword of the considered code
c_l	If not specified, a codeword of weight l

C	If not specified, the considered compound code
$d_{H\min}$	The minimum Hamming distance of the considered code
E_b	Carrier Frequency Energy per information bit
E_b/N_0	Signal to Noise Ratio per information bit
k	If not specified, the dimension of the considered constituent code
K	If not specified, the dimension of the considered compound code
l	If not specified, the weight of the considered codeword
n	If not specified the length of the considered constituent code
N	If not specified, the length of the considered compound code
$N(l)$	Weight distribution of the considered code
$\bar{N}(l)$	Average weight distribution of the considered ensemble of codes
$N_0/2$	Two sided Power Spectral Density of the AWGN
P_{eb}	Bit Error Probability (or BER)
P_{ew}	Word Error Probability
r	If not specified, the rate of the considered constituent code
R	If not specified, the rate of the considered compound code

CHAPTER 1

INTRODUCTION

With the advance of digital logic design, the last decade has observed wide application and deployment of digital communication and error protection techniques. These techniques have enabled, and induced explosive demands for, high-quality and high-speed voice-band modems, digital subscriber loops, personal wireless communications, mobile and direct-broadcast satellite communications. To achieve efficient use of bandwidth and power and, at the same time, combat against adverse channel conditions, new engineering challenges have arisen. For example the systems should have low physical and computational complexity to increase portability and reachability, allow seamless data rate changes to cope with time-varying channel conditions and higher level network protocols, and provide unequal error protection to accommodate different service rates and to differentiate bits of nonuniform importance from advanced source encoders. In this thesis, new high-performance error correcting techniques with low system complexity are developed to address these new challenges.

1.1 Coding for Digital Data Transmission

The ever increasing information transmission in the modern world is based on reliably communicating messages through noisy transmission channels; these can be telephone lines, deep space, magnetic storing media, etc. Error-correcting codes play a significant role in correcting errors incurred during transmission; this is carried out by encoding the message prior to transmission and decoding the corrupted received codeword for retrieving the original message.

The performance of a coded communication system is usually measured by its probability of decoding error called *error probability*. There are two types of error probability. Probability of word (frame or block) error is the probability that a decoded codeword at the output of the decoder is in error. This error probability is commonly called *word error rate* (WER), frame error rate (FER), or *block error rate* (BLER). Probability of bit error is the probability that a decoded information bit at the output of the decoder is in error. This error probability is commonly called *bit error rate* (BER).

Another performance measure of a coded communication system is *coding gain.* Coding gain is defined as the reduction in the required *signal-to-noise ratio* (SNR) to achieve a specific error probability for a coded communication system compared to an uncoded system that transmit information at the same rate. SNR is defined as the ratio of the average power of the demodulated message signal to the average power of the noise measured at the receiver output.

1.2 Shannon Limit

The fundamental approach to the problems of efficiency and reliability in communication systems is contained in the Noisy Channel Coding Theorem developed by C. E. Shannon [1] in 1948. Shannon's theorem states that over a noisy channel, if the transmission rate R, constituted by the ratio between the number of bits in the original message and the transmitted codeword, is less than the channel capacity C, there exists a coding scheme of code rate R that achieves reliable communication. Specifically, for every rate $R < C$, there exists at least one good code sequence with probability of error approaching zero while the blocklength, N, approaches infinity. Furthermore, he proved that the probability of error is bounded away from zero if the transmission rate is greater than capacity, no matter how large N is. Shannon's channel coding theorem clearly states that channel capacity is a dividing point: at rates below capacity, the probability of error goes to zero exponentially; at rates above capacity, the probability of error goes to one exponentially. The proof to the theorem is essentially non-constructive. It shows that for long block length, almost all codes of rate R ($< C$) would be reliable. However, it does not give an explicit construction of capacity-approaching codes, nor does it lay out practical decoding algorithms. Ever since, coding theorists have been trying to find codes that would achieve Shannon's theoretical limit. Besides good error performance, the codes for pratical applications must have realizable and preferably small encoding and decoding complexity.

1.3 Two Classes of Shannon Limit Approaching Codes

In the 50 years since Shannon determined the capacity of ergodic noisy channels, the construction of capacity-approaching coding schemes that are easy to encode and decode has been the supreme goal of coding research. In the last decade, a breakthrough was made in this field with the discovery of some practical codes and decoding algorithms which approach considerably the ultimate channel capacity limit. There are two large classes of such codes.

In 1993, Berrou, Glavieux, and Thitimajshima [2] [3] introduced turbo codes to the world. They showed that turbo codes provide capacity approaching performance with suboptimal iterative decoding. These results caused an increased interest in iterative decoding methods and iteratively decidable codes. Further research led to the rediscovery of another powerful class of iteratively decodable codes, *low-density parity-check* (LDPC) codes, by Sipser et *al.* [9], MacKay et *al.* [10], and Wiberg [11].

Turbo codes are obtained by parallel or serial concatenation of two or more component codes with intrleavers between the encoders. The component codes are mainly simple convolutional codes. Therefore, it is easy to construct and encode turbo codes. As mentioned, an interleaver is required to permute the input information sequence. It is shown that the larger the interleaver size, the better the performance of turbo codes. On the other hand, large interleaver causes large decoding delay. In decoding of a turbo code, each component code is decoded with a trellis based algorithm. Therefore, for practical implementations, only codes with simple trellises can be used as component codes of turbo codes. However, codes with simple trellises normally have small minimum distances, causing the error floor at medium to high SNR. In turbo decoding, at each decoding iteration the reliability values and estimates are obtained only for the information bits. Thus no error detection can be performed to stop the decoding iteration process. The only way to stop the decoding is to test the decoding convergence, which is usually complex. No error detection results in poor block error rate and slow termination of iterative decoding.

LDPC codes are block codes. They were discovered by Gallager in early 1960's [2] [3]. After their discovery, they were ignored for a long time and rediscovered recently. It has been proved that LDPC codes are *good*, in the sense that sequences of codes exist which, when optimally decoded, achieve arbitrarily small error probability at nonzero communication rates up to some maximum rate that may be less than the capacity of the given channel. Numerous simulation results showed that long LDPC codes with iterative decoding achieve outstanding performance. Until recently, good LDPC codes were mostly computer generated. Encoding of these computer generated codes is usually very complex due to the lack of understanding their structure. On the other hand, iterative decoding for LDPC codes is not trellis based, and it is not required for LDPC codes to have simple trellises. Thus, their minimum distances are usually better that those of turbo codes. For this reason, LDPC codes usually outperform turbo codes in moderate to high SNR region and exhibit error floor at lower error rates. Another advantage of LDPC codes over turbo codes is that their decoding algorithm provides reliability values and estimates for every code bit at the end of each iteration, enabling error detection. The decoding iteration process is stopped as soon as the estimated sequence is detected as a codeword. Therefore, LDPC codes normally provide better block error performance and faster termination of the iterative decoding.

1.4 Thesis Objective and Outline

This thesis deals firstly with the design of concatenated codes and reduced complexity iterative decoding algorithms, and secondly with the characterization of codes in channels with memory.

The thesis is organized in such a way that different chapters can be read independently. The outline for the remainder of the thesis is as follows. In Chapter 2, the construction of the original regular LDPC codes is briefly recalled. Their analytical properties as derived by Gallager are reviewed. Their graphical representation as Tanner codes on random graphs are also presented, and a review of the different construction methods is given. The chapter does not claim to be self-contained but its aim is rather to be an introduction for the *generalized irregular low-density* (GILD) codes presented in Chapter 4, whose construction is inspired by irregular LDPC codes and *generalized low-density* (GLD) codes, and that share the common properties with them.

In Chapter 3, different existing decoding methods used for decoding of LDPC codes are described and two new decoding algorithms are presented. The first algorithm is a hard-decision method, and the second one is a modification of the first to include reliability information of the received symbols. In principle and in complexity, the algorithms belong to the class of so called bit flipping algorithms. The defining attribute of the proposed algorithms is the bit selection criterion which is based on the fact that, for low density matrices, the syndrome weight increases with the number of errors in average until error weights much larger than half the minimum distance. A loop detection procedure with minimal computational overhead is also proposed that protects the decoding from falling into infinite loop traps. Simulation results show that the proposed algorithms offer an appealing performance/cost trade-offs and may deserve a place in an LDPC decoding "toolbox".

In Chapter 4, a new class of codes called *generalized irregular low-density* (GILD) codes is presented. This family of pseudo-random error correcting codes is built as the intersection of randomly permuted binary codes. It is a direct generalization of irregular LDPC codes, and is adapted from the previously known class of *generalized low density* (GLD) codes introduced independently by Lentmaier et *al.*[22], and Boutros et *al.* [23]. It is proved by an ensemble performance argument that these codes exist and are asymptotically good in the sense of the minimum distance criterion, *i.e.* the minimum distance grows linearly with the block length. Upper and lower bounds on their minimum Hamming distance are provided, together with their maximum likelihood decoding error probability. Two iterative soft-input soft-output decoding for any GILD code are presented, and iterative decoding of GILD codes for communication over an AWGN channel with binary antipodal modulation (BPSK) is studied. The results are compared in terms of performance and complexity with those of GLD codes. The high flexibility in selecting the parameters of GILD codes and their better performance and higher rate make them more attractive than GLD codes and hence suitable for small and large block length forward error correcting schemes. Comparison between simulation results of a GILD code and the best LDPC code of length 1008 and rate 0.5 shows very close performances, suggesting that variations of GILD codes may be able to match or beat LDPC codes for small block lengths.

In Chapter 5, reduced-complexity trellis-based soft-input soft-output (SISO) decoding of linear block codes is considered. A new low-weight subtrellis based SISO decoding algorithm for linear block code to achieve near optimal error performance with a significant reduction in decoding complexity is presented. The proposed scheme is suitable for iterative decoding and has the following important features. An initial candidate codeword is first generated by a simple decoding method. A low-weight subtrellis diagram centered around the candidate codeword is constructed. The MAP algorithm is then applied to the subtrellis. The generated extrinsic information is used as apriori information to improve the generation of a candidate codeword for the next stage of iteration. Simulation results indicate that the proposed algorithm provides a significant improvement in error performance over Chase-based algorithm and achieves practically optimal performance with a significant reduction in decoding complexity.

In Chapter 6, an efficient list-based soft-input soft-output (SISO) decoding algorithm for compound codes based on linear block codes is presented. Attention is focused on GLD codes. The proposed algorithm modifies and utilizes the improved Kaneko's decoding algorithm for soft-input hard-output decoding. These hard outputs are converted to soft-decisions using reliability calculations. Compared to the trellis-based Maximum a Posteriori Probability (MAP) algorithm, the proposed algorithm suffers no degradation in performance at low bit-error rate (BER), but presents the major advantages of being applicable in cases where the trellis-based MAP algorithm would be prohibitively complex and impractical. Compared to the Chase-based algorithm of [85], [86], [88], [89] the proposed algorithm is more efficient, has lesser computational complexity for the same performance and provides an effective tradeoff between performance and computational complexity to facilitate its usage in practical applications. To improve the average decoding speed of the GLD decoder, two simple criteria for stopping the iterative process for each frame immediately after the bits can be reliably decoded with no further iterations are proposed.

In Chapter 7, an analytical expression for the pairwise error probability of maximum likelihood decoding of a binary linear code on the Gilbert-Elliott (GE) channel model is derived. This expression is used to obtain the union bound on the bit error probability of linear codes on the GE channel. Comparisons between the results obtained by this analytical expression and results obtained through computer simulations in the case of turbo codes and generalized irregular low density (GILD) codes show that the analytical results are accurate in establishing the decoder performance in the range where obtaining sufficient data from simulation is impractical.

Finally, we summarize our work and comment on some future extensions in Chapter 8.

1.5 Original Contribution in the Thesis

The original contributions in the thesis include:

1. The introduction of two new decoding algorithms for LDPC codes. The first algorithm is a hard-decision decoding method, and the second one, which is a modification of the first to include reliability information of the received symbols, is between hard- and soft-decision decoding methods. In both algorithms, one bit is flipped in each iteration and the bit to be flipped is chosen in such a way that the syndrome weight decreases. Thus, the algorithms belong to the class of so called bit flipping algorithms. Simulations results on the additive white Gaussian noise channel comparing the proposed algorithms with other well known decoding algorithms show that the former achieve excellent performances in terms of the bit-error rate, while requiring lower complexity.
2. The introduction of a new class of pseudo-random error correcting codes called *generalized irregular low-density* (GILD) codes built as the intersection of randomly permuted binary codes. It is a direct generalization of irregular LDPC codes, and is adapted from the previously known class of *generalized low-*

density (GLD) codes. The high flexibility in selecting the parameters of GILD codes and their better performance and higher rate make them more attractive than GLD codes.

3. The presentation of a new low-weight subtrellis based soft-input soft-output decoding algorithm for linear block code suitable for iterative decoding. The algorithm is applied to GILD codes, and simulation results indicate that the proposed algorithm provides a significant improvement in error performance over Chase-based algorithm and achieves practically optimal performance with a significant reduction in decoding complexity.
4. The presentation of an efficient list-based soft-input soft-output decoding algorithm for compound codes based on linear block codes. The proposed algorithm modifies and utilizes the improved Kaneko's decoding algorithm for soft-input hard-output decoding. These hard outputs are converted to soft-decisions using reliability calculations. An important feature of the proposed algorithm is the derivation of a condition to rule out useless test error patterns in the generation of candidate codewords. This rule-out condition reduces many unnecessary decoding iterations and computations.
5. The derivation of an analytical expression for the pairwise error probability of maximum likelihood decoding of a binary linear code on the Gilbert-Elliott (GE) channel model. This expression is used to obtain the union bound on the bit error probability of linear codes on the GE channel. The analysis is applied to turbo codes and GILD codes, and it is shown that the analytical results are accurate in establishing the decoder performance in the range where obtaining sufficient data from simulation is impractical.

1.6 Published Work

Parts of the material in this thesis have been published, or submitted for possible publications in transactions [161]-[165]:

- T. M. N. Ngatched and F. Takawira, "Improved generalized low-density parity-check codes using irregular graphs," *SAIEEE Transactions*, vol. 94, pp. 43-49.

- T. M. N. Ngatched and F. Takawira, "Generalization of irregular low-density parity-check codes," submitted to *IEEE Trans. Commun.*, Dec. 2003.
- T. M. N. Ngatched and F. Takawira, "Union bound for binary linear codes on the Gilbert-Elliott channel with application to turbo-like codes," submitted to *IEEE Trans. Veh. Technol.*, June. 2004.
- T. M. N. Ngatched, M. Bossert, and A. Fahrner, "Two decoding algorithms for low-density parity-check codes,"submitted to *IEEE Trans. Commun.*, Jan. 2005.
- T. M. N. Ngatched and F. Takawira, "A low-weight trellis based soft-input soft-output decoding algorithm for binary linear block codes," in preparation. To be submitted to *IEEE Trans. Commun.*

Parts have also been presented and submitted for presentation at conferences [153]-[160]:

- T. M. N. Ngatched, M. Bossert, and A. Fahrner, "Two decoding algorithms for low-density parity-check codes," accepted for presentation at *IEEE International Conference on Communications (ICC)* 2005, 16 – 20 May 2005, Seoul, Korea.
- T. M. N. Ngatched and F. Takawira, "A low-weight trellis based soft-input soft-output decoding algorithm for binary linear block codes with application to generalized irregular low density codes," in Proc. *GLOBECOM* 2004, *IEEE Global Telecommunications Conference*, vol. 4, no. 1, pp. 705-710, Nov. 29 – Dec. 3, 2004.
- T. M. N. Ngatched and F. Takawira, "Pairwise error probability on the Gilbert-Elliott channel with application to turbo codes," in *Proc. IEEE AFRICON 2004*, pp. 279-284, Sept. 2004.
- T. M. N. Ngatched and F. Takawira, "An Iterative Soft-Input Soft-Output Decoding Algorithm for Linear Block Codes," in *Proc. South African Telecommunication Networks & Applications Conference (SATNAC)*, Spier Wine Estate, Western Cape, South Africa, 6-8 Sept. 2004.
- T. M. N. Ngatched and F. Takawira, "Error-Correcting Codes Based on Irregular Graphs," in *Proc. South African Telecommunication Networks & Applications Conference (SATNAC)*, George, South Africa, 8-10 Sept. 2003.

- T. M. N. Ngatched and F. Takawira, "Efficient Decoding of Generalized Low-Density Parity-Check Codes Based on Long Component codes," in Proc. *WCNC 2003, IEEE Wireless Communications and Networking Conference*, vol. 4, no. 1, pp. 705-710, Mar. 2003.
- T. M. N. Ngatched and F. Takawira, "Generalized Irregular low-Density (Tanner) Codes based on Hamming Component Codes," in Proc. *IEEE AFRICON 2002*, pp. 193-196, Oct. 2002.
- T. M. N. Ngatched and F. Takawira, "Decoding of GLD Codes in a Rayleigh Fading Channel With Imperfect Channel Estimates," in *Proc. South African Telecommunication Networks & Applications Conference (SATNAC)*, Drakensberg, South Africa, 1-4 Sept. 2002.

CHAPTER 2

LOW-DENSITY PARITY-CHECK CODES

Introduced by Gallager in 1962 [2][3], low-density parity-check (LDPC) codes are a class of linear error-correcting block codes. As their name suggests, LDPC codes are defined in terms of a sparse parity-check matrix. LDPC codes exploit the following fruitful ideas:

- The use of random permutations linking simple parity-check codes to build an efficient low complexity code that imitates random coding.
- An iterative decoding technique where *a priori* information and channel observations are both used to compute *a posteriori* and new a priori information.

Unfortunately, except for the papers by Zyablov and Pinsker [6], Margulis [7] and Tanner [8], Gallager's work has been forgotten by the majority of the scientific community during the past three decades, until the recent invention of turbo codes [4] [5] which share the same above ingredients with LDPC codes.

LDPC codes were then rediscovered by Sipser et *al.* [9], MacKay et *al.* [10], and Wiberg [11].[1] The past few years have brought many new developments in this area. Among the recent works, MacKay [14] showed that Gallager's decoding algorithm is related to Pearl's belief propagation algorithm [15], Luby et *al.* [16] [17] [18], Richardson et *al.* [19], MacKay et *al.* [20], and Chung et *al.* [21] extended Gallager's definition of LDPC codes to include *irregular* codes. The results have been spectacular, with performance surpassing the best turbo codes for large code lengths. As a direct generalization of Gallager's LDPC codes, *generalized low-density (GLD) parity-check* codes were independently introduced by Lentmaier [55] and Boutros [56].

[1] Similar concepts have also appeared in the physics literature [12] [13].

In this chapter, we briefly recall the construction of the original regular LDPC codes, their analytical properties as derived by Gallager and their iterative decoding. We present their graphical representation as Tanner codes [8] on random graphs. This chapter does not claim to be self-contained but its aim is rather to be an introduction for the *generalized irregular low-density* (GILD) codes presented in Chapter 3, whose construction is inspired by irregular LDPC codes, and that share the common properties with them. A review of the different construction methods is also given.

2.1 LDPC Structure

2.1.1 Definition of Low-Density Parity-Check Codes

A regular LDPC code C with parameters (N, j, k) is a linear block code of length N whose parity-check matrix $\boldsymbol{H}$ has j ones in each column, k ones in each row, and thus zeros elsewhere. The value of k must divide the block length N of the code. The numbers j and k have to remain small with respect to N in order to obtain a sparse matrix $\boldsymbol{H}$. Such a matrix is represented in Figure 2.1.

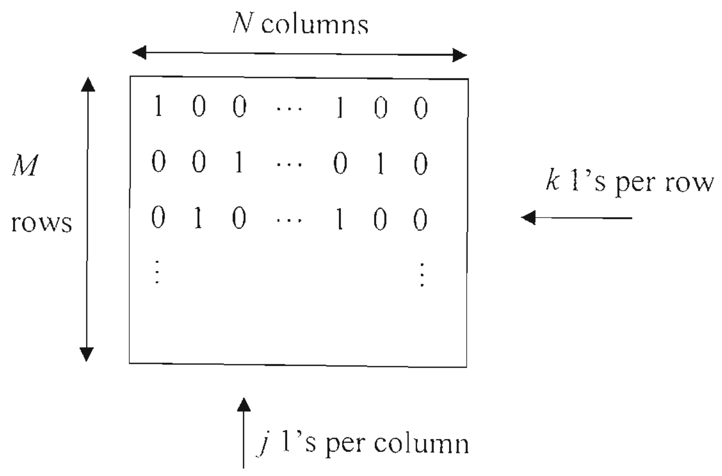

Figure 2.1: Properties of the parity-check matrix $\boldsymbol{H}$ of regular LDPC codes.

This matrix has hence $M = N - K = Nj/k$ rows *i.e.*, the number of single *parity-check equations* (pce) of the code. Each coded bit belongs to j pces, and any pce involves k coded bits. If $\boldsymbol{H}$ has full rank the corresponding code has $K = N - M = N - Nj/k$ information symbols and thus the code rate is

$$R = 1 - j/k\,. \tag{2.1}$$

As a matter of fact there have to be at least $(j-1)$ linear independent rows in $\boldsymbol{H}$ and consequently the code has less information symbols, leading to a slightly higher rate. However, with an increasing block length N a small number of linear dependent rows has only a minor effect on the rate R and equation (2.1) gives a good approximation of the actual rate of the code. It should be noted that there exist a lot of different LDPC codes with the same parameter (N, j, k). All possible codes with the same values of N, j, and k form the ensemble of (N, j, k) LDPC codes. The definition of LDPC codes does not imply that all codes within the same ensemble have the same properties.

Gallager's construction of a regular low-density parity-check matrix $\boldsymbol{H}$ with parameters (N, j, k) consists of dividing it into j sub-matrices $\boldsymbol{H}^1, \cdots, \boldsymbol{H}^j$, each containing a single one in each of its columns. The first of these, $\boldsymbol{H}^1$, looks like a "flattened" identity matrix (that is, an identity matrix where each one is replaced by k ones in a row, and where the number of columns is multiplied accordingly). The $j-1$ other sub-matrices $\boldsymbol{H}^2, \cdots, \boldsymbol{H}^j$ are derived from $\boldsymbol{H}^1$ by $j-1$ column-wise random permutations $\pi_2, \cdots, \pi_j$ of $\boldsymbol{H}^1$. Figure 2.2 shows the parity-check matrix of a particular (20, 3, 4) LDPC code. It can be noted that summing in each sub-matrix all the rows lead to an all-one row. Hence, there are at least $j-1$ dependent rows and hence the rate is greater then $1 - j/k$.

A code C can be seen as the intersection of j *super-codes* $C^1, \cdots, C^j$ whose respective parity-check matrices are the sub-matrices $\boldsymbol{H}^1, \cdots, \boldsymbol{H}^j$. Since each sub-matrix consists of N/k *independent* single-parity-check $(k, k-1)$ codes (spcc) C_0. We have:

$$C = \bigcap_{j=1}^{J} C^j\,, \tag{2.2}$$

and:

$$C'^{1} = \bigoplus_{l=1}^{N/k} C_0 . \qquad (2.3)$$

Table 2.1: Gallager's construction of the parity-check matrix of a (20, 3, 4) regular LDPC code C.

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

It is important to note that the matrix $\boldsymbol{H}$ is not in systematic form. Therefore a systematisation has to be done, for two reasons. The first one is to compute the actual dimension, and rate, of the code. The second reason is the ease of encoding. Let us denote by $\boldsymbol{H}' = [\boldsymbol{P} \mid \boldsymbol{I}]$ the result of the systematisation of $\boldsymbol{H}$, and Π_S the column permutation applied. $\boldsymbol{H}'$ is no longer a low-density parity-check matrix. $\boldsymbol{G} = \left[\boldsymbol{I} \mid \boldsymbol{P}^T\right]$ is the systematic generator matrix associated with $\boldsymbol{H}'$. If $\boldsymbol{b}$ is the information bits vector, the corresponding codeword is $\boldsymbol{c} = \boldsymbol{bG}$. We have $\boldsymbol{cH}'^T = \boldsymbol{0}$, but also $\Pi_S^{-1}(\boldsymbol{c})\boldsymbol{H}^T = \boldsymbol{0}$. Since Gallager's probabilistic decoding (see Section 2.1.3) takes benefit of the sparsity of matrix $\boldsymbol{H}$, an LDPC coding scheme uses the systematic generator matrix $\boldsymbol{G}$ at the encoder and the LDPC matrix $\boldsymbol{H}$ at the decoder. The received symbols must be interleaved by Π_S^{-1}. For a large length, the encoding complexity is high, which is a practical drawback of LDPC codes. Several authors have addressed the encoding problem of LDPC codes. Sipser et *al.* [9] and Luby et *al.* [16] suggested the use of cascaded rather than bipartite graphs. By choosing the number of stages and the relative size of each stage carefully one can construct codes which are encodable and decodable in linear time. One draw back of this approach lies in the fact that each stage (which acts like a subcode) has a length which is in general considerably smaller than the length of the overall code. This results, in general, in a performance loss compared to a standard LDPC code with the same overall length. MacKay et *al.* suggested forcing the parity-check matrix to have (almost) lower irregular form, i.e., the ensemble of codes is restricted not only by the degree constraints but also by the constraint that the parity-check matrix has lower triangular shape. This restriction guarantees a linear time encoding complexity but, in general, it also results in some loss of performance. Richardson and Urbanke [57] have shown that, even without cascade or restrictions on the shape of the parity-check matrix, the encoding complexity is quite manageable in most cases and provably linear in many cases.

2.1.2 Analytical Properties of LDPC Codes

Gallager presented several analytical results on LDPC codes. We only summarize here those related to the minimum Hamming distance of LDPC codes, since the methods used to find them are also used in our original work.

2.1.2.1 LDPC Codes with j = 3 are Asymptotically Good

Gallager compared the asymptotic *average*[2] minimum Hamming distance properties of LDPC codes to the ones of the whole ensemble of binary linear block codes of same

[2] "average" means that we consider the ensemble of LDPC codes with same parameters (N, j, k), constructed with all the possible choices for the interleavers $\pi_2, \cdots, \pi_j$, and we average the results on this ensemble.

rate *R*. By the random coding argument, it is well known that the latter reaches the Gilbert-Varshamov (GV) bound: asymptotically, when the length *N* of the parity-check codes tends to infinity, the average normalized hamming distance of their ensemble $\delta_0 = \bar{d}_{\min}/N$ satisfies:

$$H(\delta_0) = \left(1 - R\right)\log(2), \tag{2.4}$$

where *H* denotes the natural entropy function.

Computing the average number of codewords $\bar{N}_{jk}(l)$ of weight *l* of the LDPC code ensemble of parameters $\left(N, j, k\right)$, Gallager proved that when *N* is large enough:

$$\bar{N}_{jk}(l) \le C(\lambda, N)\exp\left(-NB_{jk}(\lambda)\right), \tag{2.5}$$

where B_{jk} and $C(\lambda, N)$ are defined as follows:

$$B_{jk}(\lambda) = (j-1)H(\lambda) - \frac{j}{k}\left[\mu(s) + (k-1)\ln 2\right] + js\lambda, \tag{2.6}$$

$$C(\lambda, N) = \left[2\pi N\lambda(1-\lambda)\right]^{\frac{j-1}{2}} \exp\left(\frac{j-1}{12N\lambda(1-\lambda)}\right). \tag{2.7}$$

Where $\lambda = l/N$, $\mu(s)$ is a function depending only on *k*, and *s* is a parameter that has to be optimized. We omit the details since they can be found in [2] and since we use the same approach in Chapter 4 to prove that Generalized Irregular Low density codes are asymptotically good.

Asymptotically, the sign of the exponent function $B_{jk}(\lambda)$ determines the behavior of $\bar{N}_{jk}(l)$: if $B_{jk}(\lambda) > 0$, then $\bar{N}_{jk}(l)$ tends to zero when *N* tends to infinity. The highest value of λ such that $B_{jk}(\lambda) > 0$ gives us an asymptotic lower bound on the average normalized Hamming distance δ_{jk}. We computed the values of δ_{jk} for different choices of (j,k) and compared them with the GV bound. The results are presented in Figure 2.3. We observe that $\delta_{jk} > 0$ for all the computed values with $j \ge 3$. Hence, the LDPC codes are *asymptotically good*[3] if $j \geq 3$. Furthermore, LDPC codes are close to the GV bound when j increases.

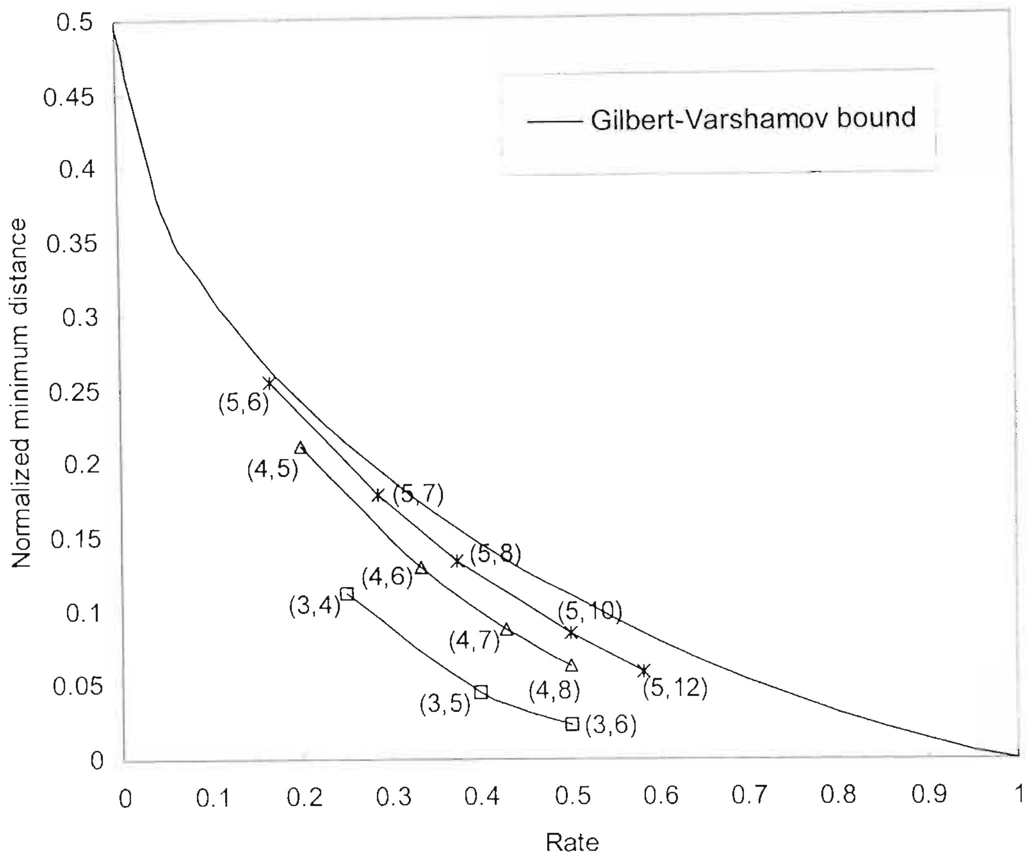

Figure 2.2: Average asymptotic values of the normalized Hamming distance of the ensemble of LDPC codes.

2.1.2.2 LDPC Codes with j = 2 are not Asymptotically Good

Gallager used a graphical argument to prove that LDPC codes with $j = 2$ are not asymptotically good. Let us consider a LDPC code with parameters $(N, 2, k)$. Let us associate a vertex with each coded bit, and consider the N/k single-parity-check equations as "super" edges linking the k vertices representing the coded bits that they involve. Since $j = 2$, any vertex belongs to two "super" edges.

[3] *i.e.* their minimum Hamming distance increases linearly with the length of the code.

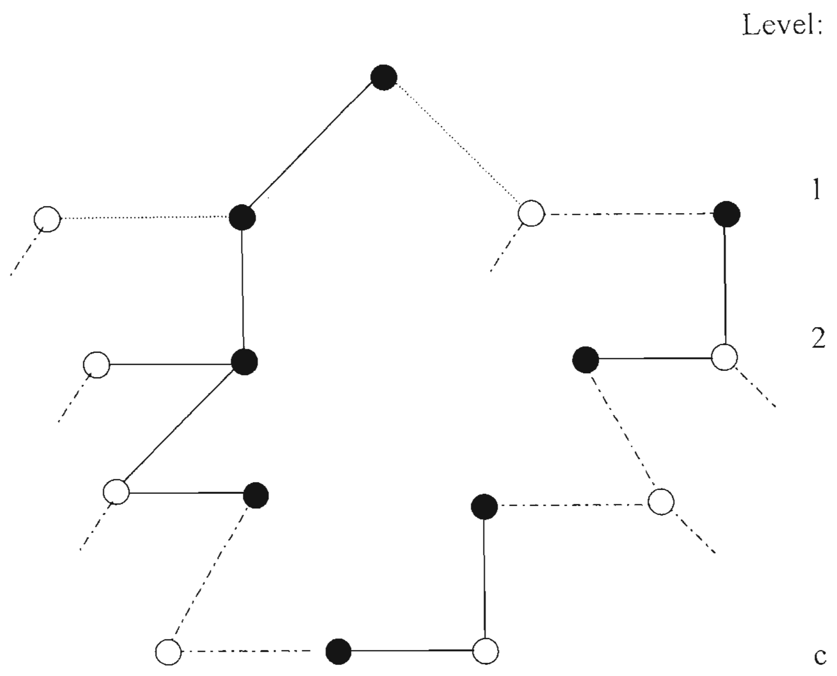

Figure 2.3: Dependency graph of an LDPC code with $j = 2$ and $k = 3$.

Let us choose any vertex and build the equivalent of the dependency graph which stems from it: we follow its two edges, and put its $2(k-1)$ neighbours in the first level. Let us iterate this process with the bits of the first level as shown in Figure 2.3 for $k = 3$. We plot the two different "super" edges of any vertex with two different types of lines in order to distinguish them easily.

Let us assume that the shortest cycle passing through the summit arises at level c. The i-th level ($i < c$) contains $2(k-1)^i$ vertices. We can roughly bound the number of vertices at level $c-1$ as:

$$2(k-1)^{c-1} \leq N, \tag{2.8}$$

which leads to:

$$c \leq 1 + \log_{k-1}\left(\tfrac{N}{2}\right). \tag{2.9}$$

For the shortest cycle, we consider the set of vertices that are at the intersection of the "super" edges in the cycle. They are represented in black in Figure 2.3. The word with all coded bits set to zero, except the ones corresponding to the black vertices that we set to one, is clearly a codeword of C. Its Hamming weight d satisfies:

$$d = 2c\,, \tag{2.10}$$

since there are exactly two black vertices on levels 1 to $c-1$ plus the summit and the last vertex. Hence, we have:

$$d \le 2 + 2\log_{k-1}\left(\tfrac{N}{2}\right). \tag{2.11}$$

We just found a nonzero codeword that has a Hamming weight that increase only logarithmically with N. Hence, LDPC codes with $j = 2$ cannot be asymptotically good.

2.2 Graphical Representation

One of the very few researchers who studied LDPC codes prior to the recent resurgence is Michael Tanner [8]. Tanner considered LDPC codes (and a generalization) and showed how any LDPC code C with parameters (N, j, k) can be represented effectively by a so-called *bipartite graph*[4], now called a Tanner graph. Its properties are the following: Its left part has N vertices, representing the coded bits. Its right part has $M = Nj/k$ vertices, representing the single-parity-check codes. There is an edge connecting a bit vertex to a spcc vertex if this bit belongs to the spcc. Hence the degree[5] of the left part is j, and the degree of the right part is k. Figure 2.4 shows this representation.

[4] A bipartite graph is a graph (nodes or vertices connected by undirected edges) whose nodes may be separated into two classes, and where edges may only connect two nodes residing in the same class. The two classes of nodes in a tanner graph are the coded bit nodes (or the variable nodes) and the check nodes (or function nodes).

[5] The degree of a vertex is the number of edges connecting this vertex to others. All the vertices of a regular graph have the same degree. A regular bipartite graph has two degrees: one for its left part, and one for its right part.

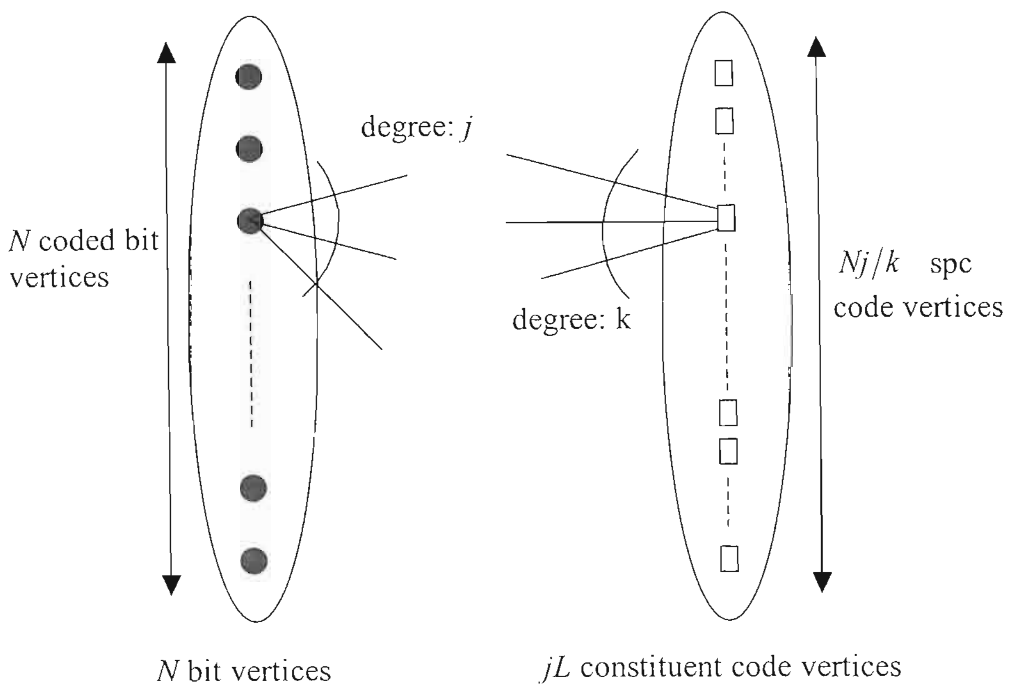

Figure 2.4: Graphical representation of an LDPC code with parameters (N, j, k) as a Tanner random code.

If the parity-check matrix is chosen at random, just satisfying the weight conditions on the rows and columns, the resulting graph is also purely random: the edges are chosen at random, just satisfying the degree conditions on the bit and spcc vertices. Gallager's construction (see Section 2.1.1 and Table 2.1) is slightly more specific. Since each coded bit belongs to one and only one spcc C_0 of any of the j super-codes $C^1, \cdots, C^j$, the right part of the graph is divided in j clumps of N/k spcc vertices. Any coded bit vertex has hence a single edge connecting it to any of the clumps.

Tanner derived bounds linking the minimum Hamming distance, the number of vertices and the girth[6] of the graph, for any compound code defined by graph. Applying these results to LDPC codes is straightforward.

[6] The girth of a Tanner graph is the minimum cycle length of the graph. A cycle of length l in a tanner graph is a path comprised of l edges which closes back on itself.

2.3 Irregular Low-Density Parity-Check Codes

The discussion above was restricted to regular LDPC codes as originally introduced by Gallager. An LDPC code is irregular if the weight per row and/or the weight per column of the parity-check matrix $\boldsymbol{H}$ is not uniform, but instead governed by an appropriately chosen distribution of weights. In terms of the Tanner graph, this means that the degrees of the nodes on each side of the graph can vary widely.

While a good amount of mathematical support exists for the efficacy of irregular codes (see Luby et *al.* [18] and Richardson et *al.* [19]), we provide some intuition as to why they should be more effective than regular codes. Consider trying to build a regular low-density parity-check code that transmits at fixed rate. It is convenient to think of the process as a game, with the messages nodes and the check nodes as the players, and each player trying to choose the right number of edges. A constraint on the game is that the message nodes and the check nodes must agree on the total number of edges. From the point of view of a message node, it is best to have high degree, since the more information it gets from its check nodes the more accurately it can judge what its correct value should be. In contrast, from the point of view of a check node, it is best to have low degree, since the lower the degree of a check node, the more valuable the information it can transmit back to its neighbours.

These two competing requirements must be appropriately balanced. Previous work has shown that for regular graphs, low-degree graphs yield the best performance [10], [14]. If one allows irregular graphs, however, there is significantly more flexibility in balancing these competing requirements. There is reason to believe that a wide spread of degrees, at least for message nodes, could be useful. Messages nodes with high degree tend to correct their value quickly. These nodes then provide good information to the check nodes, which subsequently provide better information to lower degree message nodes. Irregular graph constructions thus have the potential to lead to a wave effect, where high degree message nodes tend to get corrected first, and the message nodes with slightly smaller degree, and so on down the line.

This intuition unfortunately does not provide clues as to how to construct appropriate irregular graphs. A number of researchers have examined the optimal degree distribution among nodes. The results have been spectacular, with performance surpassing the best turbo code [21]. A brief literature survey is presented here.

- In [17, 52], attempts to find the profile by linear programming approach are presented. Given one degree sequence and an initial noise level, a complementary degree sequence for which the probability of bit error goes to zero as the number of decoding iteration increases. This analysis is conducted for a hard decision decoding scheme similar to [3].
- Density evolution is an algorithm for computing the threshold of LDPC codes with iterative decoding [53]. The decoding threshold is defined as the minimum channel E_b/N_o, where E_b is the bit energy and N_o is the one sided noise power spectral density for which the iterative decoding algorithm converges. They convert the infinite-dimensional problem of iteratively calculating message densities, which is needed to find the exact threshold, to a one dimensional problem of updating means of Gaussian densities. This approach allows to calculate the threshold quickly, to understand the behaviour of the decoder better, and to design good irregular LDPC codes for AWGN channels.
- In [59, 19], Richardson and Urbanke presented a general method for determining the capacity of message passing decoders applied to LDPC codes used over binary input memoryless channel with discrete or continuous outputs. They showed that for almost all codes in a suitably defined ensemble, transmission at rates below this capacity results in error probabilities that approach zero exponentially to the code length, whereas for transmission at rates above the capacity the error probability stays bounded away from zero. Then based on this theoretical analysis, they found some codes and provided simulation results in [54].
- In [14, 20], Mackay showed that by using different construction methods, the performances of the codes with the same profile are different.

2.4 Construction of Low-Density Parity-Check Codes

Following their rediscovery, the construction of LDPC codes became a topic of great interest in coding society, and various construction methods have been proposed. Construction of LDPC codes can be classified into two general categories: random and algebraic constructions. In this section, the major methods are briefly reviewed.

2.4.1 Random Construction

Random construction is to construct codes using computer search based on a set of design rules (or guidelines) and required structures of their Tanner graphs, such as the degree distributions of the variable and check nodes. In [10], Mackay proposed a construction method whereby the parity check matrix is generated with a weight γ per column and a uniform weight ρ per row, and with no two columns having overlap greater than 1 (more than one non-zero entries of two different columns at the same row position). Using this constraint, the graph has no cycles of length 4. He also found that there is no significant improvement in the error performance by removing cycles of length 6, 8 and higher.

Random LDPC codes in general do not have sufficient structures such as cyclic or quasi-cyclic structure to allow simple encoding. This lack of any obvious algebraic structure makes the calculation of minimum distance infeasible for long codes, and most analyses focus on the average distance function for an ensemble of LDPC codes. Furthermore, their minimum distances are often poor.

Xiao-Yu Hu et *al.* presented in [50] a simple and efficient non-algebraic, though not really random, method for constructing Tanner graphs having a large girth in a best effort sense by progressively establishing edges between symbol and check nodes in an edge-by-edge manner, called progressive edge-growth (PEG) construction. When constructing a graph with a given variable node degree distribution, the main principle in this method is to optimise the placement of a new edge, connecting a particular symbol node to specific check node on the graph such that the largest possible local girth is achieved. Thus the placement of a new edge on the graph has an impact on the girth as small as possible. After this new edge has been determined, the graph with the new edge is updated, and the procedure continues with the placement of the next edge. The PEG construction yields graphs with large girth that asymptotically guarantees a girth at least as large as the Erdös-Sachs bound [51]. The Erdös-Sachs bound is a non-constructive lower bound on the girth of random graphs and has the same significance as the Gilbert-Varshamov bound does in the context of minimum distance of linear codes.

The advantages of the PEG construction over a random one are twofold. First, it yields a much better girth distribution, thereby facilitating the task of the belief propagation (BP) or sum product algorithm (SPA) during the iterative decoding process. The decoding algorithms for LDPC codes will be explained in detail in the next chapter. Second it leads (or guarantees) a meaningful lower bound on the minimum distance, providing insight into the performance of the code at high signal-to-noise ratios. Simulation results in [50] confirmed that using the PEG algorithm for constructing short-block-length LDPC codes results in a significant improvement compared to randomly constructed codes.

2.4.2 Algebraic Construction

Algebraic construction is to construct structured LDPC codes with algebraic and combinatorial methods. Structured LDPC codes in general have encoding (or decoding) advantage over the random codes in terms of hardware implementation. Well designed structured codes can perform just as well as random codes in terms of bit-error performance, frame-error performance and error floor, collectively. Algebraic construction methods include:

2.4.2.1 Construction Based on Finite Geometries

In [22] Kou et *al.* investigated the construction of LDPC codes from a geometric approach. The construction is based on lines and points of a finite geometry. Well known finite geometries are Euclidean and Projective geometries over finite fields. Based on these two families of finite geometries, four classes of LDPC codes are constructed. Codes of these four classes are either cyclic or quasi-cyclic, and therefore their encoding can be implemented with linear feedback shift registers based on their generator (or characterization) polynomials [23, 24]. This linear time encoding is very important in practice and is not shared by other LDPC codes in general. Codes of these four classes are called finite geometry LDPC codes.

Finite geometry LDPC codes have relatively good minimum distances and their Tanner graphs do not contain cycles of length 4. They can be decoded with various decoding methods, ranging from low to high complexity and from reasonably good to very good performance. Their error performances either have no error floor or have a low error floor. Finite geometry LDPC codes of short to moderate lengths outperform equivalent random computer generated LDPC codes. Long finite geometry LDPC codes, in general, do not perform as close to the Shannon limit as random LDPC codes of the same lengths and rates in the waterfall region, i.e., in the high bit error range. However, they perform equally well in the low bit error range and have lower error floor.

A finite geometry LDPC code can be extended by splitting each column of its parity check matrix into multiple columns. This column splitting results in a new sparse matrix and hence a new LDPC code of longer length. If column splitting is done properly, the extended code performs amazingly well using the sum-product algorithm (SPA) decoding to be described in the next chapter. An error performance only a few tenths of a dB away from the Shannon limit can be achieved. New LDPC codes can also be constructed by splitting each row of the parity check matrix of a finite geometry LDPC code into multiple rows. Combining column splitting and row splitting of the parity check matrices of finite geometry LDPC codes, a large class of LDPC codes with a wide range of code lengths and rates can be obtained. A finite geometry LDPC codes can also be shortened by puncturing the columns of its parity check matrix that correspond to the points on a set of lines or a sub-geometry of the geometry based on which the code is constructed. Shortened finite geometry LDPC codes also perform well with the SPA decoding. For further results see [25-27].

2.4.2.2 Construction Based on Reed-Solomon (RS) Codes

An algebraic method for constructing regular LDPC codes based on *Reed-Solomon* (RS) codes is developed in [28]. The construction method is based on the maximum distance separable (MDS) property of RS codes with two information symbols. It guarantees that the Tanner graphs of constructed LDPC codes are free of cycles of length 4 and hence have girth at least 6. The construction results in a class of regular LDPC codes in Gallager's original form. These codes are simple in structure and have good minimum distances. They perform well with various decoding algorithms. It can be shown that certain subclasses of this class are equivalent to some existing LDPC codes, like Euclidean geometry (EG) Gallager-LDPC codes.

A possible generalization of this construction method is to use a RS code with three information symbols as the base code. In this case, there will be cycles of length 4 in the Tanner graph of the constructed code. These short cycles do not necessarily prevent the code from having good error performance with iterative decoding if the code has large minimum distance and good cycle structure in its Tanner graph.

2.4.2.3 Construction Based on Combinatorial Designs

Several well-structured LDPC codes based on a branch in combinatorial mathematics, known as balanced incomplete block designs (BIBDs) [41] are introduced in [42] and [43]. The bipartite graphs of codes based on BIBDs have girth at least 6 and they perform very well with iterative decoding. Furthermore, several classes of these codes are quasi-cyclic and hence their encoding can be implemented with simple feedback shift registers.

2.4.2.4 Construction Based on Circulant Decomposition

In [44] and [45], an algebraic method for constructing regular LDPC codes based on decomposition of circulant matrices constructed from finite geometries is presented. Codes constructed on this method perform very well with iterative decoding compared to computer generated LDPC codes. Most importantly, these codes are quasi-cyclic and hence their encoding can be implemented in linear time with linear shift-registers. A very interesting and maybe significant discovery in these works is the construction of algebraic codes which have many short cycles in their Tanner graphs but perform amazingly well with iterative decoding and close to the Shannon limit. This discovery contradicts the common belief that for a code to perform well with iterative decoding, its Tanner graph must be free of short cycles (i.e., with a relatively large girth). Its significant implication is that in search for good and easily encodable LDPC codes, it is not necessary to focus on construction of codes that have large girths, which is not very easy. In fact, codes with large girths do not necessarily perform well with iterative decoding if they have poor minimum distances.

2.4.2.5 Construction Based on Graphs

Some twenty years ago G.A. Margulis [46] proposed an algebraic construction of LDPC codes. In [47] the performance of the code proposed by Margulis is analysed. Mimicking the construction of Margulis, the authors described a new powerful regular LDPC code whose construction is based on a Ramanujan graph [48, 49]. The code performance with iterative decoding seems to be in a certain sense better than the performance of a randomly constructed code with the same design parameters.

2.5 Conclusion

In this chapter, we have briefly recalled the construction of the original regular LDPC codes, and their analytical properties as derived by Gallager. Their graphical representation as Tanner codes on random graphs is also presented, and a review of the different construction methods is given.

CHAPTER 3

DECODING OF LOW-DENSITY PARITY-CHECK CODES

In this chapter, different existing decoding methods used for decoding LDPC codes are described and two new decoding algorithms are presented. The first algorithm is a hard-decision decoding method, and the second one is between hard- and soft-decision decoding method. In both algorithms, one bit is flipped in each iteration and the bit to be flipped is chosen according to the fact that, for low-density matrices, the syndrome weight increases with the number of error until error weights much larger than half the minimum distance. The algorithms are independent of the channel characteristics, and offer a good performance-complexity tradeoff for any LDPC code whose parity-check matrix has reasonably large column weights. Simulations results on the additive white Gaussian noise channel comparing the proposed algorithms with other well known decoding algorithms show that the former achieve excellent performances in terms of the bit-error rate, while requiring lower complexity.

This chapter is organised as follows: in the next section an introduction is presented. A brief review of the known decoding methods is given in Section 3.2. The decoding algorithms are presented in Section 3.3. Some simulation results are presented and discussed in Section 3.4, and finally conclusions are drawn in Section 3.5.

3.1 Introduction

LDPC codes can be decoded with various decoding methods, ranging from low to high complexity and from reasonably good to very good performance. The simplest decoding algorithm is *majority-logic* (MLG) decoding and it belongs to hard-decision decoding methods [24]. When applied to LDPC codes, it gives relatively good coding gain only for LDPC codes whose parity-check matrices have relatively large column weight that is close to the minimum distance of the code. Another simple hard-decision decoding algorithm is bit-flipping (BF) and was introduced by Gallager [2, 3]. It provides good performance and is reasonable to be used when LDPC matrix has relatively small row weight. These two hard hard-decision algorithms can be modified in various ways to include reliability information of the received symbols. This increases decoding complexity, but improves the performance. These modifications include: weighted MLG decoding (WMLG) [29], weighted BF decoding (WBF) [30], the algorithm proposed in [31] which we call improved weighted bit flipping (IWBF). Finally, LDPC codes can be decoded using soft-decision decoding algorithms: a posteriori probability (APP) decoding [2], and iterative decoding based on belief propagation (IDBP) (commonly known as sum-product algorithm (SPA)) [14, 19, 32 – 34]. IDBP is using methods on belief propagation over networks [34], but is essentially the same as APP decoding. It offers the best performance, but has the largest complexity. Each decoding iteration requires many real number addition, subtraction, multiplication, division, exponential and logarithm operations. Some approximations of IDBP have been developed which can reduce the complexity with small loss in performance. In [35], a reduced complexity IDBP has been proposed for LDPC codes and is referred to as *uniformly most powerful* (UMP) BP-based algorithm. Unfortunately, the UMP BP-based algorithm does not work well for codes with check sums of larger weight. A normalized IDBP algorithm, which can improve the UMP BP-based algorithm by normalization, is proposed in [36]. However, this algorithm also requires real division operations.

The above decoding methods provide a wide range of trade-offs among decoding complexity, decoding speed, and error performance. MLG and BF decoding belong to hard-decision decoding methods. APP and IDBP/SPA decoding are soft-decision schemes. They require extensive decoding computation but they provide the best error performance. WMLG, WBF, and IWBF decoding are between hard and soft-decision decoding methods. They improve the error performance of the MLG and BF decoding with some additional computational complexity. They offer a good trade-off between error performance and decoding complexity. In particular, simulation results on finite geometry LDPC codes in [31] show that, with IWBF, performances less than 1 dB away from IDBP can be achieved.

In this chapter, two new decoding algorithms for LDPC codes, which are based on the algorithm [166, 167], are introduced. The idea for decoding is the fact that in case of low-density parity-check matrices, the syndrome weight increases with the number of errors in average until errors weights much larger than half the minimum distance (for the analysis, see [167]). Therefore, the idea is to flip one bit in each iteration, and the bit to be flipped is chosen such that the syndrome weight decreases. It should be noted that not only rows of the parity-check matrix can be used for decoding, but in principle all vectors of the dual code with minimum (or small) weight as well. While in [168], the extension of the algorithm in [166, 167] to soft-decision was similar to belief propagation (BP), in our new variant the scaled reliability value from the channel is used only through an addition to the corresponding position in the code after calculating binary operations only. Surprisingly, this low complex method shows excellent performance which can be further improved by the loop detection mechanism. The new algorithms belong to the class of so called bit flipping algorithms. The first decoding algorithm is a hard-decision decoding method. The second algorithm, which is a modification of the first one to include reliability information of the received symbols, is between hard- and soft-decision decoding. In some cases the changing of bits can enter a loop, which means that bits are changed and after some steps are changed back again until infinity if no stop criterion is applied. The loop detection recognizes such behavior and resolves this situation such that in many cases a correct decoding is possible.

3.2 Review of Existing Decoding Algorithms

3.2.1 Majority-Logic Decoding

The first *majority-logic decoding algorithm* was devised in 1954 by Reed [37] for a class of multiple-error-correcting codes discovered by Muller [38]. Reed's algorithm was later generalized and the first unified formulation of majority-logic decoding algorithms was given by Massey [39]. We will describe only one-step *majority-logic decoding*, but the algorithm can be generalized to L steps [24].

Consider an (n,k) block code C with parity-check matrix $\boldsymbol{H}$. The row space of $\boldsymbol{H}$ is an $(n,n-k)$ block code C_d, which is the dual code of C. The inner product of any vector $v \in C$ and any vector $w \in C_d$ is zero:

$$\boldsymbol{w} \cdot \boldsymbol{v} = w_0 v_0 + w_1 v_1 + \cdots + w_{n-1} v_{n-1} = 0. \tag{3.1}$$

The equality (3.1) is called a *parity-check* equation and clearly, there are 2^{n-k} parity-check equations.

Let $\boldsymbol{v}$ be a transmitted binary code vector and $\boldsymbol{z}$ the hard decision of the received sequence. Vector $\boldsymbol{z}$ is also binary. Let $\boldsymbol{e} = (e_0, e_1, \ldots, e_{n-1})$ be the error vector. The *i*th component of the error vector e_i is 0 if $v_i = z_i$ and it is 1 if $v_i \neq z_i$. Therefore, $\boldsymbol{z} = \boldsymbol{v} + \boldsymbol{e}$. For any vector $w \in C_d$, we can form a *parity-check sum*:

$$A = \boldsymbol{w} \cdot \boldsymbol{z} = w_0 z_0 + w_1 z_1 + \cdots + w_{n-1} z_{n-1}. \tag{3.2}$$

If $\boldsymbol{z}$ is a codeword, the parity-check sum A is zero. If $\boldsymbol{z}$ is not a codeword, A may not be zero. Since $\boldsymbol{w} \cdot \boldsymbol{v} = 0$, we have the following relationship between the check sum A and error digits in $\boldsymbol{e}$:

$$A = w_0 e_0 + w_1 e_1 + \cdots + w_{n-1} e_{n-1}. \tag{3.3}$$

An error digit e_l is said to be *checked* by the check sum A if the coefficient $w_l = 1$.

The task of any hard-decision decoding algorithm is to determine the error digits in a way that minimizes the probability of error. Now, we will explain how certain sets of check sums that possess certain properties can be used for error pattern estimation.

Suppose that for every code digit position l, $0 \leq l < n$, there exist J vectors in the dual code C_d,

$$\begin{aligned} \boldsymbol{w}_1 &= (w_{1,0}, w_{1,1}, \ldots, w_{1,n-1}) \\ \boldsymbol{w}_2 &= (w_{2,0}, w_{2,1}, \ldots, w_{2,n-1}) \\ &\vdots \\ \boldsymbol{w}_J &= (w_{J,0}, w_{J,1}, \ldots, w_{J,n-1}) \end{aligned}, \tag{3.4}$$

with the following properties:

1. The lth component of each vector is 1, i.e. $w_{1,l} = w_{2,l} = \cdots = w_{J,l} = 1$.
2. For $i \neq l$, there is at most one vector whose ith component is 1, i.e. at most one element in the set $\{w_{1,i}, w_{2,i}, \ldots, w_{J,i}\}$ can be nonzero.

These J vectors are said to be orthogonal on the lth digit position and they can be used to estimate the lth error digit. The J parity-check sums can be formed based on the J orthogonal vectors and they are related to the error digits as follows:

$$\begin{aligned} A_1 &= w_{1,0}e_0 + w_{1,1}e_1 + \cdots + e_l + \cdots + w_{1,n-1}e_{n-1} \\ A_2 &= w_{2,0}e_0 + w_{2,1}e_1 + \cdots + e_l + \cdots + w_{2,n-1}e_{n-1} \\ &\vdots \\ A_J &= w_{J,0}e_0 + w_{J,1}e_1 + \cdots + e_l + \cdots + w_{J,n-1}e_{n-1} \end{aligned}. \tag{3.5}$$

These J check sums are said to be *orthogonal on the error digit* e_l.

Suppose now that $\lfloor J/2 \rfloor$ or fewer errors occurred and we want to estimate e_l, $0 \leq l < n$. If $e_l = 1$, the other nonzero components of $\boldsymbol{e}$ can be distributed among at most $\lfloor J/2 \rfloor - 1$ check sums orthogonal on e_l. Hence, at least $J - \lfloor J/2 \rfloor + 1$, or *more than one half* of the check sums orthogonal on e_l are equal to $e_l = 1$. Hence at least $J - \lfloor J/2 \rfloor$, or *at least one half* of the check sums orthogonal on e_l are equal to $e_l = 0$. Thus, the value of e_l is equal to the value assumed by a clear majority of the parity-check sums orthogonal on e_l. Based on this discussion, the one-step majority-logic decoding algorithm can be formulated as follows:

The error digit e_l, for $0 \leq l < n$, is decoded as 1 if a clear majority of the parity-check sums orthogonal on e_l is 1; otherwise, e_l is decoded as 0.

Correct decoding of e_l is guaranteed if there are $\lfloor J/2 \rfloor$ or fewer errors. If J is the maximum number of orthogonal parity-check sums that can be formed on every error digit, then, by one-step majority logic decoding, any error pattern of $\lfloor J/2 \rfloor$ or fewer errors can be corrected. For this reason, the parameter $t_{ML} = \lfloor J/2 \rfloor$ is called the *majority-logic error-correcting capability* of the code. If $d_{\min}$ denotes the minimum distance of the code, it is clear that the one-step majority-logic decoding is effective only if $t_{ML} = \lfloor J/2 \rfloor$ is equal or close to the error-correcting capability of the code $t = \lfloor (d_{\min} - 1)/2 \rfloor$. In other words, J should be equal or close to $d_{\min} - 1$.

Let C be an (n,k) LDPC code that does not have cycles of length 4 in its Tanner graph. It can be easily shown that C can be decoded using one-step majority-logic decoding. Let $\boldsymbol{H}$ be the parity-check matrix of C, with m rows $\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{m-1}$. No cycles of length 4 in the Tanner graph of C imply that no two rows in $\boldsymbol{H}$ have more than one 1-component in common. Each row corresponds to one parity-check sum S_i, where $0 \le i < m$. Let S denote the set of check sums corresponding to the rows of $\boldsymbol{H}$, $S = \{S_0, S_1, \ldots, S_{m-1}\}$. Clearly, no two check sums in S can both check on more than one error digit. For every bit position l, we can find the set of check sums orthogonal on e_l, denoted $S^{(l)}$. Let $\gamma(l)$ be the number of 1-components in the lth column of $\boldsymbol{H}$. Then,

$$S^{(l)} = \left\{S_{i_1}, S_{i_2}, \ldots, S_{i_{\gamma(l)}}\right\}, \tag{3.6}$$

where $h_{i_j,l} = 1$ for $\left\{i_1, i_2, \ldots, i_{\gamma(l)}\right\}$.

If C is (γ, ρ)-regular LDPC code, each column of its parity-check matrix has the same number of 1-components γ, and with one-step majority-logic decoding $\lfloor \gamma/2 \rfloor$ or fewer errors can be corrected. $d_{\min}$ of C is thus, at least $\gamma + 1$. However, many regular LDPC codes have small column weight that is not close to $d_{\min} - 1$. Therefore, these codes will not perform well under one-step majority-logic decoding. A class of regular LDPC codes that offers good performance with one-step majority-logic decoding is a class of finite geometry LDPC codes [30].

Irregular LDPC codes have different column weights in their parity-check matrix. If $\gamma_{\min}$ is the smallest column weight, then for every position l we can form at least $\gamma_{\min}$ check sums orthogonal on e_l. Therefore, with one-step majority-logic decoding we can correct $\lfloor \gamma_{\min}/2 \rfloor$ or fewer errors. Optimal irregular LDPC codes usually have very small $\gamma_{\min}$, in most cases $\gamma_{\min} = 2$ [19]. Thus, $\gamma_{\min} + 1$ is usually much smaller than the minimum distance of a code and irregular LDPC codes perform poorly with one-step majority-logic decoding.

3.2.2 Bit Flipping Decoding

Bit flipping decoding algorithm was devised by Gallager [2]. It is a very simple hard decision decoding scheme designed for LDPC codes. The decoder receives hard decision z of the received sequence $\boldsymbol{r}$ and computes parity-check sums according to the parity-check matrix of an LDPC code. Then, it changes, i.e. *flips*, those code bits that are contained in more than some fixed number δ of unsatisfied parity-check sums. The parity-check sums are recomputed and the process is repeated until all parity-check sums are satisfied or a predefined maximum number of iterations is reached. The parameter δ, called threshold, is a design parameter which should be chosen to optimise the error performance while minimizing the number of computations of parity check sums. The value of δ depends on the code parameters ρ, γ, $d_{\min}$ and the signal-to-noise ratio (SNR).

The number of code bits contained in a parity-check sum is small compared to the code length. Also, any two parity-check sums contain very small number of code bits in common, usually none or just one bit in common. Thus, when most of the check sums containing one bit are not satisfied, there is a strong indication that this code bit is in error.

If decoding fails for a given value of δ, then the value of δ can be reduced to allow further decoding iterations. For error patterns with number of errors less than or equal to the error correcting capability of the code, the decoding will be completed in few iterations. Otherwise, more decoding iterations are needed. Therefore, the number of decoding iterations is a random variable and is a function of the channel SNR. A limit may be set on the number of iterations. When this limit is reached, the decoding process is terminated to avoid excessive computations. Due to the nature of low-density parity checks, the above decoding algorithm corrects many error patterns with number of errors exceeding the error correcting capability of the code.

The most efficient version of bit flipping algorithm is the one without threshold δ. The bits that are flipped in each iteration are simply the ones that are contained in the largest number of unsatisfied parity-check sums. It can be described as the following five-step procedure:

Step 1) Compute the parity-check sums, i.e. syndrome bits:

$$\boldsymbol{s} = \left(s_0, s_1, \ldots, s_{M-1}\right) = \boldsymbol{z} \cdot \boldsymbol{H}^T$$

$$s_i = \sum_{j=0}^{N-1} z_j h_{i,j} \quad \text{for} \quad 0 \le i < M.$$

If all the parity check sums are zero, stop the decoding.

Step 2) Find the number of unsatisfied parity-check sums for each bit, denoted n_i, $0 \le i < N$.

Step 3) Find the set Ω of bits for which n_i is the largest.

Step 4) Flip the bits of $\boldsymbol{z}$ belonging to the set Ω.

Step 5) Go back to *Step 1* unless a predefined maximum number of iterations is reached.

Suppose that this simple decoding scheme described above is applied on an LDPC code which does not have cycles of length 4 in its Tanner graph and whose parity-check matrix has column weight at least γ. If only one bit is in error, all the check sums that contain that bit will be unsatisfied. Thus, that bit will be in at least γ unsatisfied check sums. Any other bit can be in at most one unsatisfied check sum, due to the fact that no two columns in the parity-check matrix of the code can have more than one 1-component in common. If we have two bits in error, those two bits will be in at least

$\gamma-1$ unsatisfied check sums, while any other bit can be in at most 2 unsatisfied check sums. In general, if l bits are in error, each of them will be in at least $\gamma-l+1$ unsatisfied check sums, while all the other correct bits can be in at most l unsatisfied check sums. Therefore, if $\gamma-l+1>l$ the above algorithm will be flipping only bits in error. With each iteration, the number of bits in error will decrease and the decoding is guaranteed to be successful. Consequently, the simplest version of bit flipping algorithm guarantees correcting $\lfloor \gamma/2 \rfloor$ errors, the number equal to the majority-logic error correcting capability t_{ML}. However, due to the nature of LDPC codes many error patterns with number of error exceeding the majority-logic error-capability can be corrected as well. Simple bit flipping decoding algorithm can be improved by using adaptive threshold δ, which also increases computational complexity.

3.2.3 Weighted Majority-Logic and Bit Flipping Decodings

The simple hard-decision MLG and BF decodings can be improved to achieve better error performance by including some kind of reliability information (or measure) of the received symbols in their decoding decisions. Of course, additional decoding complexity is required for such performance improvement.

Consider the soft-decision received sequence $y=(y_0,y_1,\ldots,y_{N-1})$. For the AWGN channel, a simple measure of the reliability of a received symbol y_l is its magnitude, $|y_l|$. The larger the magnitude $|y_l|$ is, the larger the reliability of the hard-decision digit z_l is. Many algorithms for decoding linear block codes based on this reliability measure have been devised. In the following, this reliability measure is used to modify the one-step majority logic decoding and the BF decoding.

Again consider an LDPC code specified by a parity-check matrix $\boldsymbol{H}$ with M rows, $\boldsymbol{h}_0,\boldsymbol{h}_1,\ldots,\boldsymbol{h}_{M-1}$. For $0\le l\le N-1$ and $0\le j\le M-1$, define

$$\left|y_j\right|_{\min}^{(l)} \triangleq \left\{\min\{|y_i|\}:0\le i\le N-1, h_{j,i}=1\right\}, \tag{3.7}$$

and

$$E_l \triangleq \sum_{s_j^{(l)} \in S_l} \left(2s_j^{(l)} - 1\right)|y_j|_{\min}^{(l)}, \tag{3.8}$$

where S_l is the set of check sums orthogonal on bit-position l. The value E_l is simply a weighted check sum that is orthogonal on the code bit position l. Let $\boldsymbol{e} = (e_0, e_1, \ldots, e_{N-1})$ be the error pattern to be estimated. Then the one-step MLG decoding can be modified based on the weighted check sum E_l as follows:

$$e_l = \begin{cases} 1, & \text{for } E_l > 0, \\ 0, & \text{for } E_l \leq 0, \end{cases} \tag{3.9}$$

for $0 \leq l \leq N-1$. The above decoding algorithm is called weighted MLG (WMLG) decoding and was first proposed by Kolesnik in 1971 [29] for decoding majority logic decodable codes.

The decision rule given by (3.9) can be used in BF decoding. In this case the decoding is carried out as follows:

Step 1) Compute the check sums. If all the parity-check equations are satisfied, stop the decoding.

Step 2) Compute E_l based on (3.8), for $0 \leq l \leq N-1$.

Step 3) Find the bit position l for which E_l is the largest.

Step 4) Flip the bit z_l.

Step 5) Repeat Step 1 to 4. This process of bit flipping continues until all the parity-check equations are satisfied or a preset maximum number of iterations is reached.

This modified BF algorithm is called weighted BF decoding algorithm (WBF) [22].

The above weighted decoding algorithms are in a way soft-decision decoding algorithms and require real addition operations to compute the weighted check sums, E_l's, to make decisions. Since a real addition operation is much more complex than a logical operation, the computational complexities of both the weighted MLG and BF decodings are dominated by the total number of real additions needed to decode a received sequence.

3.2.4 Improved Weighted Bit Flipping Decoding

Recently, an improvement of the weighted bit flipping decoding was proposed [31]. The proposed algorithm which we call *improved weighted bit flipping* (IWBF), is designed for high-rate finite geometry codes, but we noticed that it offers good performance for any LDPC code whose parity-check matrix has reasonably large column weights. IWBF uses the fact that the codewords of an LDPC code are sparsely distributed in an N-dimensional space over GF(2), where N is the code length. Thus in almost all cases the decoding leads to either a correct codeword or it cannot find any codeword. IWBF flips only one bit in each iteration and the bit to be flipped is chosen according to metric that is supposed to conform well with characteristics of finite geometry LDPC codes, i.e. redundant check sums and somewhat higher column and row weights. Calculation of the bit selection metric requires no knowledge of the signal energy or power of AWGN channel.

Consider a binary (N,K) LDPC code C with $M \times N$ parity-check matrix $\boldsymbol{H}$. In general, $M \geq N-K$ since the parity-check matrix may contain some redundant parity-check sums. For each row i, $0 \leq i < M$, define the following index set:

$$\mathcal{N}_i = \{l : h_{i,l} = 1, 0 \leq l < N\}. \tag{3.10}$$

It is clear that $\mathcal{N}_i$ is the set of bits that participate in the ith parity-check. Similarly, for each code bit, i.e. each column j, $0 \leq j < N$, we define the set of parity-check in which the jth code bit participates:

$$\mathcal{M}_j = \{l : h_{l,j} = 1, 0 \leq l < M\}. \tag{3.11}$$

Suppose the code is used for error control over AWGN channel with zero mean and power spectral density $N_o/2$. Assume binary phase-shift-keying (BPSK) with unit energy. A codeword $c = (c_0, c_1, \ldots, c_{N-1})$ is mapped into bipolar sequence $\boldsymbol{x} = (x_0, x_1, \ldots, x_{N-1})$ before its transmission, where $x_i = (2c_i - 1)$ with $0 \leq i < N$. Let $\boldsymbol{y} = (y_0, y_1, \ldots, y_{N-1})$ be the soft-decision received sequence at the output of the receiver matched filter. For $0 \leq i \leq N-1$, $y_i = x_i + n_i$ where n_i is a Gaussian random variable with zero mean and variance $N_o/2$. An initial binary hard decision of the received sequence, $z^{(0)} = \left(z_0^{(0)}, z_1^{(0)}, \ldots, z_{N-1}^{(0)}\right)$, is determined as follows:

$$z_i^{(0)} = \begin{cases} 1, & \text{if } y_i \geq 0 \\ 0, & \text{if } y_i < 0 \end{cases}. \qquad (3.12)$$

For any tentative binary hard decision z made at the end of each decoding iteration, we can compute the syndrome vector as $s = z \cdot H^T$. We define the log-likelihood ratio (LLR) for each channel output y_i, $0 \leq i \leq N-1$:

$$L_i \triangleq \ln \frac{P(c_i = 1/y_i)}{P(c_i = 0/y_i)}. \qquad (3.13)$$

The absolute value of L_i, $|L_i|$, is called the *reliability* of the initial decision $z_i^{(0)}$. Vectors $z^{(0)}$ and $\boldsymbol{L} = (|L_0|, |L_1|, \ldots, |L_{N-1}|)$ are input to the decoder. For each parity-check sum, i.e., each row i in $\boldsymbol{H}$, $0 \leq i < M$, define *lower check reliability value* l_i and *upper check reliability value* u_i as follows:

$$l_i \triangleq \min_{j \in \mathcal{N}_i} |L_j|, \quad u_i \triangleq \max_{j \in \mathcal{N}_i} |L_j|. \qquad (3.14)$$

Suppose we are starting the *k*th iteration, and that at the end of the $(k-1)$th iteration the tentative binary hard decision was $z^{(k-1)}$ with corresponding syndrome vector $s^{(k-1)} = z^{(k-1)} \cdot H^T \neq \boldsymbol{0}$. In the *k*th, we want to flip one bit in $z^{(k-1)}$ and create a new tentative binary hard decision vector $z^{(k)}$. To chose the bit to flip, we first define for each bit j, $0 \leq j < N$, the cumulative metric over all checks in $\mathcal{M}_j$:

$$\phi_j^{(k)} \triangleq \sum_{i \in \mathcal{M}_j} \phi_{j,i}^{(k)}, \quad 0 \leq j < N, \qquad (3.15)$$

where for each check $i \in \mathcal{M}_j$

$$\phi_{j,i}^{(k)} \triangleq \begin{cases} |L_j| - l_i/2, & \text{if } s_i^{(k-1)} = 0, \\ |L_j| - (u_i + l_i/2), & \text{if } s_i^{(k-1)} = 1. \end{cases} \qquad (3.16)$$

The bit to be flipped is the one that has the smallest cumulative metric. Let $f^{(k)}$ denote the bit position to be flipped at the *k*th iteration. Then,

$$f^{(k)} = \arg\min_{0 \le j < N} \phi_j^{(k)} . \tag{3.17}$$

After flipping bit at position $f^{(k)}$ in $z^{(k-1)}$, we obtain new tentative hard decision $z^{(k)}$ and update the syndrome vector to $\boldsymbol{s}^{(k)}$. If $\boldsymbol{s}^{(k)} = \boldsymbol{0}$, we have a valid codeword and the decoding stops. Otherwise, we star another decoding iteration.

The bit to be flipped is chosen not only based on the number of unsatisfied check sums each bit is contained in, but also based on the reliability of each bit with respect to the reliabilities of the most and the least reliable bits that are contained with it in the same unsatisfied check sum. Since the metric $\phi_j^{(k)}$ maintains linearity with respect to $|L_j|$, for AWGN channel $|L_j|$ can be replaced with $|y_j|$. Thus no knowledge of signal energy or noise power is required.

In the algorithm described above, the metric $\phi_j^{(k)}$, and thus $f^{(k)}$, are functions of $z^{(k-1)}$ and $\boldsymbol{L}$. Therefore, if $z^{(k)} = z^{(k_0)}$ for some $k_0 < k$, $f^{(k)} = f^{(k_0)}$ and $z^{(k+p)} = z^{(k_0+p)}$ for any $\rho > 0$. The algorithm enters a loop, and since a codeword was not found until the *k*th iteration, it will not be found if the search continues. The loop can be detected and avoided by selecting the bit to be flipped as the one that has the next smallest value of $\phi_j^{(k)}$, instead of selecting the one with minimum $\phi_j^{(k)}$ that would result in a loop.

Loop detection can be done easily. Let us define the vector sum:

$$\boldsymbol{E}(x) \triangleq \sum_{i=x+1}^{k} \boldsymbol{e}_{f^{(k)}} , \tag{3.18}$$

where $\boldsymbol{e}_{f^{(k)}}$ is the *N*-dimensional unit vector, i.e. a vector with "1" at position $f^{(k)}$ and "0" everywhere else. It is easy to show that:

$$z^{(k)} = z^{(k_0)} + \sum_{i=k_0+1}^{k} \boldsymbol{e}_{f^{(k)}} = z^{(k_0)} + \boldsymbol{E}(k_0) , \tag{3.19}$$

for any $0 \le k_0 < k$. Thus,

$$z^{(k)} = z^{(k_0)} \quad \text{if and only if} \quad \boldsymbol{E}(k_0) = 0 . \tag{3.20}$$

This means that, to detect and prevent infinite loops, we need only the vectors $\boldsymbol{E}(k-1)$, $\boldsymbol{E}(k-2)$, $\cdots$, $\boldsymbol{E}(0)$. If any of these vectors is zero, an infinite loop is detected and the bit selection $f^{(k)}$ should be discarded. For $0 \le l < k$, $\boldsymbol{E}(l)$ can be computed iteratively as follows:

$$\boldsymbol{E}(l-1) = \begin{cases} \boldsymbol{e}_{f^{(k)}}, & \text{if } l = k \\ \boldsymbol{E}(l) + \boldsymbol{e}_{f^{(l)}}, & \text{otherwise.} \end{cases} \tag{3.21}$$

To compare $\boldsymbol{E}(l)$, $0 \le l < k$, with the all-zero vector, we denote its *Hamming weight* by $wt(l)$. Then, $wt(l-1)$ can be derived from $wt(l)$ as follows:

$$wt(l-1) = \begin{cases} 1, & \text{if } l = k \\ wt(l)-1, & \text{if } f^{(l)}\text{th bit of } \boldsymbol{E}(l) = 1 \\ wt(l)+1, & \text{if } f^{(l)}\text{th bit of } \boldsymbol{E}(l) = 0. \end{cases} \tag{3.22}$$

Of course, $\boldsymbol{E}(l) = \mathbf{0}$ if and only if $wt(l) = 0$.

In summary, IWBF decoding algorithm that incorporates loop detection and prevention can be described as the following procedure:

Step 1) *Initialization*: Set iteration counter $k = 0$. Calculate $z^{(0)}$. For each i, $0 \le i < M$, calculate $l_i/2$ and $u_i + l_i/2$. Set the exclusion list containing bit positions that cause loop $B = \varnothing$.

Step 2) Calculate syndrome $\boldsymbol{s}^{(k)}$. If $\boldsymbol{s}^{(k)} = \mathbf{0}$, stop the decoding and return $z^{(k)}$.

Step 3) $k \leftarrow k+1$. If $k > k_{\max}$, where $k_{\max}$ is the maximum number of number of iterations, declare decoding failure and stop the decoding.

Step 4) For each j, $0 \le j < N$, calculate $\phi_j^{(k)}$.

Step 5) Find $f^{(k)} = \arg \min_{0 \le j < N, j \notin B} \phi_j^{(k)}$.

Step 6) Calculate $\boldsymbol{E}(l)$, $wt(l)$ iteratively for $0 \le l < k$. If $wt(l) = 0$ for any l, $0 \le l < k$, set $B \leftarrow B \cup \{f^{(k)}\}$ and go back to Step 5.

Step 7) Determine $z^{(k)}$ by flipping the bit at position determined in previous steps. Set $B = \varnothing$ and go back to Step 2.

3.2.5 Iterative Decoding based on Belief Propagation or Sum-Product Algorithm

In 1996, LDPC codes were rediscovered [10]. To achieve near Shannon limit performance, *belief propagation* (BP) [34] was used for decoding. The goal of the BP decoder is to compute marginal *a posteriori* probability $P(c_i = 1/y, S)$ that code bit at position i is 1, conditional on the set of the received symbols y and on the event S that the transmitted digits satisfy the set of parity-checks given by the parity-check matrix $\boldsymbol{H}$. An algorithm for the computation of such marginal probabilities exists for belief networks [34]. The algorithm is exact when the underlying graph has no cycles. When applied to an LDPC code, the obtained marginal probabilities will not be exact due to many cycles in the corresponding Tanner graph. Assuming that the errors introduced due to the cycles are small, belief propagation was nevertheless applied [10]. This assumption is reasonable, since as the size of the code increases, it becomes increasingly easy to produce codes in which there are no cycles of length smaller than or equal to a specified length.

Iterative decoding based on belief propagation (IDBP), commonly known as sum-product algorithm (SPA), is a symbol-by-symbol soft-in/soft-out decoding algorithm, the same as maximum *a posteriori* (MAP) decoding [40]. It processes received symbols iteratively to improve the reliability of each decoded symbol. The processing is based on the parity-check matrix which specifies the code. After each iteration, we check whether the stopping criterion is satisfied. If the stopping criterion is not satisfied, we use the outputs of the previous iteration as inputs for the next iteration, and continue the decoding iteration process. If the stopping criteria is satisfied, the decoding stops and the hard decisions are made based on reliability values for each code bit.

Consider an LDPC code C of length N specified by a parity-check matrix $\boldsymbol{H}$ with M rows and N columns. For each check sum, i.e. for each row i, $0 \leq i < M$, we define the set $\mathcal{N}_i$ of code bits that participate in the *i*th check sum as given in (3.10). For each code bit, i.e. for each column j, $0 \leq j < N$, we define the set $\mathcal{M}_j$ of parity-checks in which the *j*th code bit participates as given in (3.11).

The algorithm associates quantities $q_{i,j}$ and $r_{i,j}$ with every non-zero entry in the parity-check matrix, and updates these quantities iteratively. It consists of two steps which are repeated iteratively. They are called *horizontal* and vertical *pass*. Horizontal pass updates quantities $r_{i,j}$, while vertical pass updates quantities $q_{i,j}$. Initial values of $q_{i,j}$ are determined by *a posteriori* probabilities at the channel output. For each nonzero entry in the parity-check matrix, we have in fact, two $q_{i,j}$ quantities, $q_{i,j}^0$, and $q_{i,j}^1$, but they can be computed from one another and their sum is 1. The same is true for $r_{i,j}$.

In the *l*th iteration, quantity $q_{i,j}^x$ represents the pseudo-posterior probability that the *j*th code bit has value x given the information coming from code bits that are distance at most $2l$ from corresponding variable node in the Tanner graph of a code, excluding code bits that are connected to the *j*th code bit trough the *i*th check-sum.

The quantity $r_{i,j}^x$ represents the probability that the *i*th check sum is satisfied given that the *j*th code bit has value x and that the other code bits participating in the check sum have value x', $x' \in \{0,1\}$, with probability $q_{i,j'}^{x'}$, where $j' \in \mathcal{N}_i$ and $j' \notin j$.

Initialization: Let p_i^0 be the prior probability that code bit v_i is 0, and let $p_i^1 = 1 - p_i^0$ be the prior probability that code bits v_i is 1. In the case of AWGN channel, these values are $p_i^0 = P(v_i = 0/y_i)$ and $p_i^1 = P(v_i = 1/y_i)$, i.e. probabilities that the transmitted code bit at position i is 0 and 1, respectively, conditional on the received value corresponding to the *i*th bit. For every code bit at position j, $0 \le j < N$, we initialise $q_{i,j}^0 = p_j^0$ and $q_{i,j}^1 = p_j^1$, for all $i \in \mathcal{M}_j$.

Horizontal pass: In the horizontal step of the algorithm, we run through the parity-checks and for each parity-check, i.e. for each row i, and for each $j \in \mathcal{N}_i$, we compute two probabilities. One is the conditional probability that the *i*th check sum is satisfied, given that the *j*th transmitted bit is 0 and the other bits in the set $\mathcal{N}_i, \{c_{j'} / j' \in \mathcal{N}_i, j' \neq j\}$, have separable distribution given by the probabilities $\{q_{i,j'}^{0}, q_{i,j'}^{1}\}$:

$$r_{i,j}^{0} = \sum_{\{c_{j'} \mid j' \in \mathcal{N}_i, j' \neq j\}} P\left(S_i \mid c_j = 0, \{c_{j'} \mid j' \in \mathcal{N}_i, j' \neq j\}\right) \prod_{j' \in \mathcal{N}_i, j' \neq j} q_{i,j'}^{c_{j'}}, \tag{3.23}$$

where S_i denotes the event that the ith check sum is satisfied. The other probability is the conditional probability that the ith check sum is satisfied, given that the jth transmitted bit is 1 and that the other bits in the set $\mathcal{N}_i$ have separable distribution:

$$r_{i,j}^{1} = \sum_{\{c_{j'} \mid j' \in \mathcal{N}_i, j' \neq j\}} P\left(S_i \mid c_j = 1, \{c_{j'} \mid j' \in \mathcal{N}_i, j' \neq j\}\right) \prod_{j' \in \mathcal{N}_i, j' \neq j} q_{i,j'}^{c_{j'}}. \tag{3.24}$$

These two probabilities can be efficiently computed if we consider $\delta r_{i,j} = r_{i,j}^{0} - r_{i,j}^{1}$ instead of $r_{i,j}^{0}$ and $r_{i,j}^{1}$ separately. Then we have the following equality:

$$\delta r_{i,j} = \prod_{j' \in \mathcal{N}_i, j' \neq j} \delta q_{i,j'}, \tag{3.25}$$

where $\delta q_{i,j} = q_{i,j}^{0} - q_{i,j}^{1}$. Once $\delta r_{i,j}$ is computed, $r_{i,j}^{0}$ and $r_{i,j}^{1}$ are computed as follows:

$$r_{i,j}^{0} = \frac{1}{2}\left(1 + \delta r_{i,j}\right), \tag{3.26}$$

$$r_{i,j}^{1} = \frac{1}{2}\left(1 - \delta r_{i,j}\right). \tag{3.27}$$

Vertical pass: In the vertical step of the algorithm, we update $q_{i,j}^{0}$ and $q_{i,j}^{1}$ based on computed $r_{i,j}^{0}$ and $r_{i,j}^{1}$ in the previous step:

$$q_{i,j}^{0} = \alpha_{i,j} p_j^{0} \prod_{i' \in \mathcal{M}_j, i' \neq i} r_{i',j}^{0}, \tag{3.28}$$

$$q_{i,j}^{1} = \alpha_{i,j} p_j^{1} \prod_{i' \in \mathcal{M}_j, i' \neq i} r_{i',j}^{1}, \tag{3.29}$$

where $\alpha_{i,j}$ is a constant such that $q_{i,j}^{0} + q_{i,j}^{1} = 1$.

After the vertical step, we also compute the pseudo-posteriori probabilities for each code bit j, with $0 \leq j < N$, given by:

$$P\left(c_j = 0 \mid y\right) = \alpha_j p_j^{0} \prod_{i \in \mathcal{M}_j} r_{i,j}^{0}, \tag{3.30}$$

$$P(c_j = 1 \mid y) = \alpha_j p_j^1 \prod_{i \in \mathcal{M}_j} r_{i,j}^1 , \tag{3.31}$$

where again α_j is chosen so that $P(c_j = 0 \mid y) + P(c_j = 1 \mid y) = 1$. Based on these probabilities, we can form a vector $z = (z_0, z_1, \ldots, z_{N-1})$ as candidate for the decoded codeword:

$$z_j = \begin{cases} 1, & \text{for } P(c_j = 1 \mid y) > 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{3.32}$$

From the determined candidate codeword z, $z \cdot \boldsymbol{H}^T$ is computed. If $z \cdot \boldsymbol{H}^T = \boldsymbol{0}$, the candidate codeword is indeed a codeword and the iterative decoding process is stopped. Then z is given at the output of the decoder as the decoded codeword. If $z \cdot \boldsymbol{H}^T \neq \boldsymbol{0}$, the iterative decoding continues, i.e. we are going back to the horizontal pass with updated values $\delta q_{i,j} = q_{i,j}^0 - q_{i,j}^1$ and we compute new values for $r_{i,j}^0$ and $r_{i,j}^1$.

Iterative decoding based on belief propagation is computationally expensive. Each decoding iteration requires many real number computations. If decoding of a particular code with this algorithm converges slowly, a large number of iterations is required to achieve the desire performance. This means that the number of computations is large as well as the decoding delay. In the next section, two reduced complexity IDBP algorithms [35] are given. Both are based on the fact that IDBP basically consists of many maximum *a posteriori* (MAP) decoders for a single parity-check (SPC) code, one corresponding to each parity-check sum, working in parallel. The MAP algorithm for SPC codes and its approximation are described in Appendix A.

3.2.5.1 Reduced Complexity Iterative Decoding

IDBP for binary LDPC codes can be shown to be iterative parallel MAP-SPC decoding. This allows us to apply existing approximations for MAP-SPC to IDBP and yields a reduced complexity decoding algorithm.

Suppose we have M MAP-SPC decoders, one for each check sum of the parity-check matrix $\boldsymbol{H}$ of an LDPC code. For each code bit at position j, we can compute $|\mathcal{M}_j|$

corresponding extrinsic log-likelihood ratios (LLRs), one for each check-sum that contains the *j*th bit, where $|\mathcal{M}_j|$ denotes the cardinality of the set $\mathcal{M}_j$, i.e. the number of 1-components in the *j*th column of the parity-check matrix. For every code bit at position j, $0 \le j < N$, and every check sum $i \in \mathcal{M}_j$, let $L_{i,j}$ denote the LLR corresponding to the *j*th bit that is used as input to the *i*th MAP-SPC decoder.

For each $0 \le j < N$ and each $i \in \mathcal{M}_j$, initialize $L_{i,j}$ with the channel LLR for code bit at position j, $L_c(c_j) = \ln\left(P(c_j = 1 \mid y_j) / P(c_j = 0 \mid y_j)\right)$. Suppose that during the first step of the iterative decoding, for $0 \le i < M$, the output of the *i*th MAP-SPC decoder gives the extrinsic LLR for every code bit $j \in \mathcal{N}_i$ as follows:

$$L_{i,j}^{ex} = 2 \cdot (-1)^{|\mathcal{N}_i|} \cdot \tanh^{-1}\left(\prod_{j' \in \mathcal{N}_i, j' \neq j} \tanh\left(\tfrac{L_{i,j'}}{2}\right) \right). \tag{3.33}$$

Suppose that during the second step of the iterative decoding, for every code bit j, $0 \le j < N$, and every check sum $i \in \mathcal{M}_j$, $L_{i,j}$ is updated by adding to the channel LLR $L_c(c_j)$ extrinsic values at the output of all MAP-SPC decoders corresponding to check-sums containing the *j*th bit, except the output of the *i*th decoder:

$$L_{i,j} = L_c(c_j) + \sum_{i' \in \mathcal{M}_j, i' \neq i} L_{i',j}^{ex}. \tag{3.34}$$

It can be shown that the following relationships exist between $\delta q_{i,j}$ and $L_{i,j}$, $\delta r_{i,j}$ and $L_{i,j}^{ex}$:

$$\delta q_{i,j} = \tanh\left(-\tfrac{L_{i,j}}{2}\right), \tag{3.35}$$

$$\delta r_{i,j} = \tanh\left(-\tfrac{L_{i,j}^{ex}}{2}\right). \tag{3.36}$$

Using these relationships, it is straightforward to obtain equation (3.33) from (3.25), as well as equation (3.34) from (3.28) and (3.29). Thus the above two step iterative decoding is equivalent to IDBP. However, it is usually referred to as the *sum-product algorithm* (SPA).

Based on the approximation for the MAP-SPC decoding given in Appendix A, we obtain the following approximation of the horizontal step:

$$L_{i,j}^{ex} \simeq (-1)^{|\mathcal{N}_i|} \cdot \prod_{j' \in \mathcal{N}_i, j' \neq j} \operatorname{sign}\left(L_{i,j'}\right) \cdot \min_{j' \in \mathcal{N}_i, j' \neq j} \left\{\left|L_{i,j'}\right|\right\}. \tag{3.37}$$

If the channel is AWGN channel, and all the code bits are statistically independent with the same probability of 0 and 1, we can replace $P\left(c_j = x \mid y_j\right)$ with $P\left(y_j \mid c_j = x\right)$, where $x \in \{0,1\}$. Thus $L_c\left(c_j\right) = \ln\left(P\left(y_j \mid c_j = 1\right) / P\left(y_j \mid c_j = 0\right)\right) = 4y_j / N_o$, where $N_o/2$ is the noise variance. The term $4/N_o$ can be factored out of all the equations. Therefore, the algorithm does not depend on the noise variance, and is *uniformly most powerful* (UMP). The algorithm is referred to as UMP BP-based iterative decoding algorithm [35].

Additional approximation can be obtained if we modify the vertical pass as follows:

$$L_{i,j} = L_j = L_c\left(x_j\right) + \sum_{i' \in \mathcal{M}_j} L_{i',j}^{ex} \quad \text{for all } i \in \mathcal{M}_j. \tag{3.38}$$

In this way, LLRs corresponding to bit j that are passed to the MAP decoders are the same for every $i \in \mathcal{M}_j$. This approximation leads to an algorithm called UMP *a posteriori* probability (APP) based iterative decoding algorithm [35].

3.3 Two New Decoding Algorithms for Low-Density Parity-Check Codes

In this section, two new decoding algorithms are presented. The first algorithm is a hard-decision decoding method, and the second one is between hard- and soft-decision decoding methods. For the sake of convenience and completeness, some notations and definitions already introduced are repeated here.

3.3.1 Notation and Basic Definitions

A binary LDPC code is completely described by its sparse binary parity parity-check matrix $\boldsymbol{H}$. For an (N,K) LDPC code, $\boldsymbol{H}$ has N columns and $M \geq N-K$ rows. For a regular LDPC code, $\boldsymbol{H}$ has a constant column weight γ and a constant row weight ρ.

Suppose a binary (N,K) LDPC code is used for error control over a binary-input additive white Gaussian noise (BIAWGN) channel with zero mean and power spectral density $N_o/2$. N is the code length and K the code dimension. Assume binary phase-shift-keying (BPSK) signaling with unit energy. A codeword $\mathbf{c}=(c_0,c_1,\ldots,c_{N-1}) \in \{GF(2)\}^N$ is mapped into bipolar sequence $\boldsymbol{x}=(x_0,x_1,\ldots,x_{N-1})$ before its transmission, where $x_i=(2c_i-1)$ with $0 \leq i \leq N-1$. Let $\boldsymbol{y}=(y_0,y_1,\ldots,y_{N-1})$ be the soft-decision received sequence at the output of the receiver matched filter. For $0 \leq i \leq N-1$, $y_i = x_i + n_i$ where n_i is a Gaussian random variable with zero mean and variance $N_o/2$. An initial binary hard decision of the received sequence, $z^{(0)}=\left(z_0^{(0)},z_1^{(0)},\ldots,z_{N-1}^{(0)}\right)$, is determined as follows:

$$z_i^{(0)} = \begin{cases} 1, & \text{if } y_i \geq 0 \\ 0, & \text{if } y_i < 0 \end{cases}. \tag{3.39}$$

For any tentative binary hard decision z made at the end of each decoding iteration, we can compute the syndrome vector as $\boldsymbol{s}=z \cdot \boldsymbol{H}^T$. We define the log-likelihood ratio (LLR) for each channel output y_i, $0 \leq i \leq N-1$:

$$L_i \triangleq \ln \frac{P(c_i=1 \mid y_i)}{P(c_i=0 \mid y_i)}. \tag{3.40}$$

The absolute value of L_i, $|L_i|$, is called the reliability of the initial decision $Z_i^{(0)}$. For any binary vector $\boldsymbol{v}=\left(v_0,v_1,\ldots,v_{N-1}\right)$, let $wt(\boldsymbol{v})$ be the Hamming weight of $\boldsymbol{v}$. Let $\boldsymbol{u}_i$ be the N-dimensional unit vector, i.e. a vector with "1" at the ith position and "0" everywhere else.

3.3.2 Algorithm I

Step 1) *Initialization*: Set iteration counter $k=0$. Calculate $z^{(0)}$ and $S^{(0)} = wt(z^{(0)} \cdot \boldsymbol{H}^T)$.

Step 2) If $S^{(k)} = 0$, then go to Step 7).

Step 3) $k \leftarrow k+1$. For each $i = 0,1,\ldots,N-1$, calculate $S_i^{(k)} = wt\left((z^{(k-1)} + \boldsymbol{u}_i) \cdot \boldsymbol{H}^T\right)$.

Step 4) Find $j^{(k)} \in \{0,1,\ldots,N-1\}$ with $j^{(k)} = \arg\min\limits_{0 \le i < N} S_i^{(k)}$.

Step 5) If $j^{(k)} = j^{(k-1)}$, then go to Step 8).

Step 6) Calculate $z^{(k)} = z^{(k-1)} + \boldsymbol{u}_{j^{(k)}}$ and $S^{(k)} = wt\left(z^{(k)} \cdot \boldsymbol{H}^T\right)$. Go to Step 2).

Step 7) Stop the decoding and return $z^{(k)}$.

Step 8) Stop the decoding and return $z^{(k-1)}$.

So the algorithm flips only one bit at each iteration and the bit to be flipped is chosen according to the fact that, on average, the weight of the syndrome increases with the weight of the error. The criterion for the decision of whether or not a position is in error differs from Gallager's algorithm because here the syndrome weights are compared where the numbers of errors differ by two. Note that in some cases, the decoder can choose a wrong position j, and thus introduce a new error. But there is still a high likelihood that this new error will be corrected in some later step of the decoding.

3.3.3 Algorithm II

The above algorithm can be modified, with almost no increase in complexity, to achieve better error performance, by including some kind of reliability information (or measure) of the received symbols. Many algorithms for decoding linear block codes based on this reliability measure have been devised.

Consider the soft-decision received sequence $\boldsymbol{y} = (y_0, y_1, \ldots, y_{N-1})$. For the AWGN channel, a simple measure of the reliability, $|L_i|$, of a received symbol y_i is its magnitude, $|y_i|$. The larger the magnitude $|y_i|$ is, the larger the reliability of the hard-decision digit z_i is. If the reliability of a received symbol y_i is high, we want to prevent the decoding algorithm from flipping this symbol, because the probability of this symbol being erroneous is less than the probability of this symbol being correct. This can be achieved by appropriately increasing the values S_i in the decoding algorithm. The solution we propose is to increase the values of S_i by the following term:

$$\alpha \times \gamma \times |L_i|, \tag{3.41}$$

where $\alpha > 0$ is some coefficient to be selected. This solution, which is very simple, happens to be very efficient as will be shown by the simulation results in Section 4. Our intuitive approach is justified by the following facts:

- A larger $|L_i|$ implies that the hard decision z_i is more reliable.
- If there is no overlap between the "non zero" elements of the different columns of $\boldsymbol{H}$, each additional column would increase the syndrome weight by γ.
- For different values of the signal-to-noise ratio (SNR), the values of $|L_i|$ are not the same. Hence the presence of the weighting factor.

The steps of the soft version of the decoding algorithm are described in detail below:

Step 1) *Initialization*: Set iteration counter $k = 0$. Calculate $z^{(0)}$ and $S^{(0)} = wt(z^{(0)} \cdot \boldsymbol{H}^T)$.

Step 2) If $S^{(k)} = 0$, then go to Step 7).

Step 3) $k \leftarrow k+1$. For each $i = 0, 1, \ldots, N-1$, calculate

$$S_i^{(k)} = wt\left((z^{(k-1)} + \boldsymbol{u}_i) \cdot \boldsymbol{H}^T\right) + \alpha \times \gamma \times |L_i|.$$

Step 4) Find $j^{(k)} \in \{0, 1, \ldots, N-1\}$ with $j^{(k)} = \arg\min_{0 \le i < N} S_i^{(k)}$.

Step 5) If $j^{(k)} = j^{(k-1)}$, then go to Step 8).

Step 6) Calculate $z^{(k)} = z^{(k-1)} + \boldsymbol{u}_{j^{(k)}}$ and $S^{(k)} = wt\left(z^{(k)} \cdot \boldsymbol{H}^T\right)$. Go to Step 2).

Step 7) Stop the decoding and return $z^{(k)}$.

Step 8) Stop the decoding and return $z^{(k-1)}$.

It is important to point out that, in both algorithms, the maximum number of iteration needs not to be specified. The algorithms have an inherent stopping criterion. The decoding stops either when a valid codeword is obtained (Step 2) or when the minimum syndrome weight at the *k*th iteration and the minimum syndrome weight at the *(k*-1*)*th iteration are found in the same position (Step 5). This has also been verified through simulations.

3.3.4 Loop Detection

Let $j^{(k)}$ denote the bit position to be flipped at the *k*th iteration. It is clear that if $j^{(k)} = j^{(k-1)}$, then $z^{(k)} = z^{(k-2)}$ and $z^{(k+\sigma)} = z^{(k-2+\sigma)}$ for any $\sigma > 0$. The algorithm enters a loop, and since a valid codeword was not found until the *k*th iteration, it will not be found if the decoding continues. The loop can be detected and avoided by selecting the bit to be flipped as the one that has the next smallest value of $S_i^{(k)}$, instead of selecting the one with minimum $S_i^{(k)}$ that would result in a loop. A loop detection technique is introduced in [31] and described in the previous section. It can be applied to any search algorithm. In the sequel, a somewhat easier implementation is presented.

Suppose that we are in the *k*th iteration, that positions of bits flipped so far are $j^{(1)}, j^{(2)}, \dots, j^{(k-1)}$, and that the bit position selected for flipping at the *k*th iteration is $j^{(k)}$. For each position j, $0 \le j < N$, and for each k', $1 \le k' \le k$, we compute the number, $n_j^{(k')}$, of times it appears in the set $\left\{ j^{(k')}, j^{(k'+1)}, \dots, j^{(k)} \right\}$ modulo 2. Next we compute the weight of the binary sum $z^{(k'-1)} + z^{(k)}$, $w^{(k')}$, as follows:

$$w^{(k')} = \sum_{0 \le j < N} n_j^{(k')} . \tag{3.42}$$

If $w^{(k')} = 0$ for any k', that indicates the loop.

For $0 \le j < N$, $n_j^{(k')}$ can be computed iteratively. Let $\boldsymbol{n}^{(k')} = \left(n_0^{(k')}, n_1^{(k')}, \dots, n_{N-1}^{(k')} \right)$, we have:

$$n_j^{(k')} = \begin{cases} n_j^{(k'+1)} + 1 \pmod 2, & \text{if } j = j^{(k')} \\ n_j^{(k'+1)}, & \text{if } j \neq j^{(k')} \end{cases}, \tag{3.43}$$

for $1 \le k' < k$, with initial condition $n_j^{(k)} = 1$ for $j = j^{(k)}$, and $n_j^{(k)} = 0$ for $j \neq j^{(k)}$. We can also compute $w^{(k')}$ iteratively as follows:

$$w^{(k')} = \begin{cases} w^{(k'+1)} + 1, & \text{if } n_{j^{(k')}}^{(k'+1)} = 0 \\ w^{(k'+1)} - 1, & \text{if } n_{j^{(k')}}^{(k'+1)} = 1 \end{cases}, \tag{3.44}$$

for $1 \le k' < k$, with initial condition $w^{(k)} = 1$.

In summary, Algorithm II described above that incorporates loop detection and prevention is as follows:

Step 1) *Initialization*: Set iteration counter $k = 0$. Calculate $z^{(0)}$ and $S^{(0)} = wt(z^{(0)} \cdot \boldsymbol{H}^T)$. Set the exclusion list containing bit positions that cause loop $B = \varnothing$.

Step 2) If $S^{(k)} = 0$, then go to Step 8).

Step 3) $k \leftarrow k + 1$. If $k > k_{\max}$, where $k_{\max}$ is the maximum number of iterations, go to Step 9).

Step 4) For each $i = 0, 1, \ldots, N-1$, calculate $S_i^{(k)} = wt\left((z^{(k-1)} + \boldsymbol{u}_i) \cdot \boldsymbol{H}^T\right) + \alpha\gamma |L_i|$.

Step 5) Find $j^{(k)} \in \{0, 1, \ldots, N-1\}$ with $j^{(k)} = \arg \min_{0 \le i < N, i \notin B} S_i^{(k)}$.

Step 6) Calculate $\boldsymbol{n}^{(k')}$ and $w^{(k')}$ iteratively for $1 \le k' \le k$. If $w^{(k')} = 0$ for any k', set $B \cup \{j^{(k)}\}$ and go back to Step 5).

Step 7) Calculate $z^{(k)} = z^{(k-1)} + \boldsymbol{u}_{j^{(k)}}$ and $S^{(k)} = wt\left(z^{(k)} \cdot \boldsymbol{H}^T\right)$. Go to Step 2).

Step 8) Stop the decoding and return $z^{(k)}$.

Step 9) Stop the decoding and return $z^{(k-1)}$.

It is necessary to point out that when loop detection is used, the algorithm no longer has an inherent stopping criterion. Thus the maximum number of iteration needs to be specified.

3.3.5 Computational Complexity

Neglecting all operations associated with modulo-2 arithmetic as conventionally done, the computational complexity of the second algorithm described above can be calculated as follows. During each iteration, $N-1$ real comparisons are required to select the bit to be flipped. After the bit is flipped, $\gamma \times \rho$ operations are required to update the syndrome. The terms $\alpha \times \gamma \times |L_i|$ need to be calculated only once before the first iteration of the algorithm. Since comparison can be considered as additions, we need altogether $N-1+\gamma \times \rho$ real additions per iteration.

Loop detection, as described in the previous section, requires only bit operations and integer counter increments/decrements. Most importantly, the total number of loop detection operations is independent of the block length N and depends only on the number of executed iterations. In practical applications, the user specified maximum iteration number is much less than the block length N. Moreover, as the decoding converges, the average number of iterations is far less than the user specified maximum iteration number. Therefore, the cost of loop detection in comparison with the $N-1+\gamma \times \rho$ real additions required by the core decoding procedure can be safely ignored. The decoding complexity associated with the proposed algorithms and some of the algorithms mentioned in Sections 3.1 and 3.2 are summarized in Table 3.1.

Table 3.1: Decoding Complexity per Iteration

Algorithm	Multiplications	Divisions	Additions
IDBP algorithm	$11N\gamma - 9N$	$N(\gamma+1)$	$N(3\gamma+1)$
Normalized BP-based	0	$N\gamma$	$4N(\gamma-1)+N\log_2 2\gamma/2$
UMP BP-based	0	0	$4N(\gamma-1)+N\log_2 2\gamma/2$
WBF	0	0	$N-1+\gamma \times \rho$
IWBF	0	0	$N-1+\gamma \times \rho$
Proposed algorithm II	0	0	$N-1+\gamma \times \rho$

3.4 Simulation Results

The error performance, in terms of bit-error rate (BER) as a function of the signal-to-noise ratio (SNR), of the algorithms presented in the previous section is compared with some existing algorithms for different types of LDPC codes. The different codes studied are given in Table 3.2. The following algorithms are simulated: MLG, BF (with adaptive threshold), WBF, IWBF, IDBP, and the algorithms proposed herein. The maximum number of iteration is set equal to 50 for IDBP, and 200 for the other algorithms.

3.4.1 Performance of Algorithm I

Figures 3.1 and 3.2 show the performances of the (2048, 1723) RS-LDPC code and the (2048, 1664) PEG-LDPC code based on different hard-decision decoding algorithms. It can be observed that, at the BER of 10^{-5}, Algorithm I outperforms BF algorithm by about 0.2 dB and 0.5 dB, respectively. It was observed through simulations of various LDPC codes that Algorithm I always outperforms BF algorithm, and that the performance gap between the two algorithms decreases with column weight of the code. For the Type-I 2-D (1023, 781) EG-LDPC code for example, the performance gap is only about 0.1 dB at the BER of 10^{-5}. As mentioned earlier, the criterion for the decision if whether or not a position is in error differs from Gallager's algorithm in that here syndrome weights are compared where the number of errors differ by two. It is necessary to point out that Algorithm I is slightly more complex than BF decoding.

3.4.2 Impact of SNR on the Optimal Value of α in Algorithm II

For a given LDPC code with given column weight, at a given SNR, the performances of Algorithm II vary with the value of the weighting factor α. The effect of α on the BER performances of the (2048, 1664) PEG-LDPC code with column weight 3, the (2048, 1664) PEG-LDPC code with column weight 6, and the (2048, 1723) RS-LDPC code with column weight 6 are shown in Figures 3.3 – 3.5 respectively. It can be seen that the optimal value of α decreases slowly as the SNR increases. It can also be observed that at low SNRs, the performance is not very sensitive to the value of α . The same observations were made for the different LDPC codes studied.

3.4.3 Impact of Column Weight on the Optimal Value of α in Algorithm II

At a given SNR, or for a given BER, the optimal value of α in Algorithm II for decoding different LDPC codes may vary with the corresponding column weights. The optimal values of α at various SNRs for decoding different LDPC codes are given in Table 3.3. It can be observed that the optimal value of α depends not only on the column weight, but also on the structure of the code. For example, at a given SNR, or at a given performance level (BER), the optimal α for the RS-LDPC codes decreases with the column weight, whereas it increases for the PEG-LDPC codes. On the other hand, the optimal α is the same for both the (2048, 1723) RS-LDPC code and the (2048, 1664) PEG-LDPC code. The two codes have a column weight 6.

3.4.4 Impact of the Value of α on the Average Number of Iterations in Algorithm II

Figure 3.6 depicts the effect of α on the average number of iterations for the decoding of the (2048, 1664) PEG-LDPC code with parameters $\gamma = 3,\ \rho = 16$. It can be seen that, for any value of the SNR, the average number of iterations is at its lowest when α is optimal. It can also be observed that at high SNRs, the average number of iterations is not very sensitive to the value of α . The same observations were made for the different LDPC codes studied.

3.4.5 Performance of Algorithm II With Optimal α

Simulation results of Algorithm II with optimal weighting factors (chosen as the optimal α at BER of 10^{-5}) for decoding various LDPC codes are shown in Figures 3.7 – 3.12. Also shown in these figures are the performances those codes based on different soft-decision decoding algorithms. It can be observed that, in all cases, Algorithm II with loop detection (LD) offers a gain of about 0.1 dB over the IWBF, and is always less than 0.6 dB away from IDBP. It can also be seen in Figures 3.7, 3.9, and 3.11 that, at a BER of 10^{-5}, the loop detection and prevention as described in the previous section improves the performance of Algorithm II by about 0.3 dB and 0.4 dB respectively. The same trends apply to other LDPC codes.

Table 3.2: Different LDPC codes simulated

Code type	Length (N)	Dimension (K)	Rate	Parameters
(EG)-LDPC code [22]	1023	781	0.763	$\gamma = 32,\ \rho = 32$
(RS)- LDPC code [28]	2048	1723	0.841	$\gamma = 6,\ \rho = 32$
	2048	1649	0.805	$\gamma = 8,\ \rho = 32$
	2048	1605	0.784	$\gamma = 10,\ \rho = 32$
	2048	1561	0.762	$\gamma = 12,\ \rho = 32$
PEG-LDPC code [50]	2048	1664	0.813	$\gamma = 3,\ \rho = 16$
	2048	1664	0.813	$\gamma = 6,\ \rho = 32$

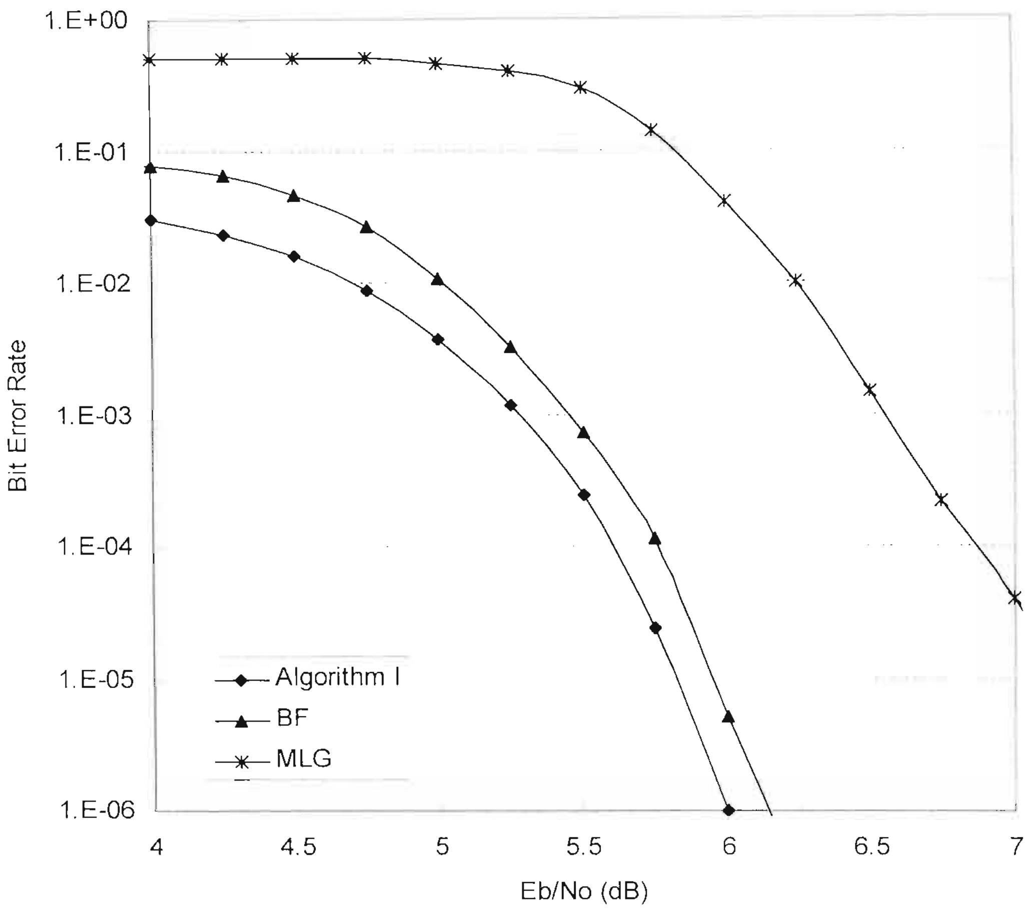

Figure 3.1: Bit-error-rate of the (2048, 1723) RS-LDPC code with parameters $\gamma = 6,\ \rho = 32$ based on different hard-decision decoding algorithms.

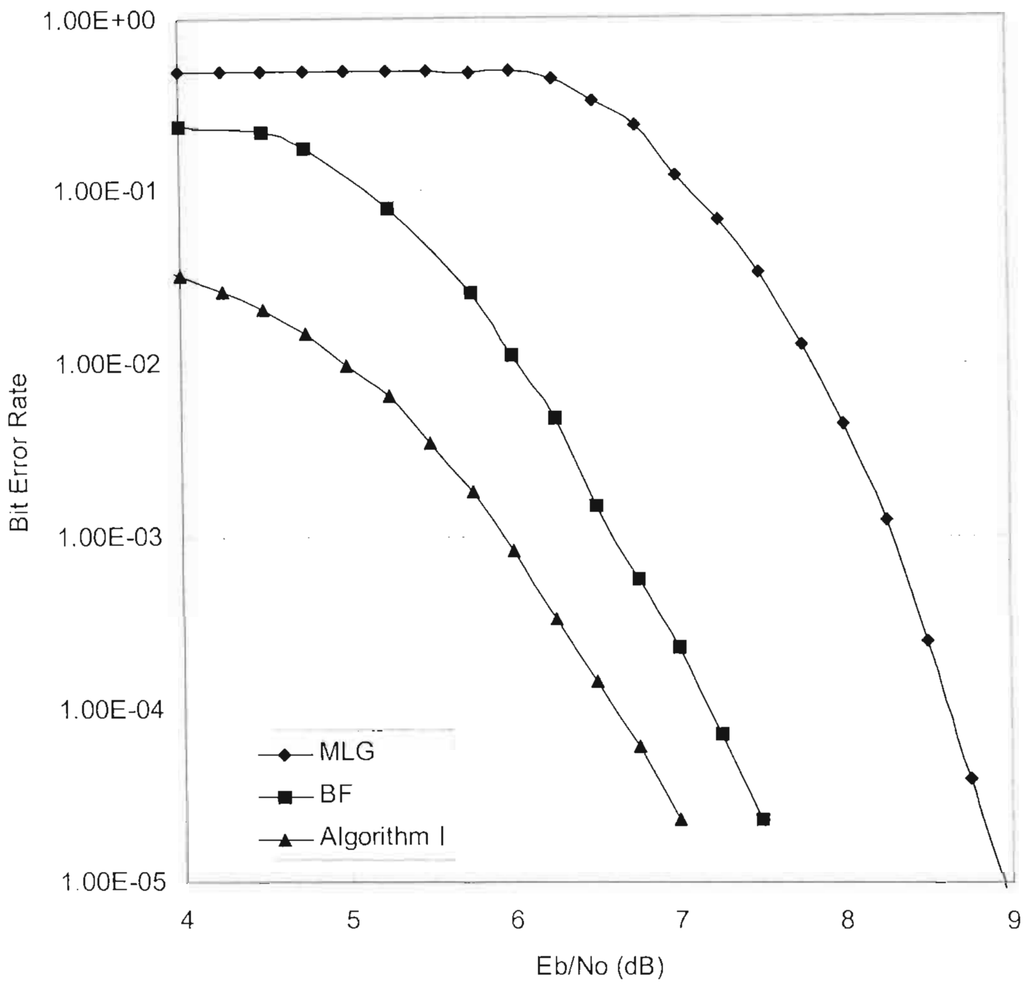

Figure 3.2: Bit-error-rate of the (2048, 1664) RS-LDPC code with parameters $\gamma = 3$, $\rho = 16$ based on different hard-decision decoding algorithms.

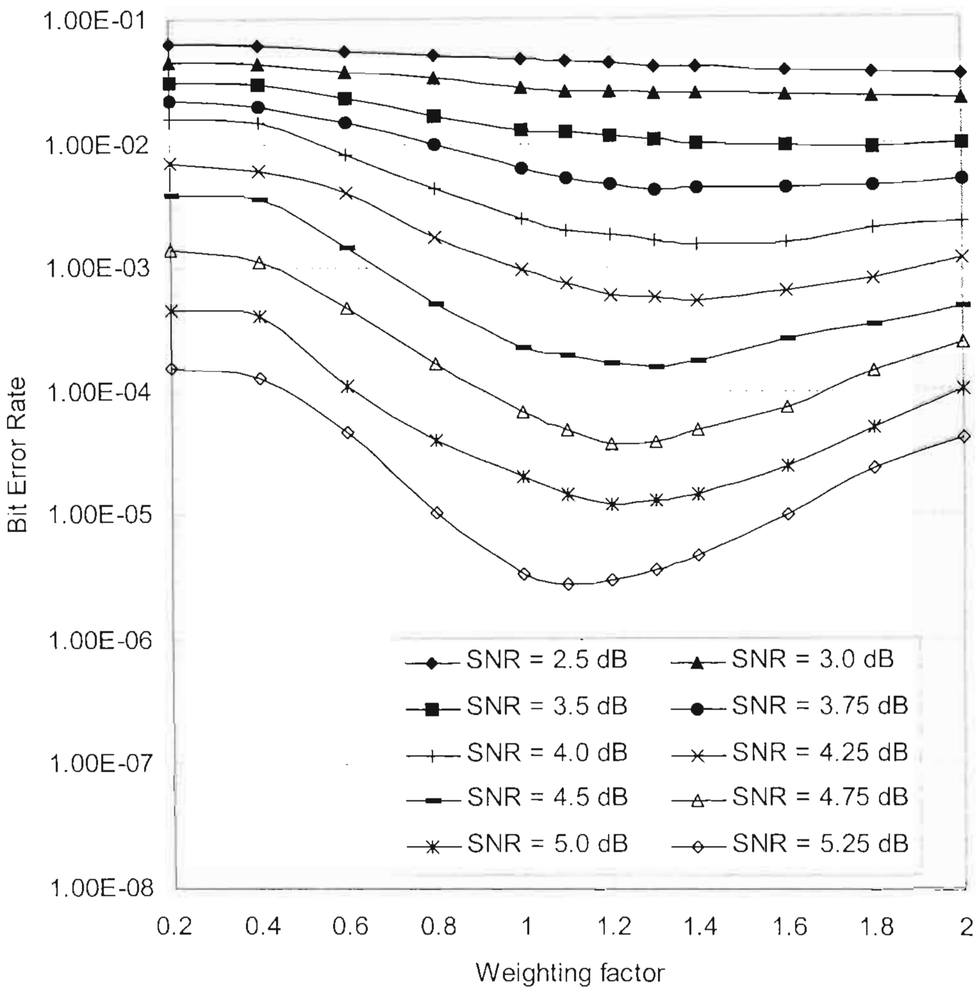

Figure 3.3: Impact of the weighting factor α in Algorithm II on the BER performance for the decoding of the (2048, 1664) PEG-LDPC code with parameters $\gamma = 3$, $\rho = 16$.

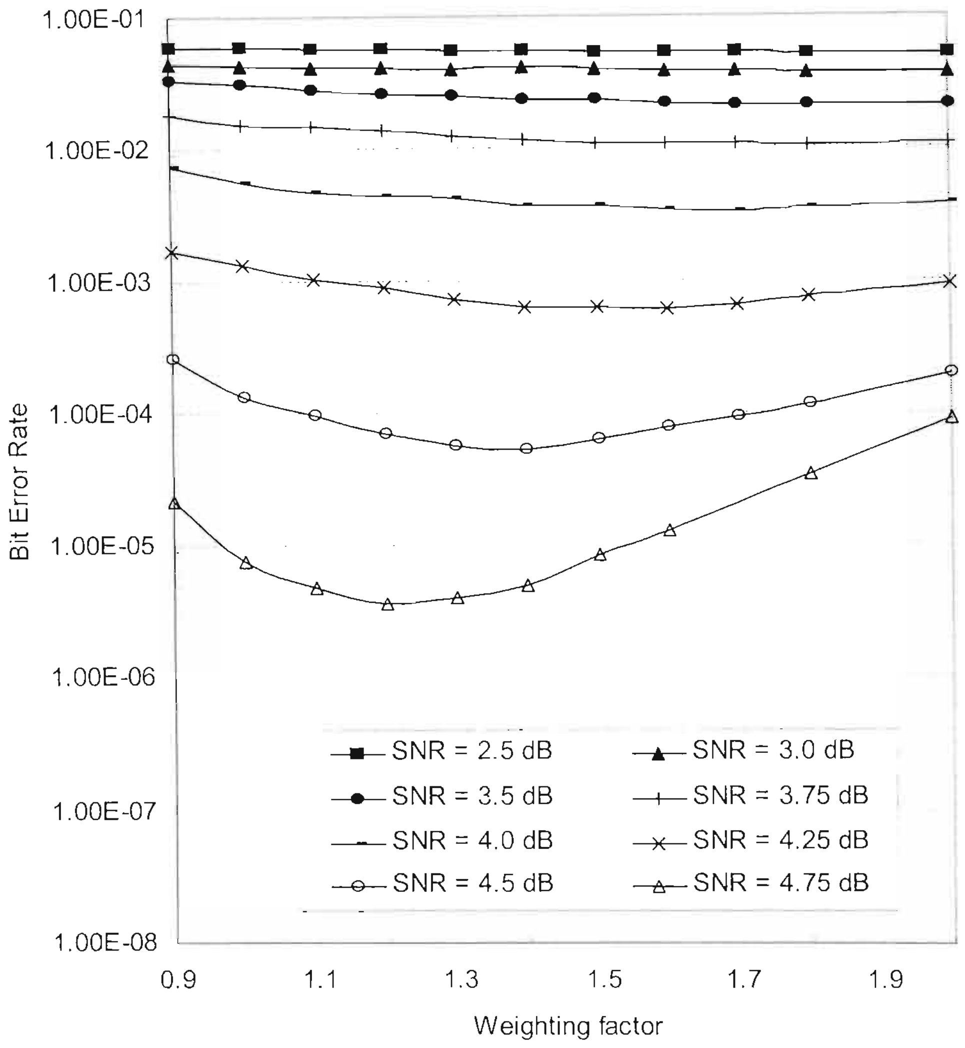

Figure 3.4: Impact of the weighting factor α in Algorithm II on the BER performance for the decoding of the (2048, 1664) PEG-LDPC code with parameters $\gamma = 6$, $\rho = 32$.

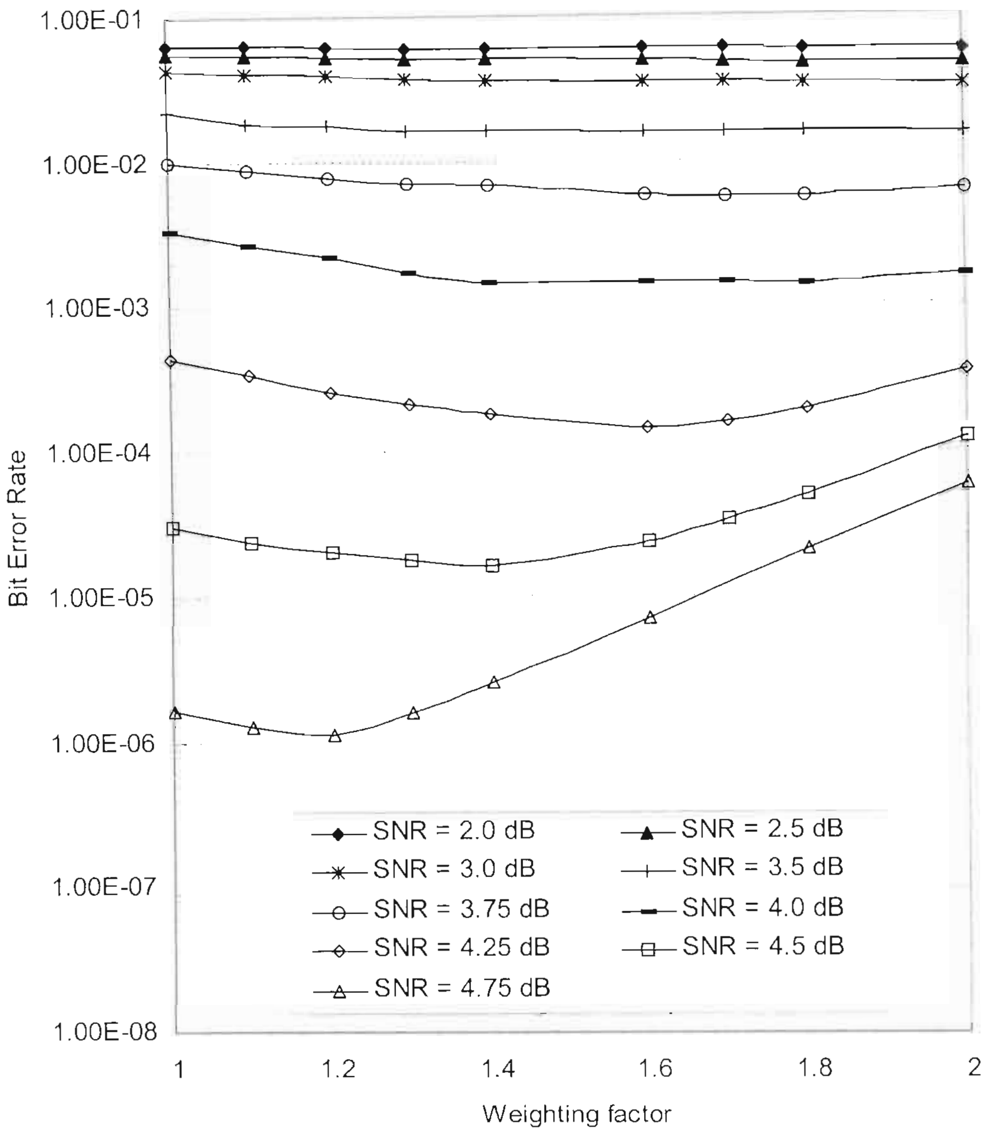

Figure 3.5: Impact of the weighting factor α in Algorithm II on the BER performance for the decoding of the (2048, 1723) RS-LDPC code with parameters $\gamma = 6$, $\rho = 32$.

Table 3.3: Impact of Column Weight on the Optimal α in Algorithm II for Different LDPC Codes.

Code type	(N, K)	w_c	SNR (dB)	BER	α_{opt}
(EG)-LDPC code [11]	(1023, 781)	32	3.5	$1.9\text{x}10^{-3}$	1.1
			3.75	$2.3\text{x}10^{-4}$	1.0
			4.0	$1.4\text{x}10^{-5}$	0.9
			4.125	$3.0\text{x}10^{-6}$	0.9
(RS)- LDPC code [22]	(2048, 1723)	6	4.0	$1.3\text{x}10^{-3}$	1.8
			4.25	$1.4\text{x}10^{-4}$	1.6
			4.5	$1.6\text{x}10^{-5}$	1.4
			4.75	$1.1\text{x}10^{-6}$	1.2
	(2048, 1649)	8	3.75	$4.1\text{x}10^{-3}$	1.5
			4.0	$7.1\text{x}10^{-4}$	1.5
			4.25	$5.6\text{x}10^{-5}$	1.3
			4.5	$3.9\text{x}10^{-6}$	1.2
	(2048, 1605)	10	3.75	$3.3\text{x}10^{-3}$	1.5
			4.0	$5.3\text{x}10^{-4}$	1.4
			4.25	$2.9\text{x}10^{-5}$	1.2
			4.5	$4.4\text{x}10^{-7}$	1.2
	(2048, 1561)	12	3.75	$3.2\text{x}10^{-3}$	1.6
			4.0	$4.0\text{x}10^{-4}$	1.4
			4.25	$2.8\text{x}10^{-5}$	1.2
			4.5	$1.3\text{x}10^{-6}$	1.2
PEG-LDPC code [23]	(2048, 1664)	3	4.0	$1.5\text{x}10^{-3}$	1.4
			4.5	$1.5\text{x}10^{-4}$	1.3
			5.0	$1.2\text{x}10^{-5}$	1.2
			5.25	$2.8\text{x}10^{-6}$	1.1
	(2048, 1664)	6	4.0	$3.2\text{x}10^{-3}$	1.7
			4.25	$6.0\text{x}10^{-4}$	1.6
			4.5	$5.2\text{x}10^{-5}$	1.4
			4.75	$4.0\text{x}10^{-6}$	1.2

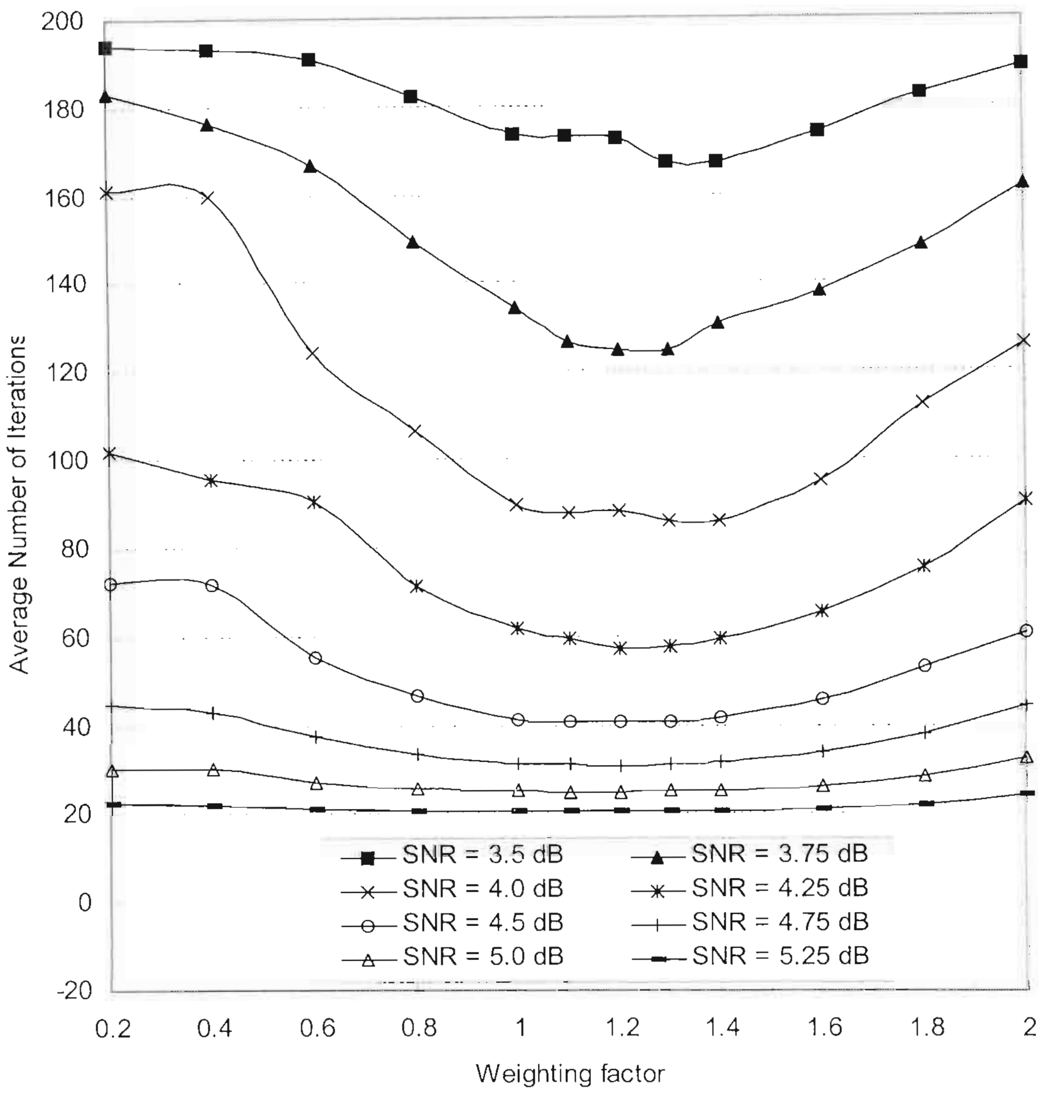

Figure 3.6: Impact of the weighting factor α in Algorithm II on the average number of iterations of the (2048, 1664) PEG-LDPC code with parameters $\gamma = 3,\ \rho = 16$.

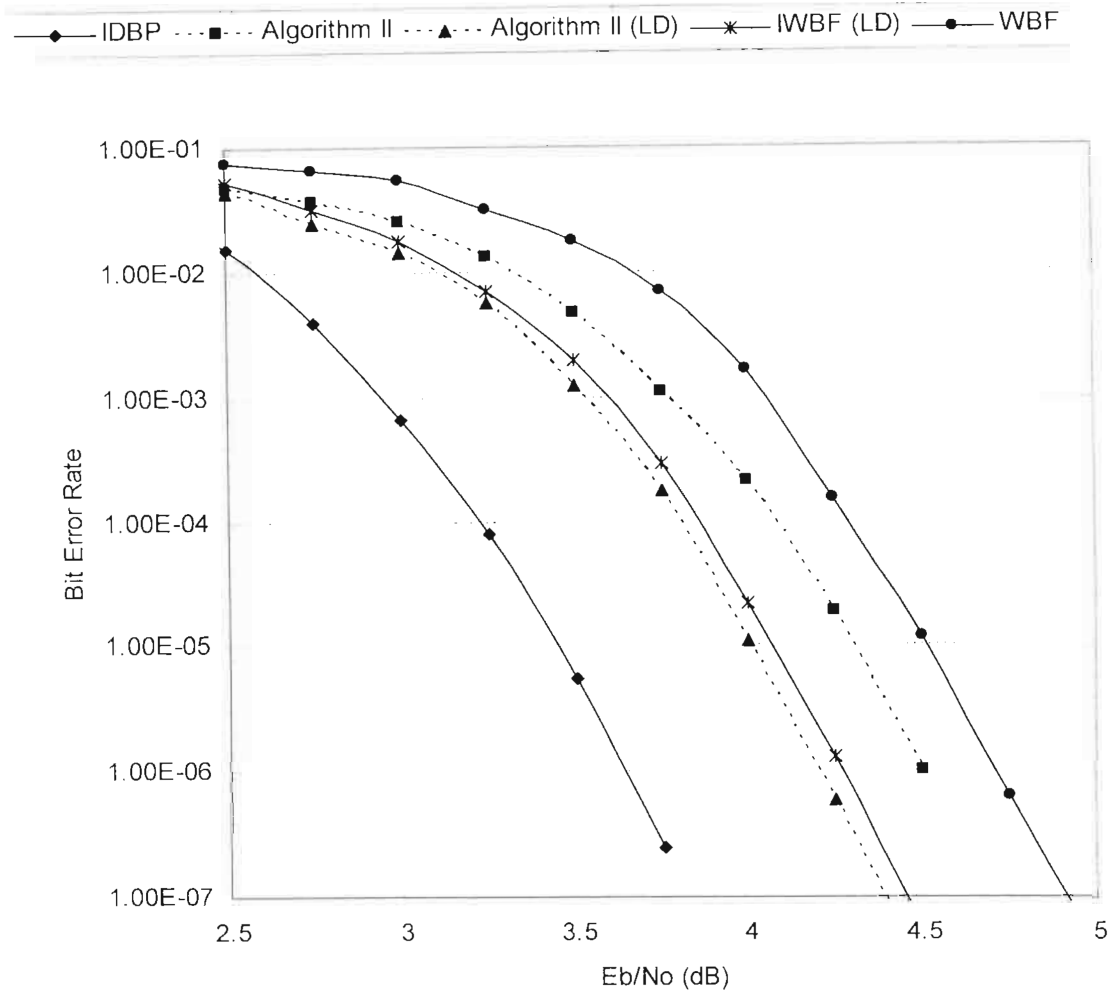

Figure 3.7: Bit-error-rate of the Type-I 2-D (1023, 781) EG-LDPC code with parameters $\gamma = 32$, $\rho = 32$ based on different soft-decision decoding algorithms.

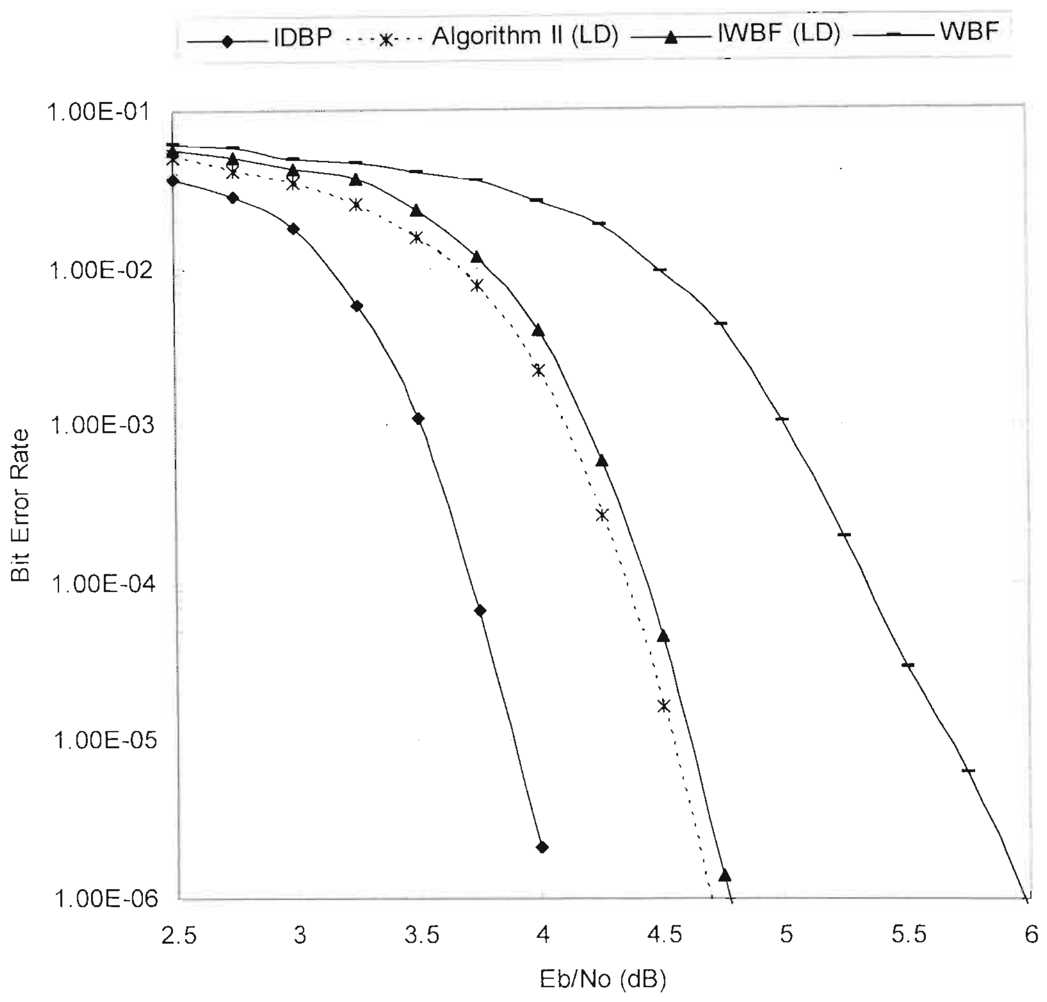

Figure 3.8: Bit-error-rate of the (2048, 1723) RS-LDPC code with parameters $\gamma = 6,\ \rho = 32$ based on different soft-decision decoding algorithms.

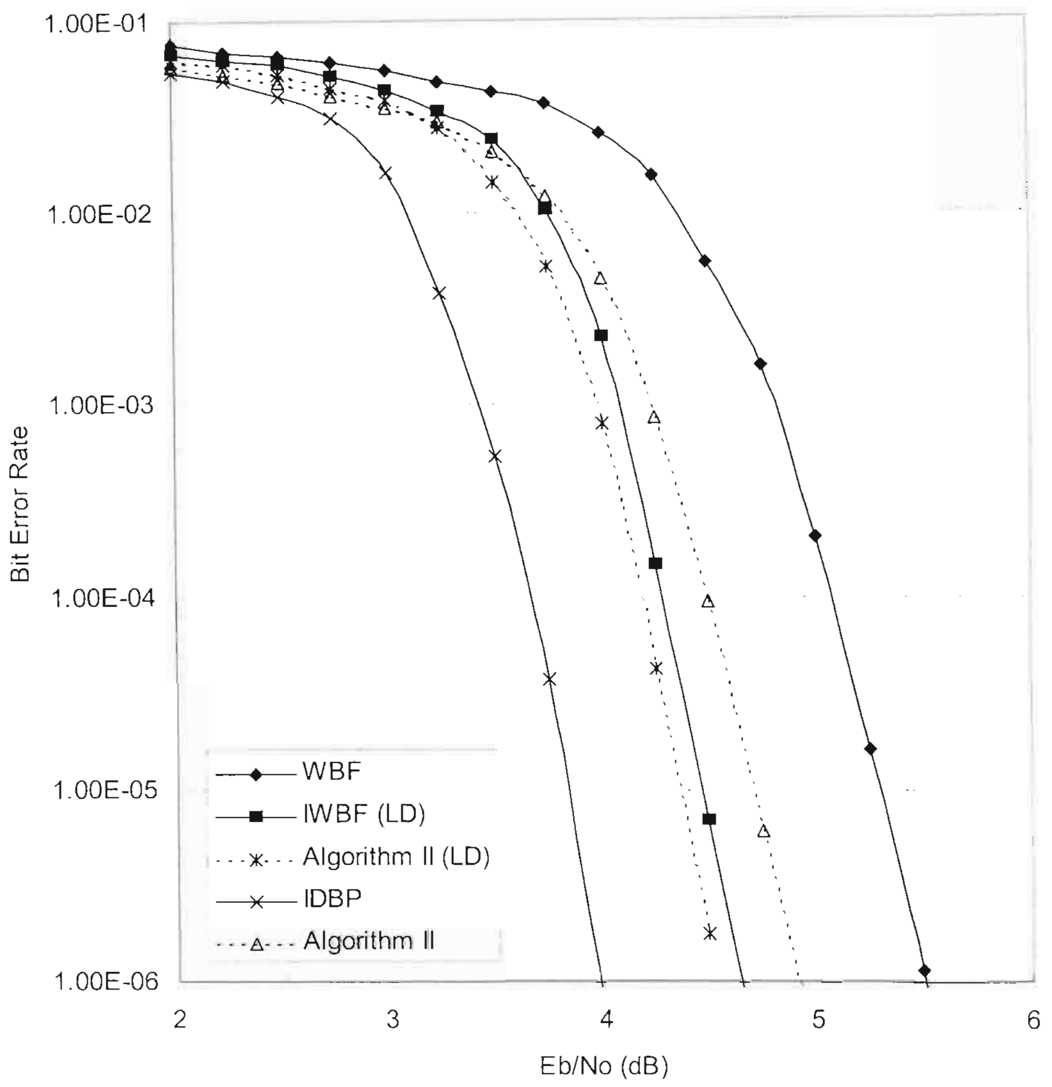

Figure 3.9: Bit-error-rate of the (2048, 1649) RS-LDPC code with parameters $\gamma = 8$, $\rho = 32$ based on different soft-decision decoding algorithms.

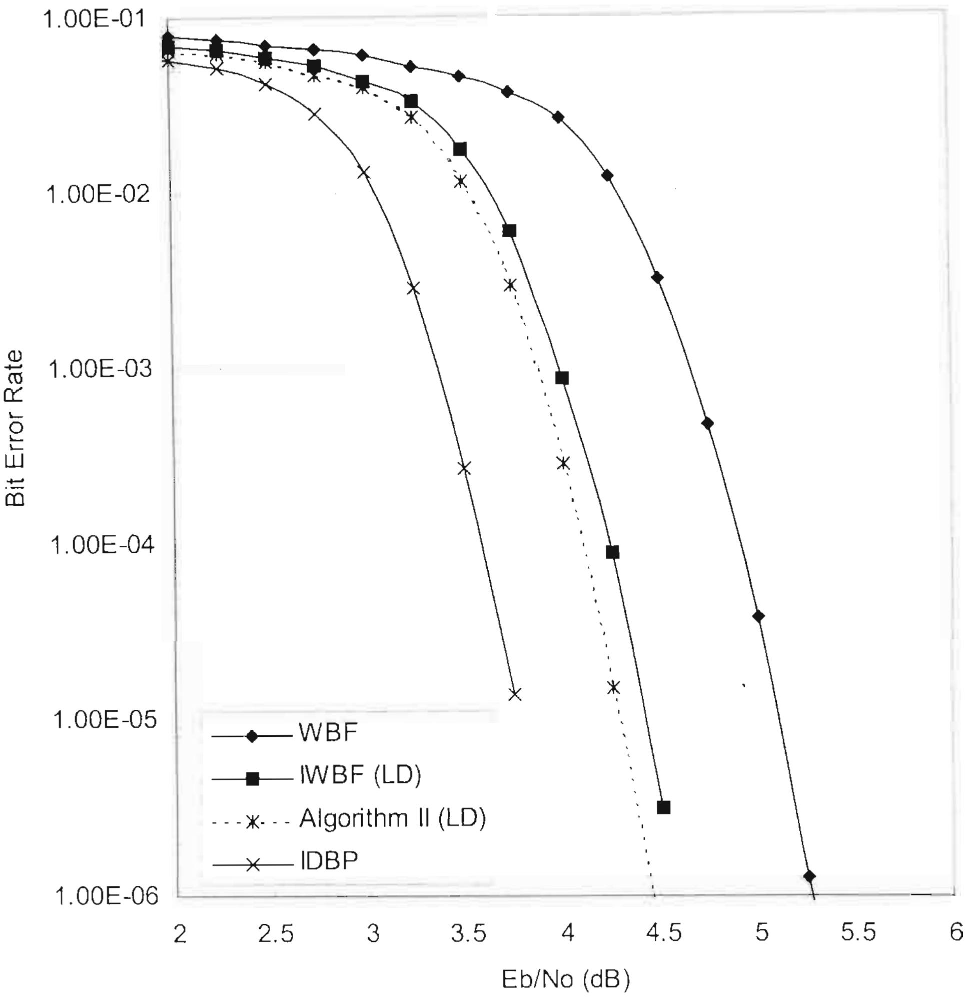

Figure 3.10: Bit-error-rate of the (2048, 1605) RS-LDPC code with parameters $\gamma = 10$, $\rho = 32$ based on different soft-decision decoding algorithms.

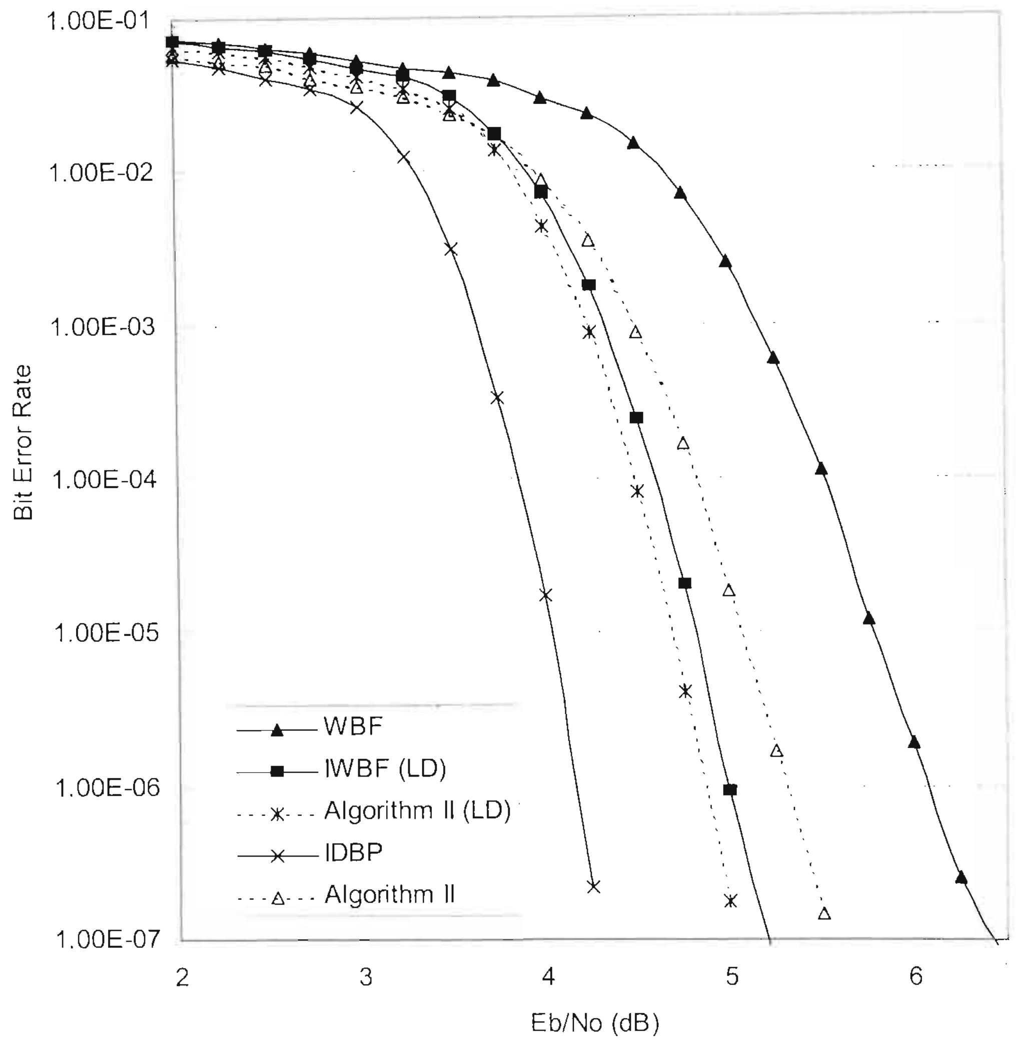

Figure 3.11: Bit-error-rate of the (2048, 1664) PEG-LDPC code with parameters $\gamma = 6,\ \rho = 32$ based on different soft-decision decoding algorithms.

3.5 Conclusions

In this Chapter, two new decoding algorithms for LDPC codes are presented. The first algorithm is a hard-decision method, and the second one is a modification of the first to include reliability information of the received symbols. In principle and in complexity, the algorithms belong to the class of bit flipping algorithms. The defining attribute of the proposed algorithms is the bit selection criterion which is based on the fact that, for low-density matrices, the syndrome weight increases with the number of errors in average until error weights much larger than half the minimum distance. A loop detection procedure with minimal computational overhead is also proposed that protects the decoding from falling into infinite loop traps. Simulation results show that the proposed algorithms offer an appealing performance/cost trade-offs and may deserve a place in an LDPC decoding "toolbox". One drawback of the soft version of the algorithm is that the optimal weighting factor α is SNR-dependent and code-dependent. However, choosing the optimal α at BER 10^{-5} and keeping it constant at any SNR does not cause significant degradation in performance. Therefore, the optimal weighting factor α can be considered to be only code-dependent

Some modifications can easily be applied to the decoding algorithms to increase the decoding speed. For example, one can flip more than one bit in each iteration. As another example, the computation of the syndromes in Step 4) can be done only for the bit positions whose reliabilities are below a certain threshold.

CHAPTER 4

GENERALAZATION OF IRREGULAR LOW-DENSITY PARITY-CHECK CODES

In this chapter, a new class of codes called *generalized irregular low density* (GILD) codes is presented. This family of pseudo-random error correcting codes is built as the intersection of randomly permuted binary codes. It is a direct generalization of irregular LDPC codes (see Chapter 2), and is adapted from the previously known class of *generalized low density* (GLD) codes introduced independently by Lentmaier et *al.*[55], and Boutros et *al.* [56] [57].

It is proved by an ensemble performance argument that these codes exist and are asymptotically good in the sense of the minimum distance criterion, *i.e.* the minimum distance grows linearly with the block length. Upper and lower bounds on their minimum Hamming distance are provided, together with their maximum likelihood decoding error probability.

Two iterative soft-input soft-output (SISO) decoding for any GILD code are presented, and iterative decoding of GILD codes for communication over an AWGN channel with binary antipodal modulation (BPSK) is studied. The results are compared in terms of performance and complexity with those of GLD codes. The high flexibility in selecting the parameters of GILD codes and their better performance and higher rate make them more attractive than GLD codes and hence suitable for small and large block length forward error correcting schemes. Comparison between simulation results of a GILD code and the best LDPC code of length 1008 and rate 0.5 shows very close performances, suggesting that variations of GILD codes may be able to match or beat LDPC codes for small block lengths.

This chapter is organised as follows: in the next section an introduction is presented. The construction of GILD codes is described in Section 4.2. The ensemble performance of these codes is studied in Section 4.3. It is shown that GILD codes are asymptotically good and lower bound on their minimum Hamming distance is provided. In Section 4.4, average upper bounds on the minimum Hamming distance of GILD codes of fixed length and constituent code are calculated semi-analytically. In Section 4.5, average bounds for the bit error probability when maximum likelihood decoding is used are presented. The decoding algorithms are presented in Section 4.6. Simulation results are presented in Section 4.7 and, finally, conclusions are drawn in Section 4.8.

4.1 Introduction

As a direct generalization of Gallager's LDPC codes, *Generalized Low-Density (GLD) Parity-Check* codes were recently introduced by Lentmaier [55] and Boutros [56], independently. GLD codes are constructed by replacing each single parity check in Gallager's LDPC codes with the parity check matrix of a small linear block code called the *constituent code*. It has been shown that GLD codes are asymptotically good in the sense of minimum distance and exhibit an excellent performance over both AWGN and Rayleigh channels [55], [56], [66]. Moreover, Pothier [66] demonstrates that GLD codes can be considered as a generalization of product codes, and because of their higher flexibility on the selection of code length, GLD codes turn out to be a promising alternative to product codes in many applications like digital audio and TV broadcasting, high speed packet data transmission and deep space applications.

As originally suggested by Tanner [8], LDPC codes are well represented by bipartite graphs in which one set of nodes, the *variable nodes*, corresponds to elements of the codeword and the other set of nodes, the *check nodes*, corresponds to the set of parity-check constraints which define the code. *Regular* LDPC codes are those for which all nodes of the same type have the same degree. This means that the parity-check matrix of the code contains the same number of ones in each row and the same number of ones in each column. *Irregular* LDPC codes, introduced in [61], [67], and further studied in [16], [17], [18], [19], [20], were demonstrated in [18] to substantially outperform similar codes based on regular graphs. For such an irregular LDPC code, the degrees of the nodes on each side of the graph can vary widely. In terms of the parity-check matrix, the weight per row and column is not uniform, but instead is governed by an appropriately chosen distribution of weights. Actually, GLD codes can be considered as a generalization of regular LDPC codes.

Inspired by these recent results showing that irregular structure improves performance, the generalization of irregular LDPC codes is introduced, where small linear block codes are used as component codes instead of single-error detecting parity-check codes.

4.2 Definition and Construction of GILD Codes

It has been shown that binary GLD codes with only $J = 2$ levels are asymptotically good [55], [56], [66] (A more detailed description of GLD codes can be found in those references). Furthermore, GLD codes with only 2 levels have the highest code rate and simple decoder structure. Thus in this work, only GILD codes with two levels are considered. However, the construction method presented here can be easily applied to GILD codes with more than two levels.

4.2.1 Definition

A binary GILD code with two levels is defined by three parameters:

- The length of the code N.
- The length n of the component code C_0, with dimension k and rate r.
- The fraction, θ ($\theta < 1$), of the code symbols connected to the second super-code.

It should be noted that both N and θN must be multiples of n.

4.2.2 Construction

To construct the parity-check matrix $\boldsymbol{H}$ of such a code, a low-density (LD) matrix with $\frac{N}{n}(1+\theta)$ rows and N columns, having exactly θN columns with 2 ones, $N(1-\theta)$

columns with 1 one, n ones in each row, and zeros elsewhere is first constructed. Then, in each row of this matrix, the N-n zero elements are replaced by a zero column-vector of length (n-k), and the nonzero elements by one of the n different columns of the component code's parity-check matrix, each used once. It is important to note that if Hamming codes or BCH codes are used as constituent codes, the parity-check matrix of the resulting GILD code is not low density and may contain many short cycles. This is because the conventional parity-check matrices of Hamming codes and BCH codes are not low-density and their Tanner graphs usually contain many short cycles.

The construction of the LD matrix can easily be achieved using a technique similar to that of [2]. This matrix is divided into 2 submatrices. The first of these submatrices contains all its 1's in descending order; that is, the i^{th} row contains 1's in columns $(i-1)k+1$ to ik. The second submatrix is not just merely a permutation of the first, but a column permutation of the first over a selected number of rows. The permutation is performed such that no two parity-check sets would contain more than one digit in common, i.e. the resulting LD matrix is free of cycle of length 4 in its Tanner graph. This, in effect, ensures that two constituent codes in the resulting GILD code have not more than one digit in common. Figure 4.1 shows an example of a LD matrix for a GILD code of length N = 20, n = 4, and θ = 0.8. Note that, unlike the interleaver in regular LDPC codes and GLD codes, π here is an "irregular" interleaver acting only on a selected number of rows of the first submatrix, 4 out 5 in this example.

The parity-check matrix $\boldsymbol{H}$ of the GILD code with two levels can be divided into two submatrices, $\boldsymbol{H}^1$ and $\boldsymbol{H}^2$. $\boldsymbol{H}^1$ is a *block diagonal* matrix and produces the direct sum of N/n constituent codes, where N is the GILD code length. The second submatrix is constructed as: $\boldsymbol{H}^2 = \pi(\boldsymbol{H}^1)$, where π represents a random column permutation over $(\theta N/n)$ rows of $\boldsymbol{H}^1$. A (N, n, θ) GILD code C can be considered as the intersection of 2 *super-codes* C^1 and C^2, whose parity check matrices are the two submatrices, $\boldsymbol{H}^1$ and $\boldsymbol{H}^2$, respectively. If the parity-check matrix $\boldsymbol{H}$ has full rank, the total rate of the GILD code is:

$$R = (1+\theta)r - \theta. \tag{4.1}$$

$$\begin{array}{c|cccccccccccccccccccc}
 & \multicolumn{20}{c}{\xleftrightarrow{\quad N = 20 \quad}} \\
 & \multicolumn{4}{c}{\xleftrightarrow{n = 4}} & & & & & & & & & & & & & & & & \\
 & 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
 & 0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\
 & 0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0 \\
 & 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0 \\
\pi\downarrow & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \\
\hline
 & 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0 \\
 & 0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&1&0&0&0 \\
 & 0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0 \\
 & 0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1
\end{array}$$

Figure 4.1: Example of a low-density matrix of a GILD code for $N = 20$, $n = 4$, and $\theta = 0.8$.

In case of smaller rank, the rate is increased accordingly. However, with increasing block length N, (4.1) gives a good approximation of the actual rate of the code. Note that the rate of the corresponding GLD code would be:

$$R' = 2r - 1. \tag{4.2}$$

Clearly $R > R'$ since $\theta < 1$. This is the first advantage of GILD codes over GLD codes. Flexibility in defining the code rate to suit particular applications can be achieved by changing the value of θ. It should be noted that if $\theta = 1$, we obtain a GLD code.

4.3 Ensemble Performance

In this section, the average weight distribution of GILD codes over all possible interleavers as a function of the component code C_0, and the length N is studied. From this distribution, an asymptotical upper bound that allows us to prove that GILD codes are asymptotically good is deduced and their minimum distance is compared with the Gilbert-Varshamov (GV) bound. The maximum transition probability of the BSC channel that an asymptotical GILD code with fixed rate can achieve with maximum likelihood decoding is also derived and is compared to the BSC capacity[7].

4.3.1 Average Weight Distribution of GILD Codes

The direct computation of the exact weight distribution of GILD code becomes rapidly intractable when N increases. However, the weight distributions of the constituent codes used in practical GILD codes (Hamming codes, extended Hamming codes, BCH codes with error correction capacity $t = 2$ and extended BCH codes) are well known. See [23] (p. 142) for Hamming and extended Hamming codes, [68] (p. 451 & p.669), and [69] (Ch. 16) for the family of BCH codes. The average weight coefficient of a GILD code can be easily obtained by averaging over all the possible interleavers π.

Let us denote by $g(s)$ the moment generating function of the component code, which is the exponential polynomial whose coefficient g_i of degree i is the normalized number of codewords with weight i. For example, the moment generating function of the (7, 4, 3) Hamming code is:

$$g(s) = \frac{1 + 7e^{3s} + 7e^{4s} + e^{7s}}{16}. \tag{4.3}$$

The first super-code, C^1, is the direct sum of N/n independent constituent codes C_0. Hence, its moment generating function $G_1(s)$ is simply a power of $g(s)$:

$$G_1(s) = g(s)^{N/n} = \sum_l Q(l) e^{ls}, \tag{4.4}$$

where $Q(l)$ is the probability that a codeword of C^1 has weight l. Since the total number of codewords in C^1 is $(2^k)^{N/n}$, the number of codewords of C^1 having weight l is:

$$N_1(l) = 2^{(kN/n)} Q(l). \tag{4.5}$$

[7] The analyses in this section are similar to those in [56] concerning GLD codes.

The second super-code, C^2, however, is the sum of $\theta N/n$ independent constituent codes C_0 and a code of length $N(1-\theta)$ generated by the all zero matrix. Thus its moment generating function is:

$$G_2(s) = g(s)^{\theta N/n} 2^{N(\theta-1)} \left(1+e^s\right)^{N(1-\theta)} = \sum_l K(l) e^{ls}, \tag{4.6}$$

where $K(l)$ is the probability that a codeword in C^2 has a weight l. Since the total number of codewords in C^2 is $\left(2^k\right)^{(\theta N/n)} 2^{N(1-\theta)}$, the number of codewords of C^2 having weight l is:

$$N_2(l) = 2^{N(\theta k/n+(1-\theta))} K(l). \tag{4.7}$$

Due to the fact that C^1 and C^2 are totally independent, the probability, $P(l)$, that a vector of weight l belongs to $C = C^1 \bigcap C^2$ is the product of the probabilities that it belongs to each super-code, that is:

$$P(l) = \frac{N_1(l)}{\binom{N}{l}} \cdot \frac{N_2(l)}{\binom{N}{l}}. \tag{4.8}$$

Finally, the average number of codewords in C having weight l is:

$$\overline{N(l)} = \binom{N}{l} \times P(l) = \frac{2^{N\left[(1+\theta)\frac{k}{n}+(1-\theta)\right]} Q(l) K(l)}{\binom{N}{l}}. \tag{4.9}$$

This formula is the starting point of different analytical or semi-analytical results presented in the following sections.

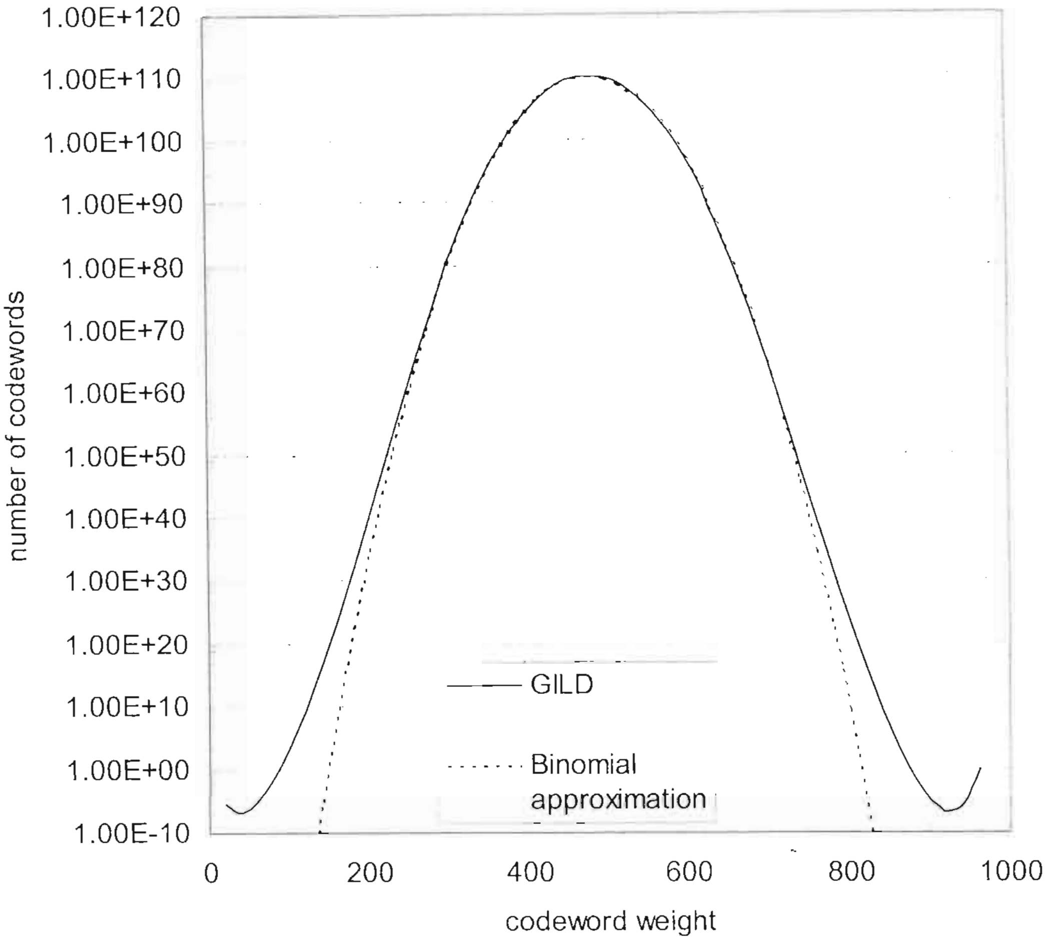

Figure 4.2: Average weight distribution of a GILD code of length $N = 961$, $\theta = 0.903$ based on the (31, 21, 5) BCH code, compared with the binomial approximation.

4.3.2 Lower Bound on the Asymptotic Minimum Distance

The following theorem gives an upper bound for the average number of codewords of weight l, $\overline{N(l)}$.

Theorem 4.1: Let C be a binary GILD code of length N built from a linear (n,k) block code C_0. Let $\overline{N(l)}$ be the average number of codewords of weight l in C averaged over all the possible interleavers π. Then for $l > 0$, $\overline{N(l)}$ is upper bounded by:

$$\overline{N(l)} \leq C(\lambda, N) \times e^{-NB(\lambda, s)}, \tag{4.10}$$

where $\lambda = l/N$ is the normalized weight. The two functions $C(\lambda, N)$ and $B(\lambda, s)$ are expressed as follows:

$$C(\lambda, N) = \sqrt{2\pi N\lambda(1-\lambda)}\exp\left(\frac{1}{12N\lambda(1-\lambda)}\right), \tag{4.11}$$

and

$$B(\lambda, s) = H(\lambda) + 2\lambda s - \frac{(1+\theta)}{n}\left(\mu(s) + k\log 2\right) - (1-\theta)\log\left(1+e^{s}\right), \tag{4.12}$$

where $H(\lambda)$ is the natural entropy function and $\mu(s) = \log(g(s))$.

Proof: By using exactly the same bounding technique as Gallager [1, pp. 13-17], *i.e.* upper bounding roughly each coefficient $Q(l)$ in (4.4) by the entire function $G_1(s)$ and each coefficient $K(l)$ in (4.6) by the entire function $G_2(s)$, we obtain:

$$Q(l) \le G_1(s)e^{-ls}, \tag{4.13}$$

and

$$K(l) \le G_2(s)e^{-ls}. \tag{4.14}$$

Defining $\mu(s) = \ln(g(s))$ and the normalized weight $\lambda = l/N$, (4.13) and (4.14) can be rewritten respectively as:

$$Q(l) \le \exp\left\{-N\left[\lambda s - \frac{\mu(s)}{n}\right]\right\}, \tag{4.15}$$

and

$$K(l) \le \exp\left\{-N\left[\lambda s - \frac{\theta\mu(s)}{n} + (1-\theta)\left(\ln 2 - \ln\left(1+e^{s}\right)\right)\right]\right\}. \tag{4.16}$$

Let us now lower-bound the denominator in (4.9), which is the binomial coefficient. Extended Stirling bounds on $z!$ are [70]:

$$\sqrt{2\pi.z}z^{z}e^{-z} \le z! \le \sqrt{2\pi.z}z^{z}e^{-z}\exp\left(\frac{1}{12z}\right), \tag{4.17}$$

and give for the binomial coefficient:

$$\frac{\exp\left\{NH(\lambda) - \dfrac{1}{12N(1-\lambda)}\right\}}{\sqrt{2\pi N\lambda(1-\lambda)}} \le \binom{N}{l} = \binom{N}{\lambda N} \le \frac{\exp\{NH(\lambda)\}}{\sqrt{2\pi N\lambda(1-\lambda)}}, \tag{4.18}$$

where $H(\lambda)$ is the natural entropy function:

$$H(\lambda) = -\lambda \ln(\lambda) - (1-\lambda)\ln(1-\lambda). \qquad (4.19)$$

Introducing (4.14), (4.15), and (4.18) gives:

$$\begin{aligned} \overline{N(l)} \leq \sqrt{2\pi N\lambda(1-\lambda)}\exp\left\{\frac{1}{12N\lambda(1-\lambda)}\right\} \\ \times\exp\left\{-N\left[H(\lambda)+2\lambda s-\frac{(1+\theta)}{n}\left(\mu(s)+k\ln 2\right)-(1-\theta)\ln\left(1+e^{s}\right)\right]\right\} \end{aligned}. \qquad (4.20)$$

We define:

$$C(\lambda, N) = \sqrt{2\pi N\lambda(1-\lambda)}\exp\left(\frac{1}{12N\lambda(1-\lambda)}\right), \qquad (4.21)$$

and

$$B(\lambda, s) = H(\lambda) + 2\lambda s - \frac{(1+\theta)}{n}\left(\mu(s)+k\log 2\right) - (1-\theta)\log\left(1+e^{s}\right). \qquad (4.22)$$

Asymptotically, when the length N tends to infinity, $\overline{N(l)}$ tends to zero if $B(\lambda, s)$ is strictly positive. For a given value of θ, our aim is now to find (if it exists) the largest value δ of λ satisfying $B(\delta, s) = 0$ and $B(\lambda, s) < 0$ for $\lambda > \delta$. This value of δ would give us an asymptotical lower bound on the normalized minimum distance of the GILD code.

$B(\lambda, s)$ is a function of an arbitrary parameter s we have to optimize in order to get the tightest bound, namely we want $B(\lambda, s)$ as large as possible for each λ. Let us denote by $f_\lambda(s)$ the part of $B(\lambda, s)$ that depends on s:

$$f_\lambda(s) = 2\lambda s - \frac{(1+\theta)}{n}\mu(s) - (1-\theta)\ln(1+e^{s}). \qquad (4.23)$$

Finding the value of s that maximizes $B(\lambda, s)$ is equivalent to finding the value that maximizes $f_\lambda(s)$. When we set the derivatives of $f_\lambda(s)$ equal to zero, we get:

$$\lambda_{opt} = \frac{(1+\theta)}{2n}\mu'(s) + \frac{(1-\theta)}{2}\frac{e^{s}}{1+e^{s}}. \qquad (4.24)$$

We do not try to invert this relation. A large range of values of s will be covered, and (4.24) will give us the value λ_{opt} of λ for which $B(\lambda, s)$ is optimal. Before doing this, we have to make sure that (4.24) is one-to-one in order to cover the whole range of values assumed by λ between 0 and 1/2 and that $f_\lambda(s)$ is a convex $\cap$ function of s.

As $g(s)$ is the moment generating function of the constituent code, it can be written as:

$$g(s) = \sum_{i=0}^{n} g_i e^{is} = \frac{1}{2^k} \sum_{i=0}^{n} w_i e^{is} , \tag{4.25}$$

where w_i is the number of codewords of C_0 of weight i. We have:

$$\mu'(s) = \frac{g'(s)}{g(s)} = \frac{\sum_{i=0}^{n} i g_i e^{is}}{\sum_{i=0}^{n} g_i e^{is}} , \tag{4.26}$$

and

$$\mu''(s) = \frac{\left(\sum_{i=0}^{n} i^2 g_i e^{is}\right)\left(\sum_{i=0}^{n} g_i e^{is}\right) - \left(\sum_{i=0}^{n} i g_i e^{is}\right)^2}{\left(\sum_{i=0}^{n} g_i e^{is}\right)^2}$$

$$= \frac{\sum_{l=0}^{2n} u_l e^{ls}}{\left(\sum_{i=0}^{n} g_i e^{is}\right)^2} , \tag{4.27}$$

where, for each $l \in [0, \cdots, 2n]$, u_l equals:

$$\begin{aligned} u_l &= \sum_{j=0}^{l} j^2 g_j g_{l-j} - \sum_{j=0}^{l} j(l-j) g_j g_{l-j} \\ &= \sum_{j=0}^{l} g_j g_{l-j} j(2j-l) \\ &= \sum_{j=0}^{\lfloor \frac{l}{2} \rfloor} g_j g_{l-j} j(2j-l) + (l-j)(l-2j) \\ &= \sum_{j=0}^{\lfloor \frac{l}{2} \rfloor} g_j g_{l-j} (2j-l)^2 . \end{aligned} \tag{4.28}$$

This proves that each u_l is non-negative, and so is $\mu''(s)$ for all s. Thus, λ_{opt} as defined by (4.24) is a monotonic increasing function of s. We have:

$$\lambda_{opt} = \frac{(1+\theta)}{2n}\mu'(s) + \frac{(1-\theta)}{2}\frac{e^s}{1+e^s}$$

$$= \frac{(1+\theta)}{2n}\frac{\sum_{i=1}^{n} ig_i e^{is}}{\sum_{i=1}^{n} ig_i e^{is}} + \frac{(1-\theta)}{2}\frac{e^s}{1+e^s},$$

$$\lim_{s\to-\infty} \lambda_{opt} = 0, \tag{4.29}$$

$$\lim_{s\to+\infty} \lambda_{opt} = 1. \tag{4.30}$$

Figure 4.3 shows $\lambda_{opt}(s)$ for a specific constituent code.

Let us now show that $f_\lambda(s)$ is a convex $\cap$ function of s. We have:

$$f_\lambda''(s) = -\frac{(1+\theta)}{n}\mu''(s) - (1-\theta)\frac{e^s}{(1+e^s)^2}. \tag{4.31}$$

Hence, $f_\lambda''(s)$ is non-positive, and $f_\lambda'(s)$ is a monotonic increasing function, which is negative for all values of s such that $\frac{(1+\theta)}{2n}\mu'(s) + \frac{(1-\theta)}{2}\frac{e^s}{1+e^s} < \lambda$, and positive for all values of s such that $\frac{(1+\theta)}{2n}\mu'(s) + \frac{(1-\theta)}{2}\frac{e^s}{1+e^s} > \lambda$. Consequently, $f_\lambda(s)$ and $B(\lambda, s)$ are convex $\cap$ function of s, with their maximum value satisfying (4.24).

The behaviour of $B(\lambda_{opt}, s)$ for a GILD code based on a specific constituent code is shown in Figure 4.4. The highest value of λ_{opt} such that $B(\lambda_{opt}, s) > 0$ gives us the average normalized Hamming distance δ. Hence the asymptotic lower bound on the minimum distance of GILD codes can now be calculated as follows:

1. Start with positive value of s.
2. Calculate λ_{opt} using to (4.24).
3. Calculate $B_{opt}(s) = B(\lambda_{opt}, s)$.

4. If $B_{opt}(s) < 0$, decrease the value of s and return to 2, else $\delta = \lambda_{opt}$ is an asymptotical lower bound on the normalized minimum distance of the GILD code.

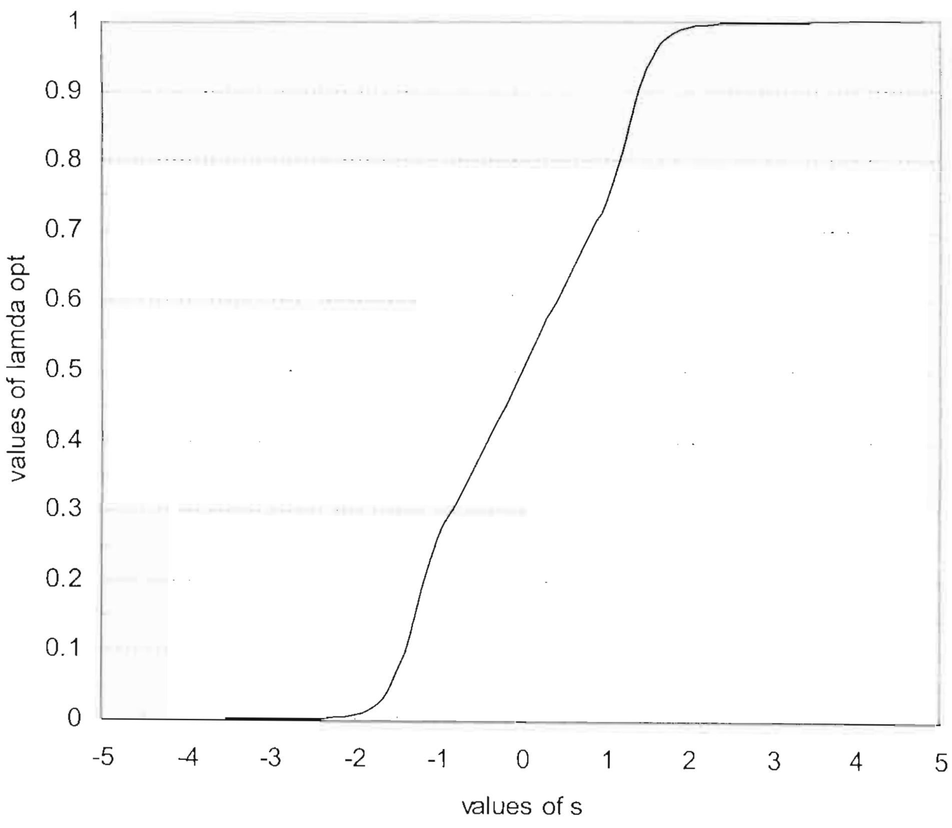

Figure 4.3: $\lambda_{opt}(s)$ for a GILD code based on the (31, 21, 5) BCH component code. θ = 0.903.

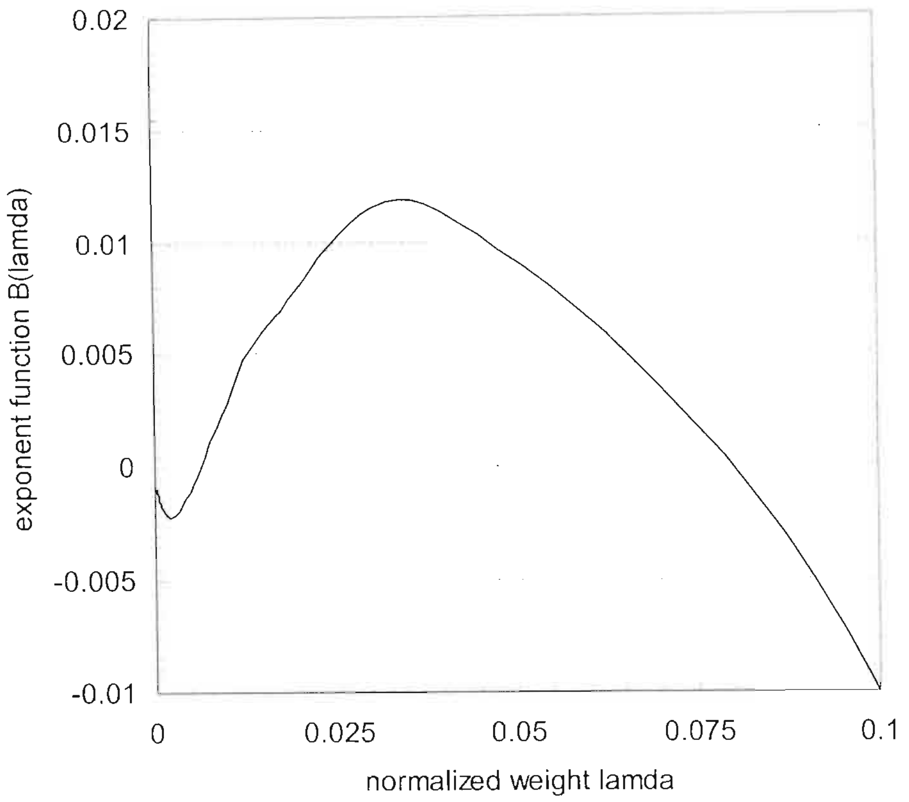

Figure 4.4: $B(\lambda_{opt}, s)$ plotted as a function of λ_{opt} for the GILD code based on the (31, 21, 5) BCH component code. $\theta = 0.903$.

Table 4.1 shows the values of δ found for some constituent codes and different values of θ. The Gilbert-Varshamov bound given by $\delta_o = H_2^{-1}(1-R)$ is also shown. All the values found are strictly positive, which means that GILD codes are asymptotically good for the corresponding values of θ. It can be seen that the asymptotical lower bound on the normalized minimum distance of the GILD code decreases as the values of θ decrease. Also shown in Table 4.1 is, for each constituent code, the value of θ for which we could not find a value of λ satisfying $B(\lambda, s) > 0$. For this value of θ and all the values below it, the corresponding GILD code is not asymptotically good.

Table 4.1: Asymptotic lower bounds on the normalized minimum hamming distance δ of some GILD Codes compared with the Gilbert-Varshamov bound δ_o. * indicates that δ does not exist.

Constituent Code C_o	Value of θ	Rate of the GILD Code	δ	δ_o
Hamming (7, 4, 3)	1	0.143	0.187	0.281
	0.95	0.164	0.144	0.266
	0.9	0.186	0.091	0.252
	0.875	0.196	*	0.245
Hamming (15, 11, 3)	1	0.467	0.026	0.121
	0.975	0.473	0.018	0.119
	0.95	0.48	*	0.117
BCH (t=2) (31, 21, 5)	1	0.355	0.116	0.164
	0.95	0.371	0.099	0.158
	0.9	0.387	0.080	0.151
	0.85	0.403	0.053	0.145
	0.825	0.411	*	0.142
Extended BCH (32, 21, 6)	1	0.313	0.143	0.183
	0.95	0.330	0.126	0.175
	0.9	0.347	0.107	0.168
	0.85	0.364	0.083	0.161
	0.825	0.373	0.066	0.157
	0.8	0.381	*	0.154
BCH (t=2) (63, 51, 5)	1	0.619	0.031	0.074
	0.95	0.629	0.021	0.071
	0.925	0.633	0.015	0.070
	0.9	0.638	*	0.069
Extended BCH (64, 51, 6)	1	0.594	0.038	0.081
	0.95	0.604	0.029	0.078
	0.925	0.609	0.022	0.077
	0.9	0.614	*	0.075

4.3.3 BSC Channel Threshold

In this section, communication over a BSC channel with transition probability p is considered, and the maximum value of p for which the word error probability P_{ew} of an ML decoder goes to zero when N is large is computed. The following theorem gives an upper bound for the asymptotic average error probability of ML decoding of GILD codes over a BSC channel with transition probability p.

Theorem 4.2: Assume a GILD code C of length N built from a linear (n,k) block code C_0 is used on a BSC with crossover probability p, and let the codewords be used with equal probability. Then the asymptotic average error probability of ML decoding of C is upper bounded by:

$$\overline{P_{ew}^{\infty}} \leq \sum_{\lambda=\delta}^{1} D(N,\lambda,p,\varepsilon)\exp\left(-NE(\lambda,s,p)\right), \tag{4.32}$$

where

$$D(N,\lambda,s,p) = N\varepsilon C(\lambda,N)\frac{\lambda}{2}\sqrt{\frac{(1-\lambda)}{2\pi N(p-\lambda/2)(1-p-\lambda/2)}}\exp\left(\frac{1}{12N(1-\lambda)}\right), \tag{4.33}$$

and

$$E(\lambda,s,p) = B(\lambda,s)+H(p)-\lambda\log 2-(1-\lambda)H\left(\frac{p-\lambda/2}{1-\lambda}\right). \tag{4.34}$$

Proof: As a first step, we will derive an upper bound of P_{ew} for any value of N, depending only on the weight distribution $N(l)$ of the GILD code. In a second step, we will use the result (4.10) on the asymptotic average weight distribution of GILD codes to compute $\overline{P_{ew}^{\infty}}$, the asymptotic average error probability of ML decoding of GILD codes, and compare it with the BSC capacity.

Let us assume that the all-zero word $\mathbf{0}$ of a linear binary block code C of length N is transmitted over the BSC channel. Let E be the ensemble of all possible error vectors, that is:

$$E = GF(2)^N \setminus v_0, \tag{4.35}$$

where v_0 is the Voronoi region of $\mathbf{0}$. Let us partition E with respect to the Hamming weight of its vectors $E = \bigcup_{i=1}^{N} E_i$ as:

$$\begin{aligned} E_i &= \left\{ \underline{e} \in E / w(\underline{e}) = i \right\} \\ &= \left\{ \underline{e} \in GF(2)^N / w(\underline{e}) = i \text{ and } \exists \underline{c} \neq \boldsymbol{0} \in C / d_H(\underline{e}, \underline{c}) \le d_H(\underline{e}, \boldsymbol{0}) \right\}, \end{aligned} \tag{4.36}$$

where $w(\ )$ is the Hamming weight of a vector and $d_H(,)$ is the Hamming distance of two vectors. We have:

$$P_{cw} = \sum_{\underline{e} \in E} p(\underline{e}) = \sum_{i=1}^{N} \sum_{\underline{e} \in E_i} p(\underline{e}) = \sum_{i=1}^{N} \sum_{\underline{e} \in E_i} p^i (1-p)^{N-i} = \sum_{i=1}^{N} |E_i| p^i (1-p)^{N-i}. \tag{4.37}$$

Let us denote by $E_i(\underline{c}_j)$ the set of error vectors of weight i that lead to the decoding of the codeword $\underline{c}_j$ of weight j:

$$E_i(\underline{c}_j) = \left\{ \underline{e} \in E_i / d_H(\underline{e}, \underline{c}_j) \le i \right\}. \tag{4.38}$$

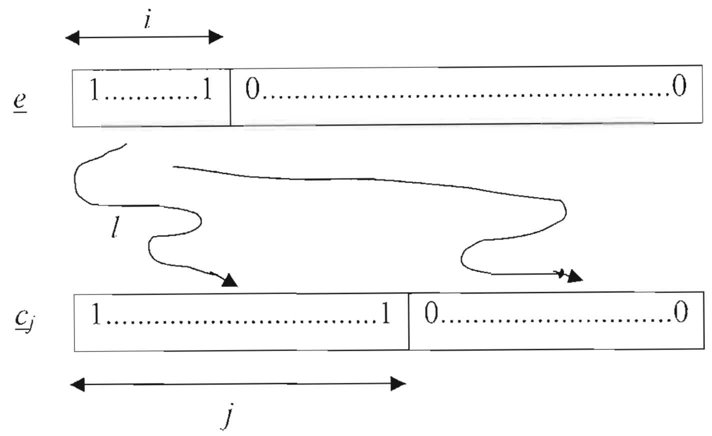

Figure 4.5: Error vector $\underline{e}$ and codeword $\underline{c}_j$. The 1s of each vector are drawn at the first positions of the vectors to help understanding.

Let us denote by l the number of 1s that an element $\underline{e}$ of $E_i(\underline{c}_j)$ has in common with $\underline{c}_j$ (see Figure 4.5). Hence, $\underline{e}$ has $i-l$ ones where $\underline{c}_j$ has zeros. The Hamming distance between $\underline{e}$ and $\underline{c}_j$ is:

$$d_H(\underline{e}, \underline{c}_j) = j - l + i - l = i + j - 2l. \tag{4.39}$$

Equation (4.38) leads to the following inequality for l:

$$\left\lceil \tfrac{j}{2} \right\rceil \le l \le \min(i,j). \qquad (4.40)$$

That means that an error occurs when at least half of the 1-bits of a codeword are covered by an error vector. The same argument is used on the ensemble of the linear binary block codes to derive the random-coding upper bound on the error probability in [23] pp. 92-99. Hence the cardinality of $E_i(\underline{c}_j)$ equals:

$$\left|E_i(\underline{c}_j)\right| \le \sum_{l=\lceil \frac{j}{2} \rceil}^{\min(i,j)} \binom{j}{l}\binom{N-j}{i-l}. \qquad (4.41)$$

Let us denote by $E_{i,j}$ the set of error vectors of weight i that lead to the decoding of a codeword of weight j. There are $N(j)$ such codewords. Because an error vector $\underline{e}$ of $E_{i,j}$ can be closer to two or more codewords of weight j than the all-zero codeword, we have the following inequality:

$$\left|E_{i,j}\right| \le N(j) \sum_{l=\lceil \frac{j}{2} \rceil}^{\min(i,j)} \binom{j}{l}\binom{N-j}{i-l}. \qquad (4.42)$$

We then have:

$$\left|E_i\right| \le \sum_{j=d_{H_{\min}}}^{N} N(j) \sum_{l=\lceil \frac{j}{2} \rceil}^{\min(i,j)} \binom{j}{l}\binom{N-j}{i-l}, \qquad (4.43)$$

and:

$$P_{ew} \le \sum_{i=1}^{N} p^i (1-p)^{N-i} \sum_{j=d_{H_{\min}}}^{N} N(j) \sum_{l=\lceil \frac{j}{2} \rceil}^{\min(i,j)} \binom{j}{l}\binom{N-j}{i-l}. \qquad (4.44)$$

Note that the bounding occurs in and only in (4.42). The last inequality can be used to compute the ML decoding word error probability of any linear code of which we know the weight distribution. It is tighter than the conventional union bound on the error vectors weight combined with the Chernoff-Bhattacharrya bound on the pairwise error probability ([71] p. 63):

$$P_{ew} \le \sum_{l=d_{H_{\min}}}^{N} N(l) \sqrt{4p(1-p)}^{\,l}. \qquad (4.45)$$

We are interested in the maximum crossover probability p of the BSC that leads to an asymptotically vanishing word error probability $\overline{P_{ew}^{\infty}}$ in (4.44) when N tends to infinity. In (4.37), we have to focus our attention on the term that occurs with the highest probability. Intuitively, when N grows, the weight I of the error vectors tends to pN. First, compute the probability that the normalized error weight $\theta = i/N$ is:

$$p-\varepsilon \leq \theta \leq p+\varepsilon\,, \tag{4.46}$$

where ε is an arbitrary nonnegative value. It is closely related to the Chernoff bound on the tails of a binomial distribution. If η is a random variable with finite moments of all order and pdf $p(\eta)$ we have:

$$\text{Prob}(\eta \geq \tau) \leq E\left(e^{s(\eta-\tau)}\right) = e^{\Gamma(s)-s\tau}\,, \tag{4.47}$$

where s is any non-negative value and $\Gamma(s) = \ln \sum_{\eta} e^{s\eta} p(\eta)$. Minimizing (4.47) on s results in:

$$\text{Prob}(\eta \geq \tau) \leq E\left(e^{s(\eta-\tau)}\right) = e^{\Gamma(s)-s\Gamma'(s)}\,, \tag{4.48}$$

where $\tau = \Gamma'(s) = \dfrac{d\Gamma(s)}{ds}$. We apply this equality to the weight of an error vector of length N. We define

$$\eta = w(\underline{e}) = \sum_{n=1}^{N} e_n\,, \tag{4.49}$$

where e_n are the *iid* components of the error vector $\underline{e}$ and take the value 0 with probability $1-p$ and 1 with probability p. Equality (4.48) can be rewritten in this case :

$$\text{Prob}(\eta \geq i) = \text{Prob}(\eta \geq \theta N) \leq e^{N[\gamma(s)-s\gamma'(s)]} \tag{4.50}$$

where

$$\gamma(s) = \frac{\Gamma(s)}{N} = \ln \sum_{e_n} e^{se_n} p(e_n) = \ln(1-p+pe^{s})\,. \tag{4.51}$$

The optimal value of θ is:

$$\theta = \gamma'(s) = \frac{pe^{s}}{1-p+pe^{s}}\,, \tag{4.52}$$

and leads to:

$$e^s = \frac{(1-p)\theta}{p(1-\theta)} \text{ and } s = \ln(1-p) + \ln(\theta) - \ln(p) - \ln(1-\theta), \tag{4.53}$$

$$\gamma(s) = \ln(1-p) - \ln(1-\theta), \tag{4.54}$$

and finally (4.50) results in:

$$\text{Prob}\left(\eta \geq \theta N\right) \leq p^{N\theta}\left(1-p\right)^{N(1-\theta)} e^{NH(\theta)} \text{ for } \theta > p. \tag{4.55}$$

Replacing η by $N-\eta$, p by 1-p and θ by $1-\theta$ leads to the opposite tail bound:

$$\text{Prob}\left(\eta \leq \theta N\right) \leq p^{N\theta}\left(1-p\right)^{N(1-\theta)} e^{NH(\theta)} \text{ for } \theta < p. \tag{4.56}$$

The asymptotic exact expression of $\text{Prob}\left(\eta \leq \theta N\right)$ can be found in [72] pp. 188-193. Let us take $\theta = p+\varepsilon$ in (4.55) and $\theta = p-\varepsilon$ in (4.56) to compute the probability that the normalized error weight is outside the interval $\left[p-\varepsilon, p+\varepsilon\right]$. We have:

$$\begin{aligned}\text{Prob}\left(\eta \geq (p+\varepsilon)\right) \leq{} & p^{Np}(1-p)^{N(1-p)} p^{N\varepsilon}(1-p)^{N(1-\varepsilon)} \\ & \times \exp\left\{N\left[H(p) + \varepsilon H'(p) + \frac{\varepsilon^2}{2}H''(p) + o(\varepsilon^2)\right]\right\}\end{aligned}. \tag{4.57}$$

As

$$H'(p) = \ln\left(\frac{1-p}{p}\right) \text{ and } H''(p) = -\frac{1}{p(1-p)}, \tag{4.58}$$

it follows that:

$$\text{Prob}\left(\eta \geq \left(p+\varepsilon\right)N\right) \leq \exp\left\{-N\frac{\varepsilon^2}{2p\left(1-p\right)} + No\left(\varepsilon^2\right)\right\}, \tag{4.59}$$

and the same bound holds for $\text{Prob}\left(\eta \leq \left(p+\varepsilon\right)N\right)$. Consequently, we have:

$$\text{Prob}\left(\eta \in \left[0, p-\varepsilon\right] \cup \left[p+\varepsilon, 1\right]\right) \leq 2\exp\left\{-N\frac{\varepsilon^2}{2p\left(1-p\right)} + No\left(\varepsilon^2\right)\right\}. \tag{4.60}$$

Hence it is always possible to choose an $\varepsilon > 0$ such that this bound tends to zero when N goes to infinity. We can now rewrite (4.37) as:

$$\begin{aligned}P_{ew} ={} & \sum_{i\in N\times\left[0,p-\varepsilon\right]\cup\left[p+\varepsilon,1\right]} \left|E_i\right| p^i\left(1-p\right)^{N-i} + \sum_{i\in N\times\left[p-\varepsilon,p+\varepsilon\right]} \left|E_i\right| p^i\left(1-p\right)^{N-i} \\ \leq{} & 2\exp\left\{-N\frac{\varepsilon^2}{2p\left(1-p\right)} + No\left(\varepsilon^2\right)\right\} \times 1 + 2N\varepsilon\left|E_{pN}\right| p^{pN}\left(1-p\right)^{N(1-p)}\end{aligned} \tag{4.61}$$

Asymptotically, we have:

$$P_{ew}^{\infty} = 2N\varepsilon P^{pN}(1-p)^{N(1-p)} \sum_{j=d_{H_{\min}}}^{N} N(j) \sum_{l=\lceil \frac{j}{2} \rceil}^{\min(pN,j)} \binom{j}{l}\binom{N-j}{pN-l}$$

$$P_{ew}^{\infty} \le N\varepsilon p^{pN}(1-p)^{N(1-p)} \sum_{\lambda=\delta}^{1} N(\lambda N) \sum_{l=\frac{\lambda}{2}}^{\min(p,\lambda)} \binom{\lambda N}{lN}\binom{N(1-\lambda)}{N(p-l)}. \tag{4.62}$$

The product of the binomial expressions is clearly maximum when $l = \lambda/2$ and it follows:

$$P_{ew}^{\infty} \le N\varepsilon p^{pN}(1-p)^{N(1-p)} \sum_{\lambda=\delta}^{1} N(\lambda N)\frac{\lambda}{2}2^{\lambda N}\binom{N(1-\lambda)}{N(p-\frac{\lambda}{2})}. \tag{4.63}$$

Using the asymptotic average distribution $\overline{N(l)}$ of GILD codes (4.10) and the asymptotic equivalent of the binomial coefficient (4.18), and focusing on the exponential term, we obtain:

$$\begin{aligned}\overline{P_{ew}^{\infty}} \le \sum_{\lambda=\delta}^{1} N\varepsilon C(\lambda,N)\frac{\lambda}{2}\sqrt{\frac{(1-\lambda)}{2\pi N\left(p-\frac{\lambda}{2}\right)\left(1-p-\frac{\lambda}{2}\right)}}\exp\left\{\frac{1}{12N(1-\lambda)}\right\} \\ \times\exp\left\{-N\left[B(\lambda,s)+H(p)-\lambda\ln 2-(1-\lambda)H\left(\frac{p-\lambda/2}{1-\lambda}\right)\right]\right\}\end{aligned} \tag{4.64}$$

we define:

$$D(N,\lambda,s,p) = N\varepsilon C(\lambda,N)\frac{\lambda}{2}\sqrt{\frac{(1-\lambda)}{2\pi N(p-\lambda/2)(1-p-\lambda/2)}}\exp\left(\frac{1}{12N(1-\lambda)}\right),$$

and

$$E(\lambda,s,p) = B(\lambda,s)+H(p)-\lambda\log 2-(1-\lambda)H\left(\frac{p-\lambda/2}{1-\lambda}\right).$$

This completes the proof of theorem 3.2.

The asymptotic average word error probability $\overline{P_{ew}^{\infty}}$ tends to zero if the smallest term $E(\lambda,s,p)$ in the exponential part is non-negative. Defining $E(p)$ as:

$$E(p) = \min_{\lambda,s}\{E(\lambda,s,p)\}, \tag{4.65}$$

we have:

$$\overline{P_{cw}^{\infty}} \to 0 \Leftrightarrow E(p) > 0. \qquad (4.66)$$

We can now find the BSC crossover probability threshold p which is the highest value for which $E(p)$ is non-negative, with the following algorithm.

1. Start with a small value of p.
2. Calculate the minimum value $E(p)$ of $E(\lambda, s, p)$ using to (4.34) and (4.12) by varying λ (and s as they are related).
3. If $E(p) > 0$, increase the value of p and return to 2, else $p_{thres} = p$ is an asymptotical upper bound on the crossover probability of the BSC that leads to a vanishing word error probability of ML decoding of the GILD code.

Table 4.2 shows the value of p compared with $P(C)$, the probability threshold for a code achieving the capacity at the same rate for different kinds of GILD codes. GILD codes thus achieve near-capacity performance on BSC channels when their length is arbitrarily large.

4.4 Upper Bound on Minimum Distance for a Fixed Length

In this section, expression (4.9) of the average weight distribution of the GILD codes is used to compute an upper bound on the minimum Hamming distance of the ensemble of GILD codes *with fixed length*. The major difference with Section 4.3.2 is that inequalities (4.13) and (4.14) are not used.

The probability that the minimum distance $d_{H\min}$ of a linear block code C of length N is lower than, or equal to, D is the probability that there exists a sequence $\underline{v}$ of length N and weight l lower than, or equal to, D that is a codeword of C:

$$\text{Prob}\left(d_{H\min} \le D\right) = \text{Prob}\left(\exists \underline{v} \in \text{GF}(2)^N \,/\, w(\underline{v}) = l \le D \text{ and } \underline{v} \in C\right). \qquad (4.67)$$

This probability is clearly less than the sum of the probabilities that the individual sequences $\underline{v}$ of the considered weight are codewords:

Table 4.2: BSC crossover probability threshold p of some GILD codes compared with the threshold $P(C)$ of the code of the same rate achieving capacity.

Constituent Code C_0	Value of θ	Rate of the GILD Code	p	$P(C)$
Hamming (7, 4, 3)	1	0.143	0.278	0.281
	0.95	0.164	0.260	0.266
	0.9	0.186	0.242	0.252
BCH (t=2) (31, 21, 5)	1	0.355	0.164	0.165
	0.95	0.371	0.156	0.158
	0.9	0.387	0.149	0.151
	0.85	0.403	0.142	0.145
Extended BCH (32, 21, 6)	1	0.313	0.183	0.183
	0.95	0.330	0.175	0.175
	0.9	0.347	0.167	0.168
	0.85	0.364	0.159	0.160
	0.825	0.373	0.155	0.157
BCH (t=2) (63, 51, 5)	1	0.619	0.071	0.074
	0.95	0.629	0.068	0.071
	0.925	0.633	0.066	0.070
Extended BCH (64, 51, 6)	1	0.594	0.079	0.081
	0.95	0.604	0.075	0.078
	0.925	0.609	0.074	0.077

$$\text{Prob}\left(d_{H\min} \le D\right) \le \sum_{\underline{v}/l \le D} \text{Prob}\left(\underline{v} \in C\right) = \sum_{l=1}^{D} \binom{N}{l} P(l), \tag{4.68}$$

where $P(l)$ is defined as the probability that a sequence of weight l is a codeword.

We calculate $P(l)$ for the average ensemble of GILD codes of fixed length N using (4.8). Applying (4.9) in (4.68) leads to:

$$\text{Prob}\left(d_{H\min} \le D\right) \le \sum_{l=1}^{D} \overline{N(l)}. \tag{4.69}$$

An upper bound for the minimum Hamming distance of C is computed by taking the right hand side of (4.69) greater than, or equal to, 1. We have:

$$d_{H\min} \leq \Delta, \tag{4.70}$$

where Δ is the smallest integer such that

$$\left\lfloor \sum_{l=1}^{\Delta} \overline{N(l)} \right\rfloor \geq 1. \tag{4.71}$$

Figure 4.6 shows this bound Δ calculated for the GILD code based on the (31, 21, 5) BCH component code for two values of θ. The granularity of the curves is due to the fact that all the codes considered do not correspond to the exact values of θ, as the length of a GILD code has to be a multiple of the constituent code length. The average minimum distance of these codes is clearly a linear function of their length. It is, however, observed that GILD codes have lower minimum Hamming distance when compared with the corresponding GLD codes.

4.5 GILD Codes Based on Different Constituent Codes

The definition and construction in Section 4.2 assume the sane constituent code is used. However, GILD can also be constructed using different constituent code in the irregular ensemble.

4.5.1 Definition

A binary GILD code with two levels based on different constituent codes is defined by four parameters:

- The length of the code N.
- The length n_1 of the constituent code C_1, with dimension k_1 and rate r_1.
- The length n_2 of the constituent code C_2, with dimension k_2 and rate r_2.
- The fraction, θ ($\theta < 1$), of the code symbols connected to the second super-code.

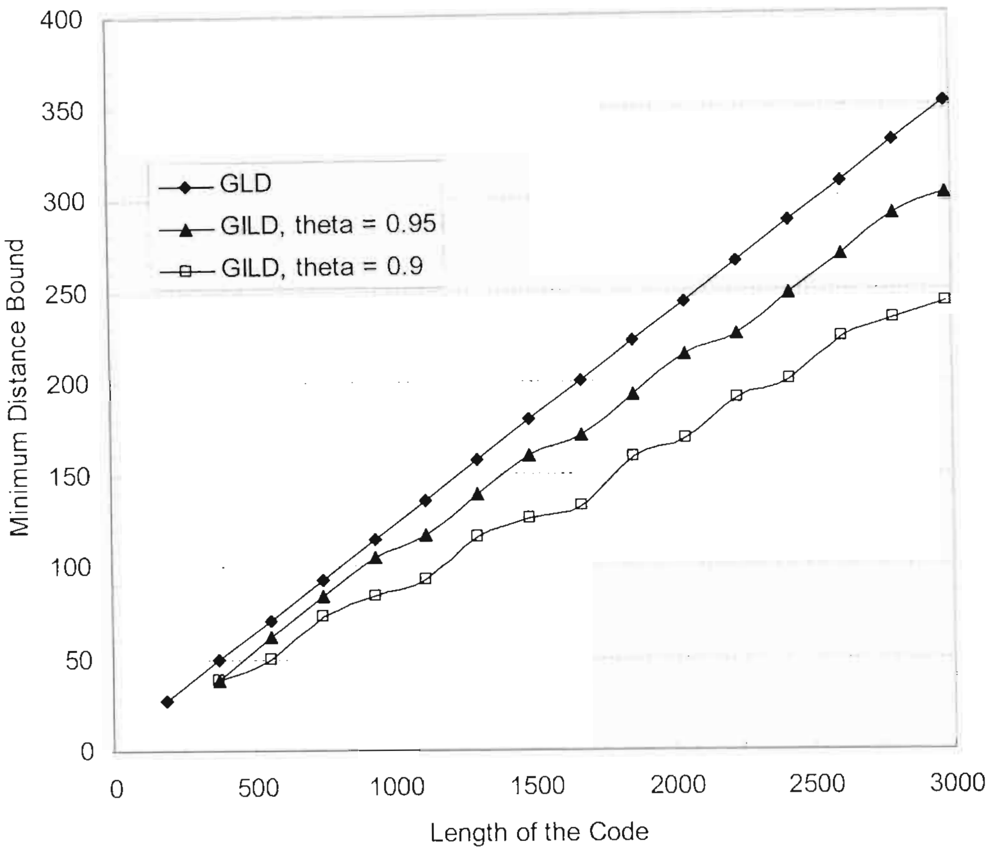

Figure 4.6: Upper bounds using (4.71) for GILD codes based on the (31, 21, 5) BCH component code.

Assuming that C_1 is used in the first super-code and C_2 is used in the second super-code, then N must be a multiple of n_1 and θN a multiple of n_2.

4.5.2 Contruction

The construction of the code is also achieved through a superposition method. To construct the parity-check matrix $\boldsymbol{H}$ of such a code, we first construct a low-density (LD) matrix with $N\left(\frac{1}{n_1}+\frac{\theta}{n_2}\right)$ rows and N columns, having exactly:

- θN columns with 2 ones
- $N(1-\theta)$ columns with 1 one

- $\frac{N}{n_1}$ rows with n_1 ones in each
- $\frac{\theta N}{n_2}$ rows with n_2 ones in each
- Zeros elsewhere

Then we do the following:

1. In each the $\frac{N}{n_1}$ rows with n_1 ones, we replace the $N-n_1$ zero elements by a zero column-vector of length (n_1-k_1), and the nonzero elements by one of the n_1 different columns of the constituent code C_1's parity-check matrix, each used once.
2. In each of the $\frac{\theta N}{n_2}$ rows with n_2 ones, we replace the $N-n_2$ zero elements by a zero column-vector of length (n_2-k_2), and the nonzero elements by one of the n_2 different columns of the constituent code C_2's parity-check matrix, each used once.

The LD matrix should also be constructed such that its Tanner graph contains no cycle of length 4. If the parity-check matrix $\boldsymbol{H}$ has full rank, the total rate of the GILD code is:

$$R'' = r_1 + \theta r_2 - \theta \tag{4.72}$$

If $\theta = 1$, we obtain a GLD code based on different constituent codes.

4.5.3 Ensemble Performance

Let C be a binary GILD code with two levels based on different constituent codes as defined above. Following the same reasoning as in Section III, it can be shown that the average number of codewords in C having weight l is given by:

$$\overline{N(l)} = \frac{2^{N\left[\frac{k_1}{n_1}+\frac{k_1}{n_2}+(1-\theta)\right]} Q(l) K(l)}{\binom{N}{l}}. \tag{4.73}$$

For $l > 0$, $\overline{N(l)}$ is upper bounded by:

$$\overline{N(l)} \leq C(\lambda, N) \times e^{-NB'(\lambda,s)}, \tag{4.74}$$

where $C(\lambda, N)$ is defined by (4.11) and

$$B'(\lambda,s) = H(\lambda) + 2\lambda s - \frac{1}{n_1}\left(\mu_1(s) + k_1 \log 2\right) - \frac{\theta}{n_2}\left(\mu_2(s) + k_2 \log 2\right) - (1-\theta)\log\left(1+e^s\right). \tag{4.75}$$

In (4.75), $\mu_1(s) = \log(g_1(s))$ and $\mu_2(s) = \log(g_2(s))$, where $g_1(s)$ and $g_2(s)$ are the moment generating functions of the constituent codes C_1 and C_2 respectively.

Asymptotically, when the length N tends to infinity, $\overline{N(l)}$ tends to zero if $B'(\lambda, s)$ is strictly positive. As in Section II, the largest value δ of λ satisfying $B'(\delta, s) = 0$ and $B'(\lambda, s) < 0$ for $\lambda > \delta$ gives an asymptotical lower bound on the normalized minimum distance of the GILD code. The optimal value of s is related to the weight by:

$$\lambda_{opt}' = \frac{\mu_1'(s)}{2n_1} + \frac{\theta\mu_2'(s)}{2n_2} + \frac{(1-\theta)}{2} \times \frac{e^s}{1+e^s}, \tag{4.76}$$

where $\mu_1'(s)$ and $\mu_2'(s)$ are the derivative of $\mu_1(s)$ and $\mu_2(s)$ relative to s. Table III shows the values of δ found for a GILD code based on the Hamming (31, 26, 3) constituent code and the BCH (31, 21, 5) constituent code. As in Table 4.3, the Gilbert-Varshamov bound is also shown. It can be seen that the order in which the constituent codes are used affects both the rate and the value of δ. As will be shown by the simulation results in Section VI, this order also affects the code performance.

Table 4.3: Asymptotic lower bounds on the normalized minimum hamming distance δ of some GILD Codes based on two different constituent codes compared with the Gilbert-Varshamov bound δ_o. * indicates that δ does not exist.

Constituent Codes	Value of θ	Rate of the GILD Code	δ	δ_o
C_1 : Hamming (31, 26, 3) C_2 : BCH (t=2) (31, 21, 5)	1	0.516	0.013	0.105
	0.985	0.521	0.01	0.103
	0.975	0.524	0.007	0.102
	0.967	0.526	0.004	0.101
	0.965	0.527	*	0.101
C_1 : BCH (t=2) (31, 21, 5) C_2 : Hamming (31, 26, 3)	1	0.516	0.013	0.105
	0.985	0.518	0.011	0.104
	0.975	0.52	0.008	0.103
	0.967	0.521	0.005	0.103
	0.965	0.522	*	0.102

4.6 ML Decoding Error Probability over the AWGN Channel

In this section, the exact average weight distribution $\overline{N(l)}$ is used to compute the average bit error probability of maximum likelihood (ML) decoding of GILD codes. Indeed, the interleaver acts on all the coded bits[8] so that they are equally protected. Hence from any bound on the word error probability over the AWGN channel, we can derive an upper bound on the average bit error probability $\overline{P_{eb}}$ without having to compute the input-output weight enumerator function (IOWEF) as in the conventional methods to bound the ML decoding performance of turbo-codes.

4.6.1 Union Bound

The Union-Bound (UB) for a transmission over an AWGN channel is given by:

$$\overline{P_{eb}} \le \sum_{l=1}^{N} \frac{l}{N} \times \overline{N(l)} \times \frac{1}{2} \operatorname{erfc}\left(\sqrt{Rl \frac{E_b}{N_o}} \right), \tag{4.77}$$

[8] This is a major difference with classical compound codes such as parallel or serial concatenated codes or product codes.

where E_b/N_0 is the signal-to-noise ratio per information bit and R the GILD code rate. The classical drawback of the UB is that it is not tight and even diverges for low values of the signal-to-noise ratio per information bit.

4.6.2 Improved Bound Based on Gallager's Bound

In [73], Duman & Saheli derived an upper bound on the word (and bit) error probability of turbo codes with ML decoding using a modified version of Gallager's bound [72] rather than the standard union bound. Their method is directly applicable to any linear code, and thus to GILD codes.

The broad outlines of this bound are the partition of the code to constant-weight sub-codes, the application of Gallager's bound on each sub-code and finally the union bound to get an upper bound on the word (and bit) error probability of the overall code. The code C is partitioned in the set of sub-codes C_l, $l = 1, \cdots, N$ defined as the collection of the all-zero codeword together with all the codewords of weight l. Note that C_l is not necessarily linear. Let us denote by D_i the Voronoi region associated with $\underline{c}_i, i = 1, \cdots, N$ when it is considered as a codeword of C, and denote by D_i' its Voronoi region when it is considered as a codeword of C_l. It is clear that for all i:

$$D_i \subseteq D_i' . \tag{4.78}$$

Assuming the all-zero codeword $\underline{c}_0$ is emitted, the union bound gives:

$$P_{cw} = \sum_{i=1}^{N} \int_{y \in D_i} p(y/\underline{c}_o)\,dy \leq \sum_{l=1}^{N} \sum_{\substack{\underline{c}_i \in C_l \\ \underline{c}_i \neq \underline{c}_o}} \int_{y \in D_i'} p(y/\underline{c}_o)\,dy \triangleq \sum_{l=1}^{N} P_{cw}^{(l)} , \tag{4.79}$$

where $p(y/\underline{c}_o)$ is the likelihood of the codeword $\underline{c}_0$. Duman & Saheli applied Gallager's bound to $P_{cw}^{(l)}$, and found after some manipulations:

$$P_{cw}^{(l)} \le N(l)^{\rho} \alpha^{-N\frac{1-\rho}{2}} \left(\alpha - \frac{\alpha-1}{\rho}\right)^{-N\frac{\rho}{2}}$$

$$\exp\left(NR\frac{E_b}{N_o}\left[-1 + \frac{\beta^2}{\alpha}(1-\rho) + \rho\left(1-\frac{l}{N}\right)\frac{\left(\beta+\frac{1-\beta}{\rho}\right)^2}{\alpha-\frac{\alpha-1}{\rho}} \right]\right) \tag{4.80}$$

for the AWGN channel case, where $\beta = (1-l/N)/[(1/\alpha)-(l/N)(1-\rho)]$, $0<\rho\le 1$, and $0<\alpha\le 1/(1-\rho)$. α and β are two parameters that have to be optimized to find the lowest value of $P_{cw}^{(l)}$.

Consequently, using the average weight distribution $\overline{N(l)}$ of the GILD codes, we have the following improved upper bound on the ML decoding bit error probabilities:

$$\overline{P_{eb}} \le \sum_{l=1}^{N} \frac{l}{N} \min_{\substack{0<\rho\le 1 \\ 0<\alpha\le 1/(1-\rho)}} \overline{P_{cw}^{(l)}}, \tag{4.81}$$

where $\overline{P_{cw}^{(l)}}$ is equal to the expression (4.80) where $N(l)$ is replaced by $\overline{N(l)}$.

4.6.3 Tangential Sphere Bound

An improved version of the tangential bound of Berlekamp [74] has been presented by Poltyrev ([75] and [76]) and applied to turbo-codes by Sason and Shamai ([77] and [78]). This *tangential sphere bound* is tighter than the Duman & Saheli bound, and its properties follow the central inequality,

$$\text{Prob}(A) \le \text{Prob}(\underline{z}\in B, A) + \text{Prob}(\underline{z}\notin B). \tag{4.82}$$

In the case of the tangential sphere bound, A is an event that represents a message decoding error, B is an N-dimensional cone with a half angle θ and radius $r = \sqrt{NE_s}\tan\theta$, and $\underline{z}$ is the noise vector added to the transmitted signal. The tangential sphere bound on the block error probability P_e is based only on the distance spectrum $\{N(l)\}$ of the binary linear block code C and it reads:

$$P_e \leq \int_{-\infty}^{+\infty} \frac{e^{-\frac{z_1^2}{2\sigma^2}} dz_1}{\sqrt{2\pi\sigma^2}} \left\{ \begin{array}{l} 1-\bar{\gamma}\left(\frac{N-1}{2}, \frac{r_{z_1}^2}{2\sigma^2}\right) \\ + \sum_{l: \frac{\delta_l}{2} < \alpha_l} N(l) \left[Q\left(\frac{\beta_l(z_1)}{\sigma}\right) - Q\left(\frac{r_{z_1}}{\sigma}\right)\right] \bar{\gamma}\left(\frac{N-2}{2}, \frac{r_{z_1}^2 - \beta_l^2(z_1)}{2\sigma^2}\right) \end{array} \right\}. \quad (4.83)$$

The following parameters are used in (4.83):

$$\left\{ \begin{array}{l} \sigma^2 = \frac{N_o}{2}, \text{ with } N_o \text{ standing for the one-sided spectral density of the AWGN} \\ r_{z_1} = \left(1 - \frac{z_1}{\sqrt{NE_s}}\right) r \\ \beta_l(z_1) = \frac{r_{z_1}}{\sqrt{1-\frac{\delta_l^2}{4NE_s}}} \cdot \frac{\delta_l}{2r} \\ \alpha_l = r\sqrt{1-\frac{\delta_l^2}{4NE_s}}, \end{array} \right. \quad (4.84)$$

δ_l is defined to be the Euclidean distance between two signals whose corresponding codewords differ in l symbols $(l \leq N)$. Thus for the case of antipodal signals $\delta_l = 2\sqrt{lE_s}$. Also, $\bar{\gamma}(a,x)$ designates the normalized incomplete gamma function defined as:

$$\bar{\gamma}(a,x) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt \text{, for positive values of } a \text{, and } x. \quad (4.85)$$

A geometric interpretation of the tangential sphere bound is presented in Figure 3.7. The upper bound (4.83) is valid for all positive values of r and thus the optimal radius r_o (or equivalently θ_o), in the sense of achieving the tightest upper bound, is determined by nullifying the derivative of the right hand side of the bound (4.83), yielding the following optimisation equation:

$$\begin{cases} \sum_{l:\frac{\delta_l}{2}<\alpha_l} N(l)\int_0^{\theta_l}\sin^{n-3}\phi d\phi = \dfrac{\sqrt{\pi}\Gamma\left(\frac{N-2}{2}\right)}{\Gamma\left(\frac{N-1}{2}\right)} \\ \theta_l = \cos^{-1}\left(\dfrac{\delta_l}{2r}\dfrac{1}{\sqrt{1-\dfrac{\delta_l^2}{4NE_s}}}\right) \end{cases} . \tag{4.86}$$

The bit error probability bound on the ensemble of GILD code is obtained by replacing $N(l)$ by $l/N(l)\times \overline{N}(l)$ in (4.83) and (4.86).

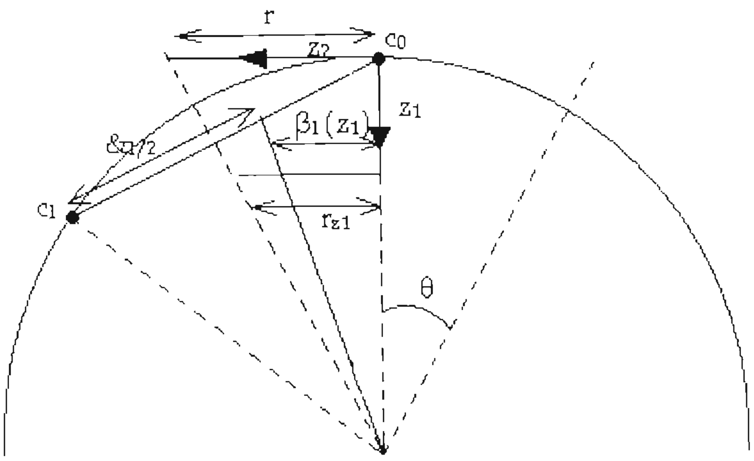

Figure 4.7: Illustration of the tangential sphere bound.

4.7 Decoding of GILD Codes

GILD codes are not LDPC codes, and thus cannot be decoded with the belief-propagation or sum-product algorithm (SPA). The decoding of GILD codes is based on iterative soft-input soft-output (SISO) decoding of individual constituent codes. For low rate GILD codes, exploiting the fact that the constituent code usually has a small code length and high code rate, a trellis-based algorithm can be used to obtain high error-correcting performance with reasonable decoding complexity. In our implementation, we used the trellis-based MAP (maximum a posteriori probability) algorithm [40], also referred to as BCJR, on the syndrome trellis of the constituent code. The algorithm is summarized below. However, if long and powerful constituent codes are used, any trellis-based solution becomes prohibitively complex and impractical as the trellis complexity grows exponentially with the dimension of any sequence of good codes [79]. In these cases, sub-optimal decoding algorithms are required. There are a variety of sub-optimal soft-decision decoding algorithms available to decode block codes, including Chase's algorithm [80], the generalized minimum distance (GMD) decoding algorithm [81], the order-i reprocessing [82], the Kaneko algorithm [83]. If these soft-input hard-output (SIHO) algorithms are to be used in an iterative decoding context they must produce soft information. One technique for doing this is discussed in [84] and another in [85], [86]. In our implementation, we used the Chase-based algorithm of [85] and [86]. This algorithm is also summarized below.

4.7.1 MAP Decoding of Linear Block Codes

Consider a binary (*n*, *k*) linear block code *C* with minimal bit-level trellis *T* [87]. Each path through the trellis *T* represents one distinct codeword in *C*. Let $\boldsymbol{u} = (u_1, u_2, \cdots, u_n)$ be a codeword in *C*. Assumed $\boldsymbol{u}$ is modulated with binary phase-shift keying (BPSK) and transmitted over a time discrete AWGN channel with a one-sided noise spectral density *No*. Denote the received noisy sequence as $\boldsymbol{r} = (r_1, r_2, \cdots, r_n)$.

Let σ_k denote a state in the trellis *T* at time-*k* and $B_k(C)$ denote the set of all branches (σ_{k-1}, σ_k) that connect the states at time-(*k*-1) and time-*k* in *T*. Let $B_k^0(C)$ and $B_k^1(C)$ denote the two disjoint subsets of $B_k(C)$ that correspond to the output code bits $u_k = 0$ and $u_k = 1$ respectively. The MAP rule provides us the log-likelihood ratio (LLR) associated with each bit u_k:

$$L_R(u_k) = L_c.r_k + L(u_k) + \log \frac{\sum_{(\sigma',\sigma)\in B_k^1(C)} \alpha_{k-1}(\sigma').\beta_k(\sigma)}{\sum_{(\sigma',\sigma)\in B_k^0(C)} \alpha_{k-1}(\sigma').\beta_k(\sigma)}, \qquad (4.87)$$

where the channel reliability factor $L_c = \frac{4E_s}{N_o}$ (E_s denotes the energy per symbol), $L(u_k)$ is the a priori value of $\log \frac{P(u_k = 1)}{P(u_k = 0)}$ about u_k. The third term in (4.87) represents *extrinsic* information, $L_{extr}(u_k)$, in which forward recursion metric $\alpha_k(\sigma)$ and backward recursion metric $\beta_k(\sigma)$ are recursively calculated as:

$$\alpha_k(\sigma) = \sum_{(\sigma',\sigma) \in B_k(C)} \alpha_{k-1}(\sigma').\exp((L_c.r_k + L(u_k)).x_k/2), \tag{4.88}$$

and

$$\beta_{k-1}(\sigma') = \sum_{(\sigma',\sigma) \in B_k(C)} \beta_k(\sigma).\exp((L_c.r_k + L(u_k)).x_k/2), \tag{4.89}$$

respectively, where $x_k = 2l_k - 1$ and $l_k \in \{0, 1\}$ represents the label of the branch (σ', σ). In the iterative decoding, only the extrinsic information can be passed on to a subsequent decoder as a priori value $L(u_k)$.

4.7.2 Chase-Based Decoding Algorithm

The second algorithm described in [80] is used to produce a set of codewords which are used to calculate the extrinsic information required in iterative decoding. The Chase decoder generates $2^{\lfloor d_H/2 \rfloor}$ test patterns, where d_H is the minimum Hamming distance of the component code. The test patterns form the set of sequences of length n, the length of the component code, containing all binary combinations in the $\lfloor d_H/2 \rfloor$ least reliable positions in the soft-input to the decoder. The $p \geq \lfloor d_H/2 \rfloor$ least reliable positions can be considered as in [85], leading to 2^p test patterns. Test sequences are created by adding the test pattern to the hard threshold decisions on the soft-input (using modulo 2 addition). Each test sequence is then algebraically decoded to produce a list of possible codewords. If an extended block code is used then the codeword consists of an inner block code with an overall parity added. Only the inner code is used in the search for the p least reliable positions. The Chase decoder algebraically decodes the test sequences for the inner code. It then calculates an overall parity bit for the decoded inner codeword to produce the extended codeword.

Once the set of codewords containing the closest codewords to the received vector is produced by the Chase decoder, the closest codeword in the Euclidean distance sense to this vector is selected as the decision. After the decision is obtained, the reliability of each of its components is calculated to generate soft-decision at the output of the decoder. The soft-output is required so that the extrinsic information generated can be passed to the next decoder in an iterative decoding scheme. See [85] and [86] for more details.

4.7.3 Iterative Decoding of GILD Codes

GILD codes can effectively be decoded using the following decoding scheme. For each bit, we compute its probability given its received sample considering that it belongs to the super-code C^1. We use N/n SISO decoders working in parallel on the N/n independent constituent codes of C^1. Each decoder is implemented using one of the above algorithms. This step generates for each coded symbol an *a posteriori* probability and an *extrinsic* probability. The latter one is fed through the interleaver to the second step as a *priori* information for the $\theta N/n$ SISO decoders working on the $\theta N/n$ constituent codes of C^2. This process is iterated on each super-code: $C^1 \rightarrow C^2 \rightarrow C^1 \rightarrow C^2 \rightarrow C^1 \rightarrow \ldots$ until the preset maximum iteration number is reached or a stopping criterion is satisfied.

4.8 Simulation Results

Iterative decoding of several GILD codes suitable for small frame systems for an AWGN channel with binary input was simulated. The construction each code is achieved as described in Section 4.2 using a random interleaver. For each code the constituent code are decoded using the MAP algorithm. In the decoding process, we limited the number of iteration to 10 unless otherwise stated. The performances are given in terms of the bit-error rate (BER) versus the normalized signal-to-noise ratio (SNR) per information bit. At each SNR value, at least 100 codeword are detected.

The first code has length $N = 961$. Its constituent code is the $(31,21,5)$ BCH code. Its performance is shown in Figure 4.8 for different values of θ, and compared with the corresponding GLD code. The rate of the GLD code is 0.355 and the rates of the GILD codes are 0.386, 0.417, 0.428, and 0.438 for $\theta = 0.903$, $\theta = 0.806$, $\theta = 0.774$, and $\theta = 0.742$ respectively. This corresponds to an increase in the code rate of about 9%, 17%, 21%, and 23% respectively. It can be observed that the GILD codes with $\theta = 0.903$ and $\theta = 0.806$ outperform the GLD code, with an improvement of about 0.3 dB and 0.6 dB at a BER of 10^{-4} respectively. It can also be observed that the performances of the GILD code with $\theta = 0.774$ and $\theta = 0.742$ start to degrade at medium to high signal-to-noise ratio. This is probably due to the fact that for this values of θ, the GILD code has a much smaller minimum distance.

In Figure 4.9, the performance of a GILD code of length $N = 961$ based on two different constituent codes is shown for different values of θ. The constituent codes are the $(31,21,5)$ BCH code and the $(31,26,3)$ Hamming code. The BCH code is used in the first super-code and the Hamming code is used in the second super-code. Compared to the first case where the BCH code is used in both super-codes, this results in higher rate codes. The rate of the GLD code is 0.516 and the rates of the GILD codes are 0.532, 0.547, and 0.558 for $\theta = 0.903$, $\theta = 0.806$, and $\theta = 0.742$ respectively. It can be observed that, at a BER of 10^{-4}, the GILD code with $\theta = 0.806$ outperforms the GLD code by about 0.25 dB. In Figure 4.10, the constituent codes are used in the reverse order, i.e. the Hamming code is used in the first super- code and the BCH code is used in the second super-code. For the same value of θ, this results in further increase in the code rate. However, there is a degradation in the code's performance and the allowable values of θ. This reveals the interesting finding that if two different constituent codes are used, the stronger code should be used in the first super-code.

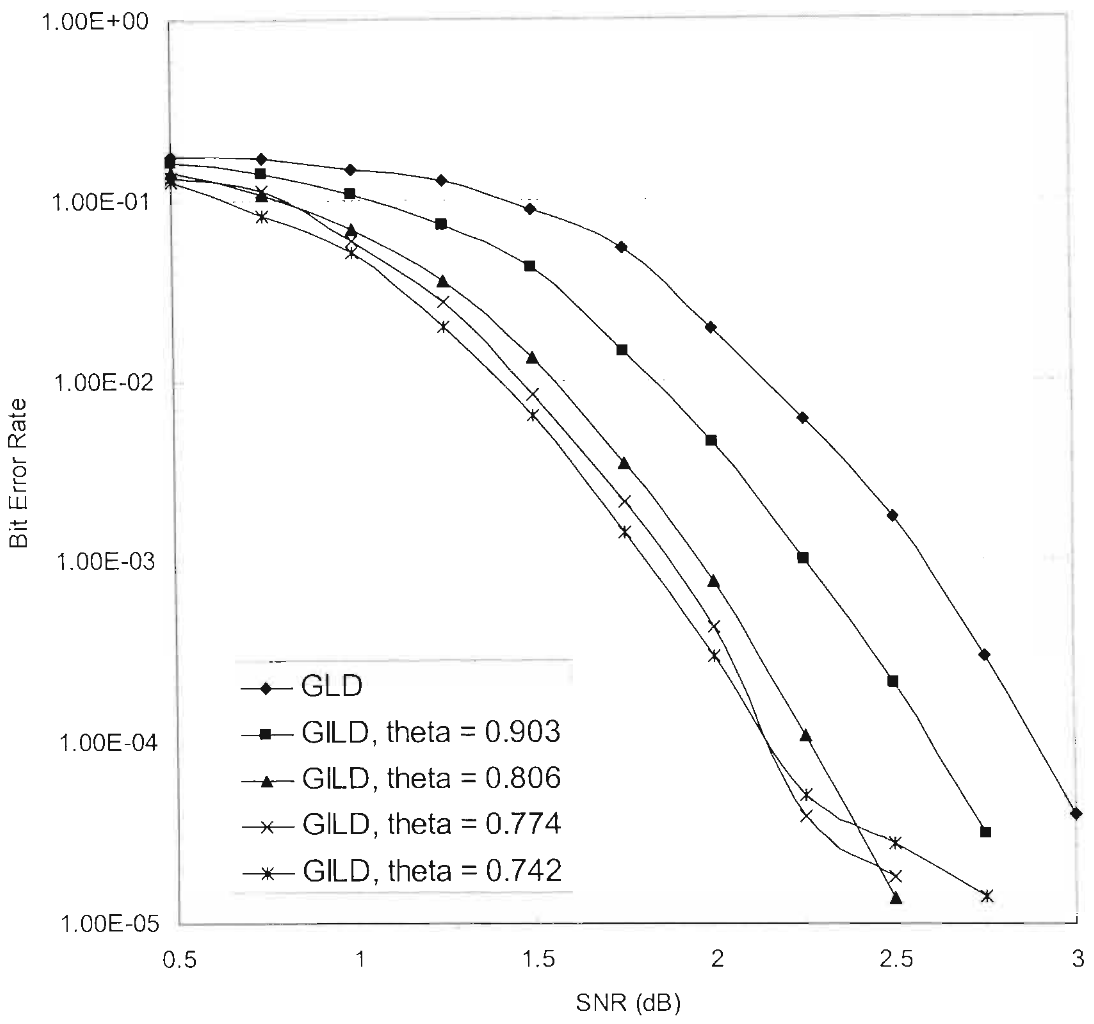

Figure 4.8: Simulated performance of four GILD codes and the corresponding GLD code built from the $(31,21,5)$ BCH constituent code, length $N = 961$, AWGN channel.

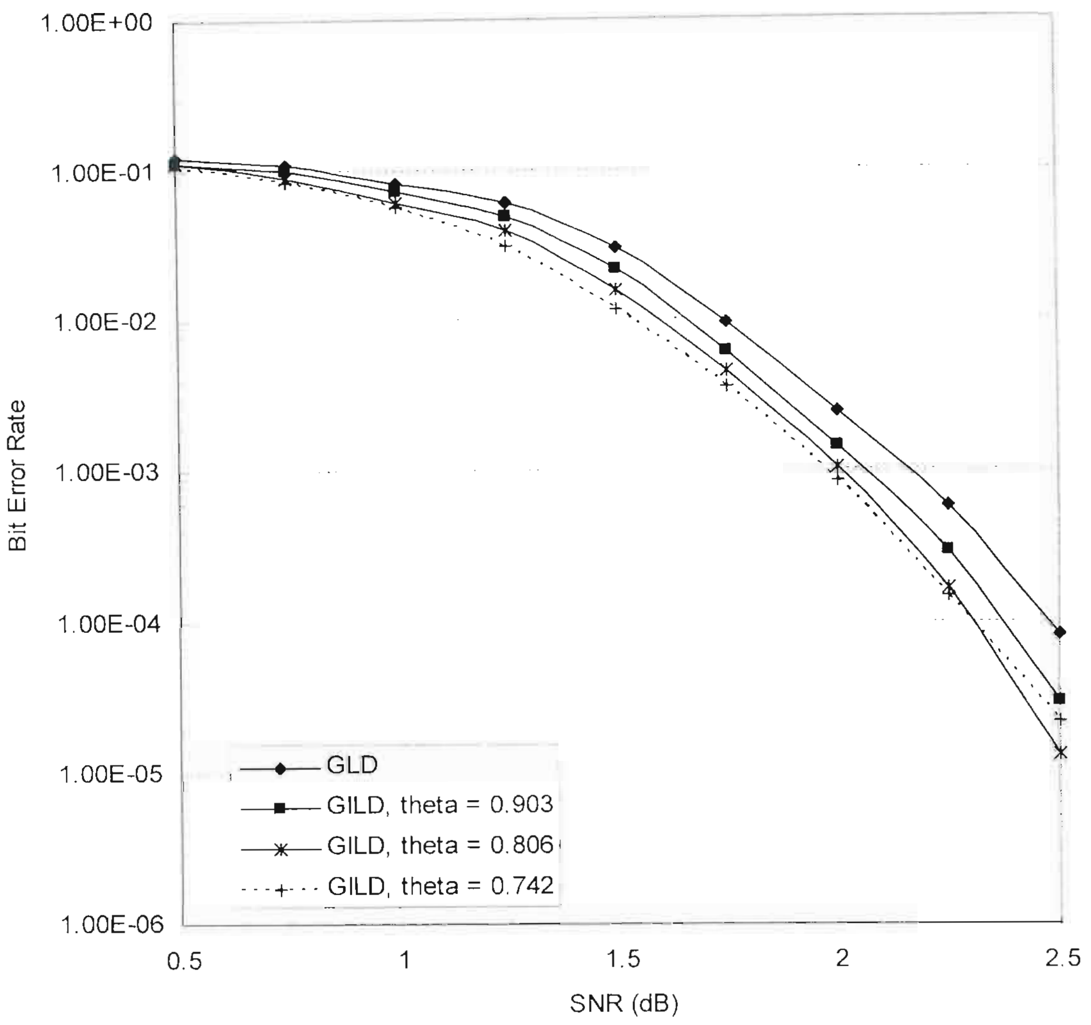

Figure 4.9: Simulated performance of three GILD codes and the corresponding GLD code built from two different constituent codes. The $(31,21,5)$ BCH (used in the first super-code) and the $(31,26,3)$ Hamming code (used in the second super-code), length $N = 961$, AWGN channel.

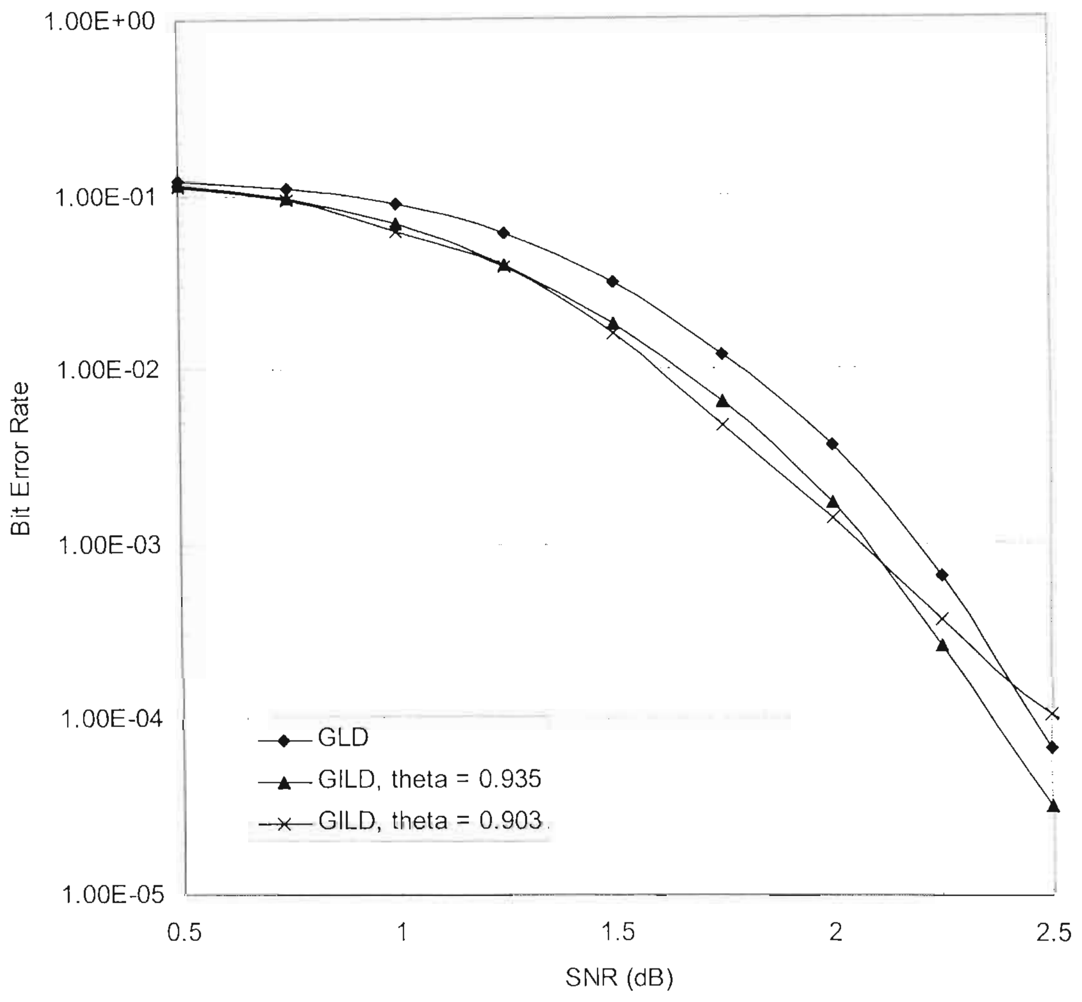

Figure 4.10: Simulated performance of two GILD codes and the corresponding GLD code built from two different constituent codes. The $(31,26,3)$ Hamming code (used in the first super-code) and the $(31,21,5)$ BCH code (used in the second super-code), length $N = 961$, AWGN channel.

In Figure 4.11, a GILD code of length $N = 1005$, $\theta = 0.910$, and rate 0.491 built from the $(15,11,3)$ Hamming code is compared with two LDPC codes of length 1008 and rate 0.5 constructed using the progressive edge growth (PEG) algorithm [50, 169]. The PEG construction creates matrices with very large girth and, to the best of our knowledge, yields the best binary LDPC codes at short block lengths. The regular LDPC code has a column weight of 3. The code-node degree distributions used by the PEG algorithm to generate the LDPC code are designed by the density evolution [19] approach and are given in [60]. Note that, in the PEG algorithm, the check-node distribution is not needed as the check-degree sequence is made as uniform as possible by the algorithm. To the best of our knowledge, this irregular PEG LDPC code is the best known code of this block length and rate to date [169, 170]. The LDPC codes are decoded using the SPA algorithm with the maximum number of iterations in each simulation set to 100 and the maximum number of iterations in the decoding process of the GILD code is set to 20. It can be observed that, at a BER of 10^{-5}, the GILD code outperforms the regular LDPC code by about 0.25 dB and is less than 0.1 dB from the irregular one. The significance of this result should not be underestimated, considering that the GILD code is constructed using a random interleaver. It can be anticipated that if the code is carefully designed, using a deterministic interleaver that presents good properties in terms of iterative decoding, its performance can be further improved. Thus, variations of GILD codes may be able to match and even beat the best LDPC codes for small block lengths.

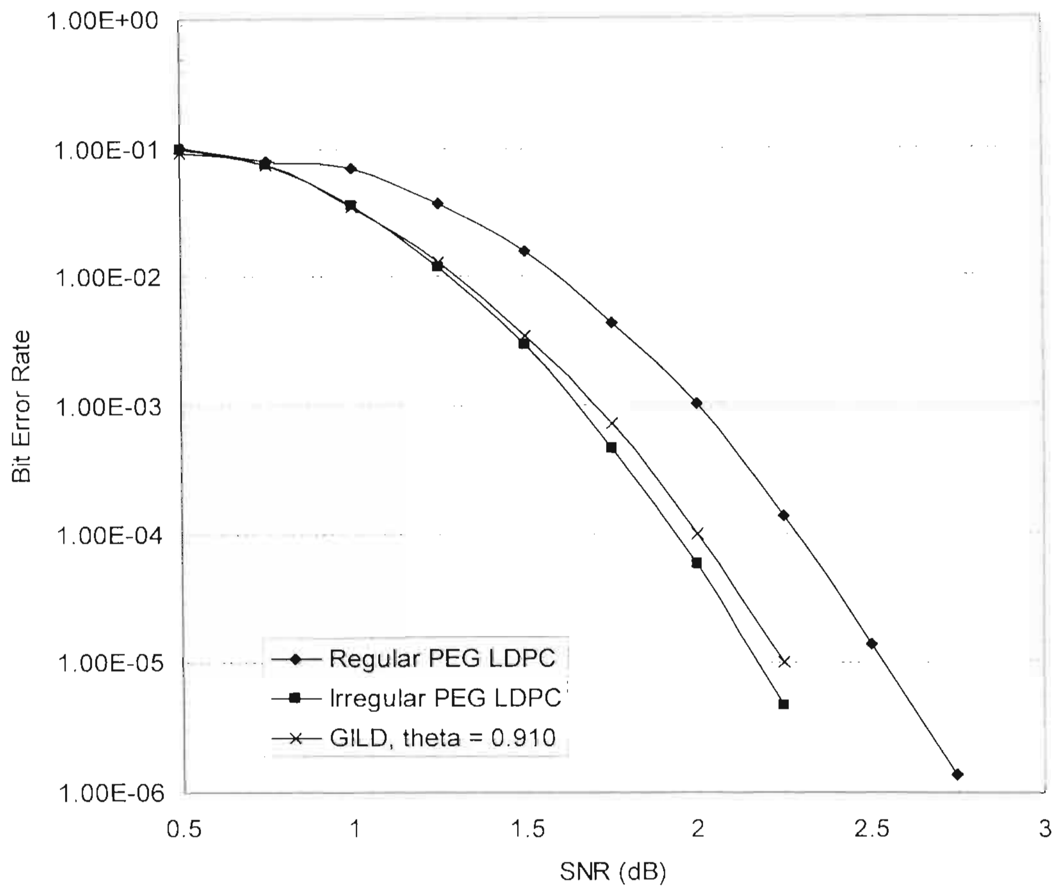

Figure 4.11: Simulated performance of a GILD code of length $N = 1005$ and rate 0.491 $(\theta = 0.910)$ built from the $(15,11,3)$ Hamming code and two LDPC codes of length $N = 1008$ and rate 0.5, AWGN channel.

In Figure 4.12, the performance of the first code with $\theta = 0.903$ is shown for different iteration steps, and compared to the various bounds described in Section 4.5. It can be seen that there is a loss in a large range of E_b/N_0 between the simulation results and the bounds. This loss is probably due to the interleaver choice and the decoding. The influences of the interleaver choice and the decoding algorithm on the performance of GILD codes are subject of further research.

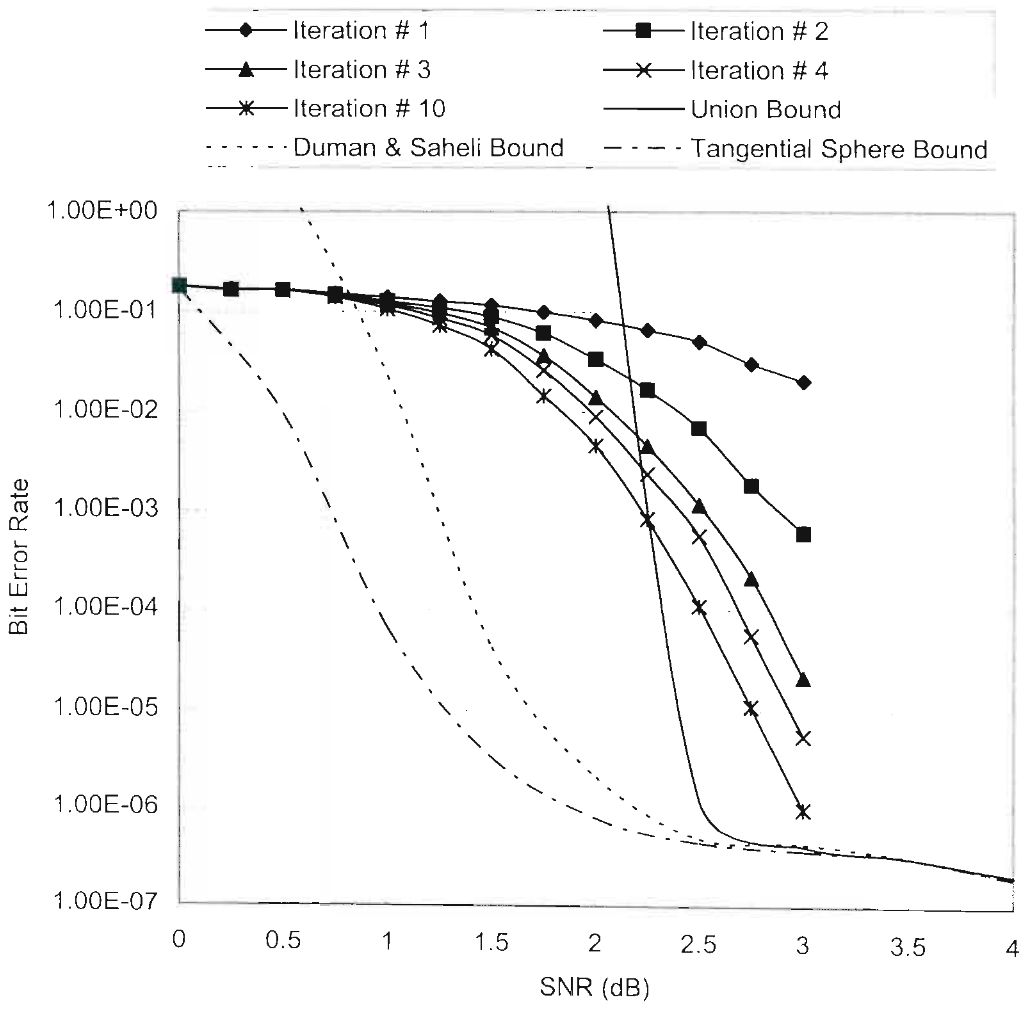

Figure 4.12: Bit Error Probability upper bounds and simulations results for the GILD code based on the (31, 21, 5) BCH component code. $\theta = 0.903$.

4.9 Conclusions

In this chapter, we have investigated the construction of new families of error-correcting codes based on irregular low-density parity-check codes which we called GILD codes, where small linear block codes are used as component codes instead of single-error detecting parity-check codes. It is proved that there exist such GILD codes for which the minimum Hamming distance is growing with the block length, and a lower bound of the minimum distance is given. It is shown that GILD codes performance approaches the channel capacity limit. The decoding of GILD codes is based on SISO decoding of the component code and the appropriate algorithm to achieve this depends on the length and dimension of the component code. Iterative decoding of GILD codes after transmission over an AWGN channel indicates they perform better than the corresponding GLD codes. GILD codes also have a higher rate than the corresponding GLD codes. On the other hand, like GLD codes, GILD codes have fewer problems with an error floor because of their large minimum distance. Simulation results also show that variations of GILD codes may be able to match or beat the best LDPC codes for small block lengths.

CHAPTER 5

A LOW-WEIGHT TRELLIS-BASED SOFT-INPUT SOFT-OUTPUT DECODING ALGORITHM FOR LINEAR BLOCK CODE

In this chapter, reduced-complexity trellis-based soft-input soft-output (SISO) decoding of linear block codes is considered. A new low-weight subtrellis based SISO decoding algorithm for linear block code to achieve near optimal error performance with a significant reduction in decoding complexity is presented. The proposed scheme is suitable for iterative decoding and has the following important features. An initial candidate codeword is first generated by a simple decoding method that guarantees a successful decoding. By successful decoding, we mean that the candidate codeword is indeed a codeword, but not necessarily the transmitted codeword. A low-weight subtrellis diagram centered around the candidate codeword is constructed. The MAP algorithm is then applied to the subtrellis. The generated extrinsic information is used as apriori information to improve the generation of a candidate codeword for the next stage of iteration. Simulation results indicate that the proposed algorithm provides a significant improvement in error performance over Chase-based algorithm and achieves practically optimal performance with a significant reduction in decoding complexity.

The organization of this chapter is as follows: in the next section an introduction is presented. The trellis representation of linear block codes is briefly discussed in Section 5.2. An algorithm for purging a minimal trellis diagram of a linear block code to yield a subtrellis diagram is presented in Section 5.3. In Section 5.4 a method to build a subtrellis diagram using supercodes is described. The SISO algorithm is presented in Section 5.5. Simulation results on the error performance of a GILD code are presented in Section 5.6, and finally, conclusions are drawn in Section 5.7.

5.1 Introduction

The problem of finding computationally efficient and practically implementable soft-input soft-output (SISO) decoding algorithms for block codes is still an open and challenging problem. Any SISO decoding of block codes using a full code trellis can be primitively complex. This is especially true if these codes are used as component codes in compound coding scheme such as GILD and GLD codes discussed in this thesis. One alternative approach is to use a list-based decoding algorithm as discussed in the next chapter. List-based decoding algorithms try to produce a list containing the closest codewords to the soft input by encoding or decoding test sequences and calculate extrinsic information using this list. However, this method cannot be extended effectively to all classes of codes. Moreover, it is sub-optimal.

To overcome the state and branch complexity problems of large trellises for long block codes, several new approaches have been proposed [146]-[151]. Most recently, Moorthy et *al.* [95] have shown that the minimum-weight subtrellis of a code is sparsely connected and has much simpler state and branch complexities than the full code trellis. Based on this fact, they proposed a minimum-weight subtrellis-based iterative decoding algorithm for linear block codes to achieve suboptimum error performance with a drastic reduction in decoding complexity compared with a trellis-based MLD algorithm, using a full code trellis. This algorithm was improved by Koumoto et *al.*[152].

In this chapter a new low-weight subtrellis based SISO decoding algorithm for linear block code to achieve near optimal error performance with a significant reduction in decoding complexity is presented. By low-weight subtrellis, we mean a subtrellis of the code trellis that consists of only codewords of low weights with respect to a given codeword. The proposed scheme is suitable for iterative decoding and has the following important features. An initial candidate codeword is first generated by a simple decoding method that guarantees a successful decoding. By successful decoding, we mean that the candidate codeword is indeed a codeword, but not necessarily the transmitted codeword. The decoding method is basically the syndrome computation of test patterns that are constructed based on the reliability of the soft-decision received symbols. In contrast to the algorithm in [95], no algebraic decoding algorithm is used. A low-weight subtrellis diagram centered around the candidate codeword is constructed. The subtrellis diagram is sparsely connected and much simpler than the full trellis diagram of the code. The MAP algorithm is then applied to the subtrellis. The generated extrinsic information is used as apriori information to improve the generation of a candidate codeword for the next stage of iteration. Instead of a theoretical analysis of the proposed algorithm some simulation results that indicate that it achieves practically optimal performance with a significant reduction in decoding complexity compared with the MAP based on the full trellis diagram of the code are presented.

5.2 Preliminaries

In this section the trellis representation of linear block codes is briefly discussed. In 1978, Wolf [128] showed that every (n,k) linear block code has an n-section trellis diagram. In recent years, there have been exciting developments in trellis structure of linear block codes [129] - [139]. Some well-known codes have been proved to have relatively simple trellis structures. A trellis is a directed graph $T=(S,W)$, where the set $S=\{\sigma\}$ of nodes (states) of the graph is decomposed into a union of $n+1$ disjoints subsets $S=S_0 \cup S_1 \cup \ldots \cup S_n$ that are called levels of the trellis. Similarly, there exists a partitioning of the sets of branches (edges) $W=\{w_t\}=W_1 \cup W_2 \cup \ldots \cup W_n$. A node $\sigma \in S_t$ of level t may be connected with a node $\sigma' \in S_{t+1}$ of the level $t+1$ by one or several branches. Each branch w_t is directed from a node σ of level t to a node σ' of the next level $t+1$. We assume that the end levels have only one node, namely $S_0=\{\sigma_0\}$ and $S_n=\{\sigma_n\}$. A trellis is a compact method to represent all codewords of a code. Each branch of the trellis w_t is labeled by a code symbol $v_t(w_t)$. Each distinct codeword corresponds to a distinct path in the trellis, i.e., there is a one-to-one correspondence between each codeword $\boldsymbol{c}$ in the code and a path $\boldsymbol{w}$ in the trellis: $\boldsymbol{c}(\boldsymbol{w})=c_1(w_1),\cdots,c_n(w_n)$.

For a linear code, a minimal trellis can be obtained using its parity-check matrix [128]. Such a trellis is called syndrome trellis. Let $\boldsymbol{H}=(\boldsymbol{h}_1,\cdots,\boldsymbol{h}_n)$ denote the parity-check matrix of C, where $\boldsymbol{h}_i$ denotes the ith column vector. The term minimal trellis indicates that this trellis has the minimal possible number $|S|$ of nodes. The nodes of the trellis are identified by $(n-k)$-tuples with elements from the binary field $\mathbb{F}_2 = \{0,1\}$, with $\mathbf{0}$ referring to the all zero $n-k$-tuple. At level $t=0$ and level $t=n$ the trellis contains one node, the all zero node $\sigma_0 = \sigma_n = \mathbf{0}$. The following algorithm describes a trellis construction which differs from that of [128], but results in the same graph.

- Step 1: construct the head of the trellis, i.e., all states and branches of the levels $t = 0,\cdots,\lfloor \frac{n-1}{2} \rfloor$. Set $\sigma_0 = \mathbf{0}$. For each $t = 1,\cdots,\lfloor \frac{n-1}{2} \rfloor$, the collection of nodes at level $t+1$ is obtained from the collection of nodes at level t by $\sigma_{t+1} = \sigma_t + c_t \boldsymbol{h}_t$ for all $\sigma_t \in S_t$ and $c_t \in \mathbb{F}_2$.
- Step 2: Construct the tail of the trellis, i.e., all states and branches of the levels $t = n, n-1,\cdots,\lfloor \frac{n+1}{2} \rfloor$. Set $\sigma_n = \mathbf{0}$. For each $t = n-1,\cdots,\lfloor \frac{n+1}{2} \rfloor$, the collection of nodes at level $t-1$ is obtained from the collection of nodes at level t by $\sigma_{t-1} = \sigma_t + c_{t-1} \boldsymbol{h}_{t-1}$ for all $\sigma_t \in S_t$ and $c_{t-1} \in \mathbb{F}_2$.
- Step 3: Connect the head and tail of the trellis. We connect the collection of nodes at level $t = \lfloor \frac{n-1}{2} \rfloor$ with the collection of nodes at level $t+1$ by $\sigma_{t+1} = \sigma_t + c_t \boldsymbol{h}_t$ for all $\sigma_t \in S_t$, $\sigma_{t+1} \in S_{t+1}$, and $c_t \in \mathbb{F}_2$.

5.3 Low-Weight Trellis Diagram

In the following, we considered, according to [95], purging the minimal n-section trellis diagram T for a linear block code C to obtain a subtrellis diagram, denoted $T_\rho(\mathbf{0})$, which consist of only the all-zero codeword and the codewords $\boldsymbol{c}$ of C such that $d_H(\boldsymbol{c},\mathbf{0}) \leq \rho$, where $d_H(,)$ is the Hamming distance of two vectors. Such a trellis is called the ρ-*weight trellis* diagram for C. We say that this ρ-weight trellis diagram is centered around the all-zero codeword $\mathbf{0}$.

For two adjacent states σ and σ', with $\sigma \in S_t$ and $\sigma' \in S_{t+1}$, let $l(\sigma,\sigma')$ denote the set of parallel branches connecting states σ and σ'. Let w_t be a branch in $l(\sigma,\sigma')$ and $\lambda(w_t)$ denote the Hamming weight of w_t. Let $\alpha\left(l(\sigma,\sigma')\right)$ denote the minimum hamming weight of the parallel branches in $l(\sigma,\sigma')$.

For any state $\sigma \in S_t$ with $0 \le t \le n$, let (σ) denote the minimum of the Hamming weights of all paths from σ_0 to σ. We call (σ) the *minimum path weight* to the state σ. Clearly, $(\sigma_0) = 0$. For every state $\sigma \in T$, (σ) can be computed by an algorithm very similar to the well-known Viterbi algorithm as follows: Assume that (σ) is known for every state in S_j for $0 \le j < t$. Let σ' be a state in S_t. Then (σ') is given by:

$$(\sigma') = \min_{F(\sigma')} \left\{ (\sigma) + \alpha\left(l(\sigma,\sigma')\right) \right\}, \tag{5.1}$$

where

$$F(\sigma') = \left\{ \sigma \in S_{t-1} : l(\sigma,\sigma') \ne \phi \right\}. \tag{5.2}$$

The process begins with σ_0. Once (σ) is determined for every state $\sigma \in S_t$, the states in the $(t+1)$ th-level state space $S_{(t+1)}$ can be processed. This computation is repeated until $(\sigma_n) = 0$ is determined.

For any state $\sigma \in S_t$ with $0 \le t \le n$, let $[\sigma]$ denote the minimum of the Hamming weights of all paths from σ to the final state σ_n. We call $[\sigma]$ the *minimum path weight from* σ. For every state σ in T, $[\sigma]$ can be computed recursively from σ_n as follows: assume that $[\sigma]$ is known for every state $\sigma \in S_j$ for $t < j \le n$. Let σ' be a state in S_t. Then $[\sigma']$ is given by:

$$[\sigma'] = \min_{G(\sigma')} \left\{ [\sigma] + \alpha\left(l(\sigma,\sigma')\right) \right\}, \tag{5.3}$$

with

$$G(\sigma') = \left\{ \sigma \in S_{(t+1)} : l(\sigma',\sigma) \ne \phi \right\}. \tag{5.4}$$

Note that among all the paths passing through a given state σ, there is a least one path with Hamming weight $(\sigma)+[\sigma]$ and no path passing through that state has Hamming weight less than $(\sigma)+[\sigma]$. Let the all-zero path be denoted by the state sequence $\sigma_0 = \sigma_0^0, \sigma_1^0, \cdots, \sigma_n^0 = \sigma_n$. It is clear that $(\sigma_t^0) = [\sigma_t^0] = 0$ for $0 \le t \le n$ and for any state σ not in this sequence, $(\sigma)+[\sigma] > 0$.

A systematic method of deleting states and branches in $\mathcal{T}$ to obtain a subtrellis which contains all the codewords with weight up to and including ρ where ρ is a nonzero weight in the weight profile ω of code C is now described. This method consists of the following rules [95].

- Rule 1: If for any state $\sigma \in \mathcal{T}$, $(\sigma)+[\sigma] > \rho$, delete that state and all the branches to and from σ.
- Rule 2: Let σ, σ' be any two connected states at depths t and $(t+1)$, $0 \le t \le n$, respectively. Let w_t be a single branch from σ to σ'. If $(\sigma)+[\sigma']+\lambda(w_t) > \rho$ then, delete the branch w_t.
- Rule 3: If as a result of application of Rules 1 and 2 to all the states and branches of $\mathcal{T}$ any state in $\mathcal{T}$ has no incoming branches, then that state and all the outgoing branches from that state are deleted. The above purging rules are applied repeatedly until further application leaves the trellis unaltered. Let $\mathcal{T}_\rho(\mathbf{0})$ denote the resultant purged trellis.

The subtrellis $\mathcal{T}_\rho(\mathbf{0})$ contains the following codewords of C: 1) All codewords of weights up to and including ρ and 2) possibly some codewords of weight greater than ρ which correspond to nonzero paths in $\mathcal{T}$ that diverges from and remerges with the all-zero path more than once. For each of these paths, the weight of the partial path between the adjacent diverging and merging states, called the *side-loop*, is ρ or less but not zero.

Now the subtrellis diagram $\mathcal{T}_\rho(\mathbf{0})$ is modified so that the resultant trellis contains only the all-zero path and all the paths which correspond to codewords of weight up to and including ρ of C (no other paths). This resultant trellis, denoted $\mathcal{T}_\rho(\mathbf{0})$, is called the ρ*-weight trellis diagram* for C. The modification of $\mathcal{T}_\rho(\mathbf{0})$ is done as follows:

- Step 1: Create $n-1$ new states $O_1, O_2, \cdots, O_{n-1}$ one at each level except the 0th and nth levels.
- Step 2: For $1 \le t < n-1$, connect the state O_t to state O_{t+1} by a single branch with the *0* label. Also, connect the state O_{n-1} to the final state σ_n by a single zero branch.
- Step 3: For every branch with label $w \neq 0$ in $\mathcal{T}_\rho(\mathbf{0})$ that merges into one of the states, σ_t^0 on the zero path $\sigma_1^0, \sigma_2^0, \cdots, \sigma_{t-1}^0$ from any state σ, delete that branch and create a new branch w' with the same label from σ to the state O_t.

Steps 1)-3) ensure that there is no path in the new trellis $\mathcal{T}_\rho(\mathbf{0})$ that after merging with the all-zero path diverges from it again before terminating at the final state σ_n. Consequently, $\mathcal{T}_\rho(\mathbf{0})$ contains only the all-zero codeword and the codewords of weight up to and including ρ of C. Therefore $\mathcal{T}_\rho(\mathbf{0})$ is the desired ρ*-weight trellis diagram* for C. Let $\mathcal{T}_\rho(\mathbf{0}) = (\overline{S}, \overline{W})$, where $\overline{S}$ is the set of active nodes and $\overline{W}$ is the set of active branches. By active nodes and active branches, we mean the nodes and branches which have not been deleted.

The ρ*-weight trellis diagram* for a linear block code C is sparsely connected and has much smaller state and branch complexities than those of the full trellis diagram for C. Table 5.1 shows the state complexity profile for the full trellis and two subtrellises of the (31, 21, 5) BCH code. For this code, the *minimum-weight trellis diagram* (ρ = 5) has a total of 1860 branches, and the 7-*weight trellis diagram* (ρ = 7) has a total of 6640 branches, whereas the full trellis has a total of 26620 branches. This reduction in number of states and branches contributes to a significant reduction in computational complexity.

Table 5.1: State complexity profile for the full trellis and two subtrellises of the (31, 21, 5) BCH code.

Trellis level	Full trellis	Subtrellis (ρ = 5)	Subtrellis (ρ = 7)
0	1	1	1
1	2	2	2
2	4	4	4
3	8	7	8
4	16	12	16
5	32	20	28
6	64	29	47
7	128	42	75
8	256	56	116
9	512	71	170
10	1024	86	238
11	1024	102	302
12	1024	115	362
13	1024	129	408
14	1024	136	408
15	1024	139	434
16	1024	139	434
17	1024	136	427
18	1024	129	408
19	1024	115	362
20	1024	102	302
21	1024	86	238
22	512	71	170
23	256	56	116
24	128	42	75
25	64	29	47
26	32	20	28
27	16	12	16
28	8	7	8
29	4	4	4
30	2	2	2
31	1	1	1

5.4 Low-Weight Trellis Diagram Using Supercodes

A low complexity approach to construct a low-weight trellis diagram is to use supercodes. A supercode [140] of a linear code C is a linear code C' such that $C \subset C'$.

For the sake of simplicity, we consider only the case where we have two supercodes. Let $\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \end{pmatrix}$ be the parity-check matrix of the code C, where the submatrix $\boldsymbol{H}_1$ consists of the upper $\lceil \frac{n-k}{2} \rceil$ row vectors of $\boldsymbol{H}$, and $\boldsymbol{H}_2$ consists of the lower $\lfloor \frac{n-k}{2} \rfloor$ row vectors. Then the sub-matrices $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ define the supercodes C_1 and C_2, respectively. Let $\bar{\mathcal{T}}_\rho^{(1)}(\mathbf{0})$ and $\bar{\mathcal{T}}_\rho^{(2)}(\mathbf{0})$ denote the ρ*-weight trellies diagram* of C_1 and C_2, respectively. That is, $\bar{\mathcal{T}}_\rho^{(1)}(\mathbf{0}) = \left(\bar{S}^{(1)}, \bar{W}^{(1)}\right)$ and $\bar{\mathcal{T}}_\rho^{(2)}(\mathbf{0}) = \left(\bar{S}^{(2)}, \bar{W}^{(2)}\right)$. The following algorithm constructs a trellis $\mathcal{T}_\rho(\mathbf{0})$ representing the set $\bar{\mathcal{T}}_\rho^{(1)}(\mathbf{0}) \cap \bar{\mathcal{T}}_\rho^{(2)}(\mathbf{0})$. A node in this trellis is represented by an $n-k$ tuple $\sigma_t = \left(\sigma_t^{(1)} \sigma_t^{(2)}\right)$, where $\sigma_t^{(1)}$ is an $\lceil \frac{n-k}{2} \rceil$ tuple and $\sigma_t^{(2)}$ is an $\lfloor \frac{n-k}{2} \rfloor$ tuple.

- Step 1: set $S_0 = \{\mathbf{0}\}$ and set $t = 1$.
- Step 2: the set S_t is constructed from the set S_{t-1}, the parity-check matrices $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$, and sets $\bar{S}_t^{(1)}$ and $\bar{S}_t^{(2)}$. The collection of nodes at level t is obtained from the collection of nodes at level $t-1$ by:

$$\sigma_t^{(1)} = \sigma_{t-1}^{(1)} + c_t \boldsymbol{h}_{1,t}$$

$$\sigma_t^{(2)} = \sigma_{t-1}^{(2)} + c_t \boldsymbol{h}_{2,t}$$

  for all $\sigma_{t-1} \in S_{t-1}$ and $c_t \in \mathbb{F}_2$ such that $\sigma_t^{(1)} \in \bar{S}_t^{(1)}$ and $\sigma_t^{(2)} \in \bar{S}_t^{(2)}$.
- Step 3: If $t < n$, then continue with the next level, i.e., increment t by one and go to step 2.
- Step 4: Remove all nodes (except σ_n) which have no outgoing branch, and remove the corresponding incoming branches.

5.5 A Low-Weight Soft-Input Soft-Output Decoding Algorithm

In this section, a low-weight soft-input soft-output decoding algorithm suitable for iterative decoding of linear block codes when they are used as component codes in a compound or concatenated coding scheme is described. Candidate codewords are generated one at a time based on syndrome computation of test sequences which are constructed in a manner similar to list-based decoding algorithm. To be more specific, error patterns are generated in a serial fashion in increasing order of bit reliability values. Thus the ith ($i = 0,1,2,\ldots,2^n - 1$, where n is the length of the codewords) error pattern corresponds to number i in the binary number system. The bit positions are arranged in the order of their reliabilities such that the least reliable bit is the least significant bit and so on. After generating an error pattern, the error pattern is added to the hard decision corresponding to the received vector, and syndrome computation is performed on the resultant test sequence. It is important to note that, contrary to list-based decoding algorithms, no algebraic decoding algorithm is used. If the syndrome of the test sequence is zero, then it is considered as the candidate codeword. Otherwise, the next test sequence is generated and tested using syndrome computation.

When a candidate codeword c is generated, a subtrellis diagram $\mathcal{T}_p(\boldsymbol{c})$ that contains only the codewords of weight up to and including ρ centered around $\mathbf{c}$ is constructed. First $\mathcal{T}_p(\mathbf{0})$ centered around the all-zero codeword $\mathbf{0}$ is constructed before the decoder implementation (or design). A method of constructing $\mathcal{T}_p(\mathbf{0})$ was presented in the previous section. $\mathcal{T}_p(\boldsymbol{c})$ is isomorphic to $\mathcal{T}_p(\mathbf{0})$ and is obtained by adding the candidate codeword $\boldsymbol{c}$ to every codeword in $\mathcal{T}_p(\mathbf{0})$. The MAP (maximum a posteriori probability) algorithm [40], also referred to as BCJR, is then applied to the subtrelis diagram $\mathcal{T}_p(\boldsymbol{c})$. The algorithm was summarized in Chapter 4. Other trellis-based SISO decoding algorithms, optimal or suboptimal, such as Log-MAP, Max-log-MAP [141] or even SOVA [142], can also be used. The generated extrinsic information is used a priori information to improve the generation of a candidate codeword for the next stage of iteration.

5.6 Simulation Results

The proposed low-weight subtrellis based SISO decoding algorithm has been applied to GILD codes discussed in the previous chapter. The GILD code of length $N = 961$, $\theta = 0.806$ based on the (31, 21, 5) BCH code was used. We assumed AWGN channel BPSK signaling. The BER performance of this code using the proposed algorithm is shown in Figure 5.1 for $\rho = 6$, and $\rho = 7$. It is observed that the performance improves with ρ, with diminishing return. No significant improvement in performance was observed for greater values of ρ. The performance of the proposed algorithm is now compared with the MAP algorithm applied to the full code trellis. For the proposed algorithm $\rho = 7$. The performance of both algorithms is shown in Figure 5.2. We see that the error performance of this code based on the proposed algorithm falls on top of its performance using the MAP algorithm applied to the full code trellis. Thus the proposed algorithm achieves practically optimal error performance with a significant reduction in computational complexity. In Figure 5.3 the BER performance of the proposed algorithm is compared to that of the Chased-based algorithm presented in the previous chapter, with each Chase decoder using a set of 16 test sequences. The results show that the proposed algorithm performs better than the Chase-based algorithm. At a BER of 10^{-5}, the improvement in performance is about 0.13 dB.

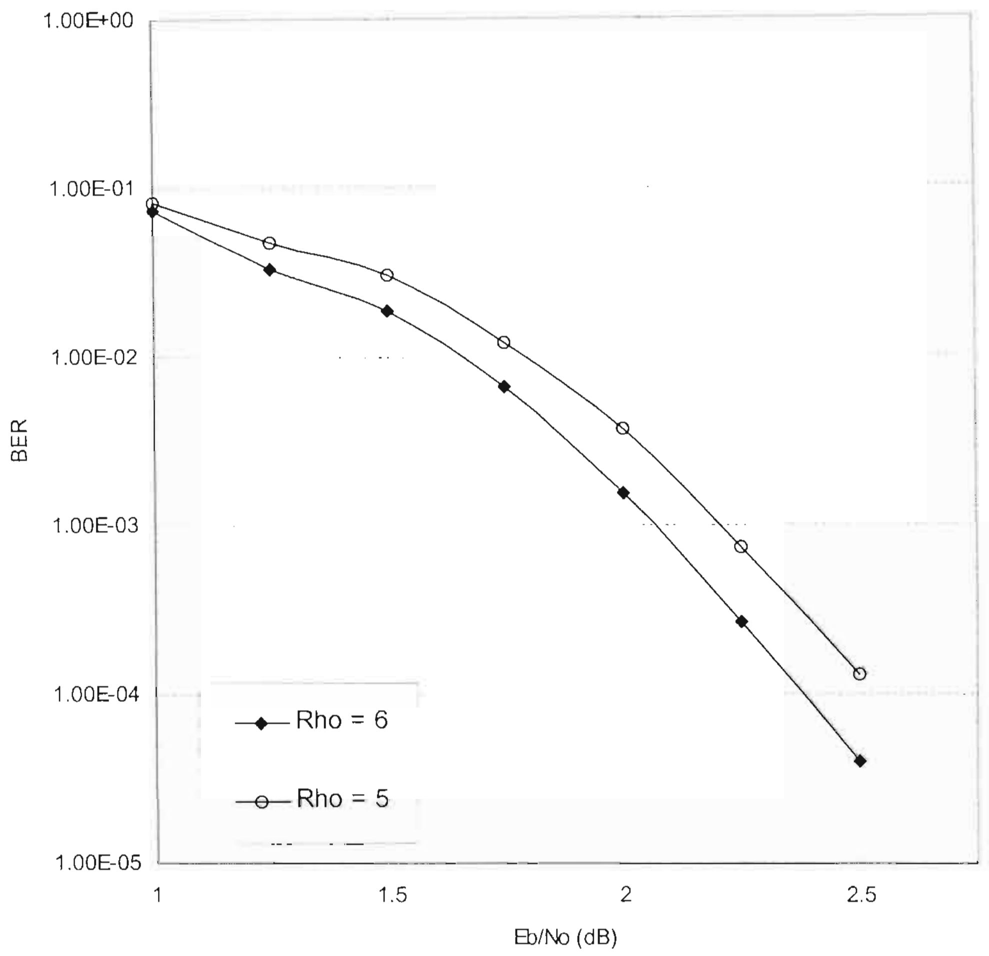

Figure 5.1: Simulated performance of the GILD code of length $N = 961$, $\theta = 0.806$, built from the (31, 21, 5) BCH component code, AWGN channel, 8 iterations.

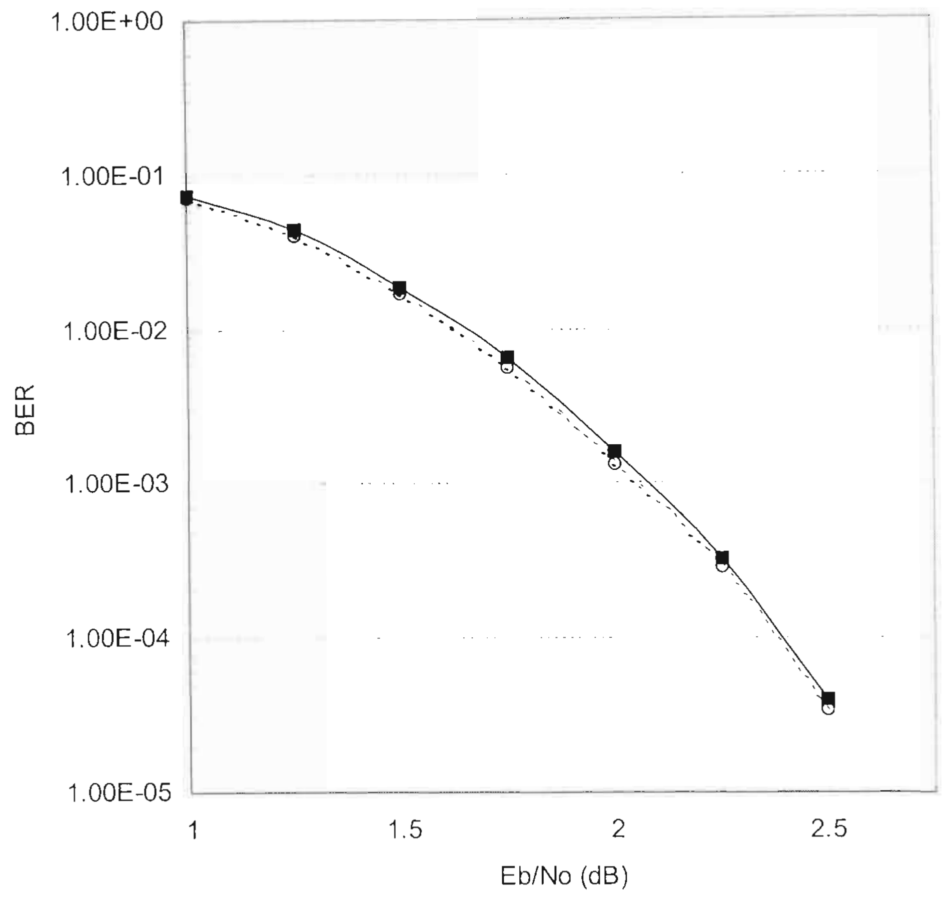

Figure 5.2: Performance of the simulated GILD using the proposed algorithm with ρ = 7 (solid) and the MAP algorithm applied to the full trellis (dashed) after 8 iterations in both cases. N = 961, θ = 0.806, built from the (31, 21, 5) BCH component code, AWGN channel.

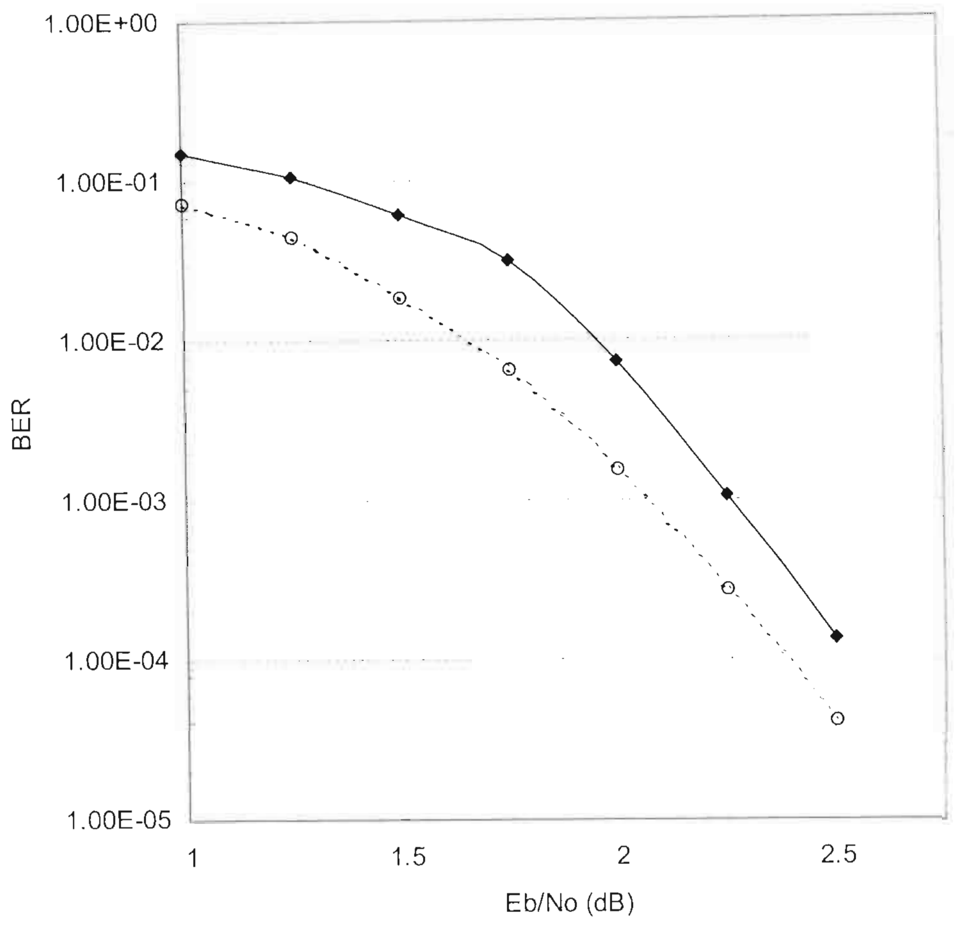

Figure 5.3: Performance of the simulated GILD code using the proposed algorithm (dashed) and the chase-based algorithm (solid) after 8 iterations in both cases. $N = 961$, $\theta = 0.806$, built from the (31, 21, 5) BCH component code, AWGN channel.

5.7 Conclusion

In this chapter, a soft-input soft-output decoding algorithm for binary linear block codes based on low-weight trellis to achieve practically optimum error performance with a significant reduction in decoding complexity has been presented. The algorithm is suitable for iterative decoding and can be applied to any compound/concatenated code based on linear block codes.

CHAPTER 6

HIGH-PERFORMANCE LOW-COMPLEXITY DECODING OF GENERALIZED LOW-DENSITY PARITY-CHECK CODES

In this chapter, an efficient list-based soft-input soft-output (SISO) decoding algorithm for compound codes based on linear block codes is presented. Attention is focused on GLD codes. The proposed algorithm modifies and utilizes the improved Kaneko's decoding algorithm for soft-input hard-output decoding. These hard outputs are converted to soft-decisions using reliability calculations. Compared to the trellis-based Maximum a Posteriori Probability (MAP) algorithm, the proposed algorithm suffers no degradation in performance at low bit-error rate (BER), but presents the major advantages of being applicable in cases where the trellis-based MAP algorithm would be prohibitively complex and impractical. Compared to the Chase-based algorithm of [85], [86], [88], [89] the proposed algorithm is more efficient, has lesser computational complexity for the same performance and provides an effective tradeoff between performance and computational complexity to facilitate its usage in practical applications.

This chapter is organized as follows: First, a brief introduction to motivate the problem is presented. Then a short review of GLD codes structure is given. The decoding algorithm is presented in Section 6.3. In Section 6.4 the stopping criteria proposed for preventing unnecessary computations and decoding delay are described. The simulation model and results are presented in Section 6.5 and, finally, conclusions are drawn in Section 6.6.

6.1 Introduction

GLD codes can effectively be decoded based on iterative SISO decoding of individual constituent codes, where the code performance and decoding complexity are heavily dependent on the employed SISO decoding algorithm. For low rate GLD codes, exploiting the fact that the constituent code usually has a small code length and high code rate, a trellis-based algorithm can be used to obtain high error-correcting performance with reasonable decoding complexity. In [55], the Johansson-Zigangirov A *Posteriori* Probability (APP) algorithm [90], which operates with one forward recursion through a syndrome trellis was used, whereas the MAP algorithm [40], also referred to as BCJR, was used in [56] and [66]. However, if long and powerful constituent codes are used, these trellis-based solutions become prohibitively complex and impractical as the trellis complexity grows exponentially with the dimension of any sequence of good codes [79]. In these cases sub-optimal decoding algorithms are required. There are a variety of sub-optimal soft-decision decoding algorithms available to decode block codes, including Chase's algorithm [80], the generalized minimum distance (GMD) decoding algorithm [81], the order-i reprocessing [82], the Kaneko algorithm [83], and the improved Kaneko algorithm [91]. If these soft-input hard-output (SIHO) algorithms are to be used in an iterative decoding context they must produce soft information. One technique for doing this is discussed in [84] and another in [85], [88], [89]. In particular, the Chase algorithm and the Kaneko algorithm have been considered for the decoding of block turbo codes (BTCs) in [85] and [92] respectively.

In this chapter, an efficient SISO iterative decoding algorithm for GLD codes is proposed. The algorithm modifies and utilizes the improved Kaneko's decoding algorithm for SIHO decoding. Similar to the decoding strategy presented in [85], [88], [89], [92], a list of codewords containing the closest codewords to the soft input is generated by decoding test sequences using a simple algebraic decoder. The codeword that is most likely is selected from the list, followed by reliability calculations to obtain soft-decisions based on the remaining codewords in the list. To avoid algebraic decoding resulting in a codeword already found, a simple test is proposed before decoding of each test sequence. Each time when a new test sequence is generated, it is tested. If the test holds, then this test sequence cannot produce a candidate codeword different from the candidate codewords already generated and hence it is useless. In this case, the test sequence is ruled out for decoding and the next test sequence is generated unless the test error patterns have been exhausted. This preprocessing test is very effective since its complexity is considerably smaller than that of an algebraic decoding operation. This procedure avoid useless decoding iterations and hence reduces the complexity of the algorithm when compare to those used in [85], [88], [89], [92]. To improve the average decoding speed of the GLD decoder, two simple criteria for stopping the iterative process for each frame immediately after the bits can be reliably decoded with no further iterations are proposed. For each decoded frame, the number of iterations performed is determined by the number of passes before a certain condition or rule for stopping is satisfied. The stopping conditions are hard-decision rules and are computed based on the data available to the decoder at each iteration during the decoding of each specific frame. Thus they require no extra data storage. More explicitly, at the end of each iteration, the decoder performs a check on the condition for stopping. If the condition is true, the iterative process on the frame is terminated, and the decoded sequence from the current iteration is sent to the output; otherwise, the iterative process continues to the next iteration. To prevent an endless loop should the stopping rule never be satisfied, it is required that the decoder cease after the maximum number of iterations, M.

6.2 Structure of GLD Codes

In the following the construction of GLD codes according to [56] is briefly described. As a generalization of LDPC codes, GLD codes are also defined by a sparse parity check matrix $\boldsymbol{H}$, constructed by replacing each row in LDPC parity check matrix with $(n-k)$ rows including one copy of the parity check matrix $\boldsymbol{H_o}$ of the constituent code $C_o(n,k)$, of a k-dimensional linear code of length n. The structure of the GLD parity check matrix $\boldsymbol{H}$ is depicted in Figure 6.1. We divide $\boldsymbol{H}$ into J submatrices, $\boldsymbol{H^1}, \ldots, \boldsymbol{H^J}$, each containing a single column of constituent parity check matrix $\boldsymbol{H_o}$ in each column. $\boldsymbol{H^1}$ is a *block diagonal* matrix and produces the direct sum of N/n constituent codes, where N is the GLD code length. All other submatrices are constructed as:

$\boldsymbol{H}^{j} = \pi_{j-1}\left(\boldsymbol{H}^{1}\right)$ for $j = 2, \ldots, J$, where π_{j-1} represents a random column permutation. A (N, J, n) GLD code C can be considered as the intersection of J *super-codes* $C^{1}, \ldots, C^{j}$, whose parity check matrices are the J submatrices, $\boldsymbol{H}^{1}, \ldots, \boldsymbol{H}^{J}$, respectively. If $C_{0}(n,k)$ has a rate $r = k/n$ and the parity check matrix $\boldsymbol{H}$ has full rank, the total rate of the GLD code is:

$$R = 1 - J(1-r). \tag{6.1}$$

In case of smaller rank, the rate is increased accordingly. However, with increasing block length N, (6.1) gives a good approximation of the actual rate of the code.

It has been shown that binary GLD codes with only $J = 2$ levels are asymptotically good [55], [56], [66]. Furthermore, GLD codes with 2 levels have the highest code rate and simple decoder structure. Thus in this work, only the decoding of $(N, 2, n)$ binary GLD codes based on primitive binary BCH codes is considered. For practical applications, efficient GLD codes can be built from primitive shortened or extended binary BCH codes.

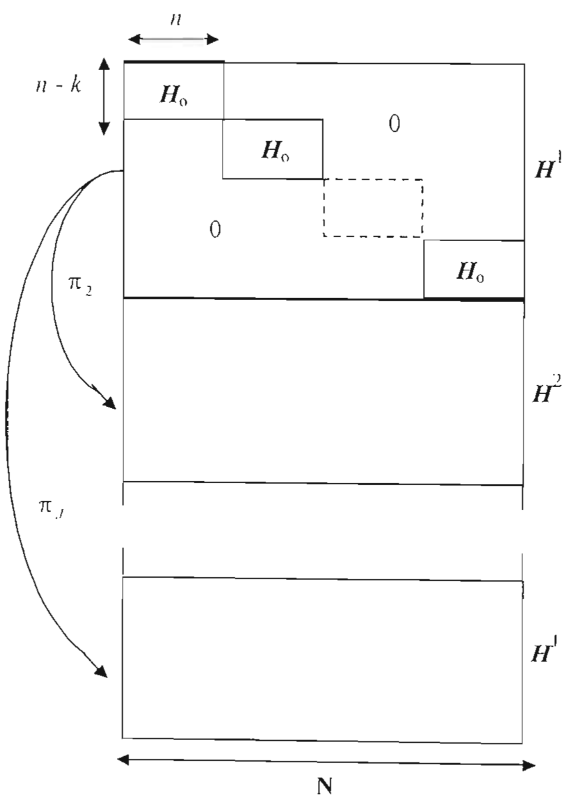

Figure 6.1: Structure of a GLD parity-check matrix.

6.3 Proposed Algorithm

In the proposed algorithm, each component code is decoded using the improved Kaneko algorithm [91], followed by reliability calculation to convert the hard-output of the decoder to a soft-output.

The improved Kaneko algorithm (IKA), which is summarized below, is a candidate codeword generation algorithm. The basic concept is to reduce the average complexity by selecting input vectors for hard-decision bounded-distance decoding so that the resultant codewords are likely to be transmitted. This concept is quite similar to that of Chase algorithms [80] except that the complexity of Chase algorithms is fixed while its complexity is probabilistic.

6.3.1 Soft Decision Decoding of Linear Block Codes Using the Improved Kaneko Algorithm

Suppose a binary linear block code C of length n, message length k, and minimum distance d, denoted as an (n, k, d) code is used for error control over the additive white Gaussian (AWGN) channel using binary phase-shift keying (BPSK) signalling. Let $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ be the transmitted codeword which is the output of the encoder, where c_i is an element of GF(2). For BPSK transmission, $\boldsymbol{c}$ is mapped into a bipolar sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ with $x_i = S(c_i) \in \{\pm 1\}$ for $1 \leq i \leq n$, where S is a function defined as:

$$S(c_i) = \begin{cases} -1, & c_i = 0 \\ +1, & c_i = 1 \end{cases}. \tag{6.2}$$

Suppose $\boldsymbol{x}$ is transmitted; at the receiver, the demodulator generates the reliability sequence $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ from the received sequence $y = (y_1, y_2, \ldots, y_n)$, where y_i is the received signal when x_i is transmitted. The reliability of the component y_i is defined using the log-likelihood ratio (LLR) of the decision y_i as:

$$\Lambda(y_i) = \alpha_i = \ln\left(\frac{\text{pr}\{(x_i = +1)/(y_i)\}}{\text{pr}\{(x_i = +1)/(y_i)\}}\right) = \left(\frac{2}{\sigma^2}\right) y_i . \tag{6.3}$$

The received vector $\boldsymbol{y}$ at the output of the AWGN channel is given by:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{g} , \tag{6.4}$$

where components g_i of $\boldsymbol{g} = (g_1, g_2, \ldots, g_n)$ are AWGN samples of standard deviation σ. Let $\boldsymbol{y}^H = (y_1^H, y_2^H, \ldots, y_n^H)$ be the binary hard decision received sequence obtained from $\boldsymbol{y}$ using the hard decision function given by:

$$y_i^H = \begin{cases} 0, & \alpha_i \le 0 \\ 1, & \alpha_i > 0 \end{cases} \qquad i = 1,2,\ldots,n. \tag{6.5}$$

We refer to $|\alpha_i|$ as the reliability of y_i^H. A larger $|\alpha_i|$ implies that the hard decision y_i^H is more reliable. We use $|y_i|$ as the reliability measure of the received symbol y_i since this value is proportional to the log-likelihood ratio associated with the symbol hard-decision. It is known that the maximal-likelihood decoder selects $\boldsymbol{c}_m$ as an estimate if $S(\boldsymbol{c}_m)\cdot\boldsymbol{\alpha} \ge S(\boldsymbol{c}_{m'})\cdot\boldsymbol{\alpha}$ for all $m' \ne m$ [93], [94]. The objective becomes to find a codeword maximizing the dot product $S(\boldsymbol{c}_m)\cdot\boldsymbol{\alpha}$.

For some positive integer $t \le \lfloor (d-1)/2 \rfloor$, a hard-decision bounded-distance decoder decodes $\boldsymbol{y}^H$ to a codeword $\boldsymbol{c}$ satisfying $d_H(\boldsymbol{y}^H, \boldsymbol{c}) \le t$ if it exist, where $\lfloor x \rfloor$ denotes the largest integer not greater than x, and $d_H(\boldsymbol{a}, \boldsymbol{b})$, the Hamming distance between $\boldsymbol{a}$ and $\boldsymbol{b}$. If no codeword $\boldsymbol{c}$ satisfying $d_H(\boldsymbol{y}^H, \boldsymbol{c}) \le t$ exist, decoding failure occurs. The positive integer t is the error-correcting capability of the code with hard-decision bounded-distance decoding. In the following, bounded-distance decoding always means hard-decision bounded-distance decoding and it is assumed that there exists an efficient bounded-distance decoding for C with error correcting capability t. Furthermore, for convenience, it is assumed that components of all vectors are numbered in the order of increasing reliability, so that if $i \le j$, $|\alpha_i| \le |\alpha_j|$.

Let $\{e^{(r)}\}$, $r = 0,1,2,\cdots, e^{(r)} = \left(e_1^{(r)}, e_2^{(r)}, \cdots, e_n^{(r)}\right)$ be a sequence of binary vectors of length n. Let f_1 and f_2 be two functions which transfer $e^{(r)}$ to $e^{(r+1)}$ as follows:

$$f_1\left(e^{(r)}\right) = e^{(r+1)},\ e_i^{(r+1)} = \begin{cases} 0, & i < \lambda_1 \\ 1, & i = \lambda_1 \\ e_i^{(r)} & i > \lambda_1 \end{cases}, \tag{6.6}$$

and

$$f_2\left(e^{(r)}\right) = e^{(r+1)},\ e_i^{(r+1)} = \begin{cases} 0, & i < \lambda_2 \\ 1, & i = \lambda_2 \\ e_i^{(r)} & i > \lambda_2 \end{cases}, \tag{6.7}$$

where

$$\lambda_1 = \min\left(i \mid e_i^{(r)} = 0,\quad 1 \le i \le n-t\right), \tag{6.8}$$

and

$$\lambda_2 = \min\left(i \mid e_i^{(r)} = 0, \text{ and } e_{i-1}^{(r)} = 1,\quad 2 \le i \le n-t\right). \tag{6.9}$$

Note that f_1 is a simple binary counting function.

For each $e^{(r)}$, $0 \le r < 2^{n-t}$, define a vector

$$\tilde{e}^{(r)} = \left(\tilde{e}_1^{(r)}, \tilde{e}_2^{(r)}, \cdots, \tilde{e}_n^{(r)}\right) \tag{6.10}$$

such that

$$\tilde{e}_i^{(r)} = \begin{cases} e_i^{(r)}, & i < \mu \\ 1, & \mu \le i < \mu + t \\ 0, & i \ge \mu + t \end{cases}, \tag{6.11}$$

where

$$\mu = \max\left(i \mid e_i^{(r)} = 1,\ 1 \le i \le n-t\right) \tag{6.12}$$

for $r \ge 1$ and $\mu = 0$ for $r = 0$.

Let $c^{(r)}$ be the output of the bounded-distance decoder when $y^H \oplus e^{(r)}$ is an input vector. If the output of the bounded-distance decoder is not a codeword (for example, when decoding failure occurs), $c^{(r)}$ is not defined. Then define Δ_r by:

$$\Delta_r = \max_{0 \le i \le r} S\left(c^{(i)}\right) \cdot \alpha . \tag{6.13}$$

The improved Kaneko algorithm is described as follows, where $w_H(a)$ denotes the Hamming weight of a and $\hat{c}$ the estimation of the transmitted codeword, respectively.

```
1. Initialization: set $r = 0$, $\Delta_{-1} = -\infty$.
2. Repeat the following procedure in order $r = 0, 1, 2, \cdots$:
   If $\Delta_{r-1} < S(y^H \oplus \tilde{e}^{(r)}) \cdot \alpha$, then
      Update $\Delta_r$ and $\hat{c} = c^{(r)}$ if $\Delta_r > \Delta_{r-1}$.
      Set $e^{(r+1)} = f_1(e^{(r)})$.
   If $\Delta_{r-1} \ge S(y^H \oplus \tilde{e}^{(r)}) \cdot \alpha$, then
      If $w_H(e^{(r)}) = 1$, then halt.
      Otherwise, set $e^{(r+1)} = f_2(e^{(r)})$.
```

The main difference between this algorithm and the one in [83] is the use of the functions f_1 and f_2 to generate input vectors for hard-decision bounded-distance decoding. The algorithm in [83] uses only f_1. The function f_2 avoids trial by which the bounded-distance decoder never outputs a codeword maximizing the likelihood function.

Example 1: Suppose code C is the (15, 7, 5) Bose-Chaudhuri-Hocquengham (BCH) code, whose generator polynomial is $g(x) = x^7 + x^4 + x^2 + x + 1$ over GF(2). We now decode y whose reliability vector, without reordering components, is $\alpha' =$ (-1.82, -1.26, -0.08, -1.24, -0.70, -1.42, -0.70, -1.42, -0.54, -0.40, -0.36, -1.66, 0.24, -2.02, -0.32, 1.04, -0.48). The reliability vector, whose components are numbered in the order of increasing reliability, is $\alpha =$(-0.08, 0.24, -0.32, -0.36, -0.40, -0.48, -0.54, -0.70, 1.04, -1.24, -1.26, -1.42, -1.66, -1.82, -2.02) and the hard-decision vector of y is $y^H =$ (010000001000000). The process of the decoding algorithm with a bounded-distance decoder with error-correcting capability $t = 2$ is presented in Table 6.1. In this example, the bounded-distance decoding is applied seven times and the number of e's generated during the decoding is 12.

Table 6.1: An example of decoding procedure for (15, 7, 5) BCH code using the IKA.

r	$e^{(r)}$	$S(y^H \oplus \tilde{e}^{(r)}).a$	Δ_{r-1}	$x^{(r)}$	$S(x^{(r)}).a$	function
0	000000000000000	12.94	$-\infty$	000000000000000	11.02	f_1
1	100000000000000	12.30	11.02	failure	—	f_1
2	010000000000000	11.74	11.02	000000000000000	11.02	f_1
3	110000000000000	11.58	11.02	000000000000000	11.02	f_1
4	001000000000000	11.42	11.02	failure	—	f_1
5	101000000000000	11.26	11.02	111100001001000	9.22	f_1
6	011000000000000	10.94	11.02	—		f_2
7	000100000000000	11.10	11.02	010111001000000	11.10	f_1
8	100100000000000	10.94	11.10	—		f_2
9	010100000000000	10.62	11.10	—		f_2
10	001100000000000	10.46	11.10	—		f_2
11	000010000000000	10.74	11.10	halt		

6.3.2 Test Error Patterns

A key element in the above algorithm is a set of test error patterns and their efficient utilization in the generation of candidate codewords through a simple algebraic decoding. The choice of the test error patterns determines the performance and effectiveness of the above algorithm. It is obvious that only the most probable error patterns should be used as the test error patterns. Moreover, if many test error patterns generate the same codeword, only one should be used to decrease the time complexity of the algorithm.

Again, Let $c^{(r)}$ be the output of the bounded-distance decoder when $y^H \oplus e^{(r)}$ is an input vector. If the output of the bounded-distance decoder is not a codeword (for example, when decoding failure occurs), $c^{(r)}$ is not defined. A test error pattern $e^{(r)}$ is said to be decodable if $c^{(r)}$ is defined. Two distinct error patterns, $e^{(r)}$ and $e^{(j)}$, are said to be equivalent if they are both decodable and if $c^{(r)} = c^{(j)}$. Let $e^{(r)}$ be a decodable error pattern and let $Q(e^{(r)})$ denote the set of all test error patterns, which are equivalent to $e^{(r)}$. Since all the test error patterns in an equivalence class generate the same candidate codeword, only one should be used. How to partition the set of error patterns into equivalent classes affect the efficiency of any decoding algorithm that utilizes test pattern set such as the one described above and those in [80], [81], and [83]. Lemma 6.1 characterizes a simple test, which can be performed before the decoding of each test sequence to determine whether the associated test error pattern is equivalent to a test error pattern already used.

Lemma 6.1: For some r such that $r > 0$ and some j such that $0 \le j < r$, assuming that the error patterns $e^{(r)}$ and $e^{(j)}$ are both decodable, i.e. $c^{(r)}$ and $c^{(j)}$ are both defined, then $e^{(r)}$ and $e^{(j)}$ are equivalent if and only if $d_H\left(y^{(r)} \oplus e^{(r)}, c^{(j)}\right) \le t$.

Proof: First, we show that if $e^{(r)}$ and $e^{(j)}$ are equivalent, then $d_H\left(y^{(r)} \oplus e^{(r)}, c^{(j)}\right) \le t$.

Since

$$\begin{aligned} d_H\left(y^{(r)} \oplus e^{(r)},\ c^{(j)}\right) &\le d_H\left(y^{(r)} \oplus e^{(r)},\ c^{(r)}\right) + d_H\left(c^{(r)},\ c^{(j)}\right) \\ &\le t + d_H\left(c^{(r)},\ c^{(j)}\right) \\ &\le t\,. \end{aligned} \tag{6.14}$$

Suppose

$$t < d_H\left(y^{(r)} \oplus e^{(r)},\ c^{(j)}\right), \tag{6.15}$$

equations (6.14) and (6.15) would then imply that $t < t$. This contradiction shows that $d_H\left(y^{(r)} \oplus e^{(r)}, c^{(j)}\right) \le t$ whenever $e^{(r)}$ and $e^{(j)}$ are equivalent. This proves the first part of the Lemma.

Next, we show that if the condition $d_H\left(y^{(r)} \oplus e^{(r)}, c^{(j)}\right) \le t$ holds, then $e^{(r)}$ and $e^{(j)}$ are equivalent.

$$d_H\left(y^{(r)} \oplus e^{(r)},\ c^{(j)}\right) \le d_H\left(y^{(r)} \oplus e^{(r)},\ c^{(r)}\right) + d_H\left(c^{(r)},\ c^{(j)}\right)$$

$$\le t + d_H\left(c^{(r)},\ c^{(j)}\right). \qquad (6.16)$$

Suppose $e^{(r)}$ and $e^{(j)}$ are not equivalent, i.e. $c^{(r)}$ and $c^{(j)}$ are different. We show in Appendix B that

$$d_H\left(c^{(r)},\ c^{(j)}\right) \le d_H\left(e^{(r)}, e^{(j)}\right) + 2t. \qquad (6.17)$$

Hence, (6.17) becomes:

$$d_H\left(y^{(r)} \oplus e^{(r)},\ c^{(j)}\right) \le d_H\left(e^{(r)},\ e^{(j)}\right) + 3t, \qquad (6.18)$$

contradicting the fact that the bounded distance decoder can correct at most t errors. This contradiction shows that whenever $d_H\left(y^{(r)} \oplus e^{(r)}, c^{(j)}\right) \le t$ holds, $e^{(r)}$ and $e^{(j)}$ are equivalent. This proves the second part of the Lemma.

Assume that $y^H \oplus e^{(r)}$ is decoded by bounded-distance decoding in the order $r = 0, 1, 2, \cdots$, then Lemma 6.1 provides a simple preprocessing test which can be used to avoid bounded-distance decoding resulting in a codeword already found. Prior to the decoding of each test sequence, it is required to check the Hamming distances between $y^H \oplus e^{(r)}$ and the codewords already found. If any of these Hamming distances is less than or equal to t, then bounded-distance decoding of $y^H \oplus e^{(r)}$ is not needed. This procedure decreases the number of times to apply bounded-distance decoding and hence the decoding delay of the algorithm. We will see that it improves the performance of the iterative decoding algorithm as well. Note that this test for eliminating useless test error patterns requires considerably less computational complexity than that of the bounded distance-t decoding. A different procedure for generating test error patterns is also presented in [95].

In Table 6.2, the process of the decoding algorithm with the above preprocessing test applied to example 1 is presented. It can be noticed that the bounded-distance decoding is applied five times instead of seven, thus reducing the time complexity of the decoding algorithm.

Table 6.2: An example of decoding procedure for (15, 7, 5) BCH code of example 3.1 using the IKA with the preprocessing test.

r	$e^{(r)}$	$S(y^H \oplus \tilde{e}^{(r)}).a$	Δ_{r-1}	$x^{(r)}$	$S(x^{(r)}).a$	function
0	000000000000000	12.94	$-\infty$	000000000000000	11.02	f_1
1	100000000000000	12.30	11.02	failure	—	f_1
2	010000000000000	11.74	11.02	—	11.02	f_1
3	110000000000000	11.58	11.02	—	11.02	f_1
4	001000000000000	11.42	11.02	failure	—	f_1
5	101000000000000	11.26	11.02	111100001001000	9.22	f_1
6	011000000000000	10.94	11.02	—		f_2
7	000100000000000	11.10	11.02	010111001000000	11.10	f_1
8	100100000000000	10.94	11.10	—		f_2
9	010100000000000	10.62	11.10	—		f_2
10	001100000000000	10.46	11.10	—		f_2
11	000010000000000	10.74	11.10	halt		

6.3.3 Soft-Input Soft-Output Decoding of Linear Block Codes Using the Improved Kaneko algorithm

Once a list of codewords containing the closest codewords to the received vector is generated by the decoder, the closest codeword in the Euclidean distance sense to this vector is selected as the decision. Let $\boldsymbol{d} = (d_1, d_2, \cdots, d_n)$ be the decoded codeword. After $\boldsymbol{d}$ is obtained, the reliability of each of the components of vector $\boldsymbol{d}$ is calculated to generate soft-decisions at the output of the decoder. The soft-output is required so that the extrinsic information generated can be sent to the next decoder in an iterative decoding scheme. The reliability of the decision d_j (jth bit of $\boldsymbol{d}$) is calculated taking into account the fact that $\boldsymbol{d}$ is one of the codeword of C and is defined as the LLR of the transmitted symbol x_j (jth bit of $\boldsymbol{x}$) which is given by:

$$\Lambda(d_j) = \ln\left(\frac{\mathrm{pr}\{(x_j = +1)/y\}}{\mathrm{pr}\{(x_j = -1)/y\}}\right). \tag{6.19}$$

Applying Bayes' rule, and assuming that the different codewords are uniformly distributed, the LLR can be expressed as:

$$\Lambda(d_j) = \ln\left(\frac{\sum_{c' \in B_j^{+1}} \mathrm{p}\{\boldsymbol{y}/(\boldsymbol{x} = S(\boldsymbol{c}'))\}}{\sum_{c' \in B_j^{-1}} \mathrm{p}\{\boldsymbol{y}/(\boldsymbol{x} = S(\boldsymbol{c}'))\}}\right), \tag{6.20}$$

where B_j^{+1} is the set of codewords $\{\boldsymbol{c}'\}$ such that $S(c_j') = +1$, B_j^{-1} is the set of codewords $\{\boldsymbol{c}'\}$ such that $S(c_j') = -1$, and $p\{\boldsymbol{y}/(\boldsymbol{x} = S(\boldsymbol{c}'))\}$ is the probability density function of $\boldsymbol{y}$ conditioned on $\boldsymbol{x}$ and is defined by the particular channel in consideration. For an AWGN channel $p\{\boldsymbol{y}/(\boldsymbol{x} = S(\boldsymbol{c}'))\}$ is given by:

$$p\{\boldsymbol{y}/(\boldsymbol{x} = S(\boldsymbol{c}'))\} = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{|\boldsymbol{y} - S(\boldsymbol{c}')|^2}{2\sigma^2}\right). \tag{6.21}$$

The LLR gives the soft-output for every decision and can be used for calculating the extrinsic information and hence, the soft-input for the next stage of iteration. The calculation of the LLR using (6.20) can be very tedious and often impractical. Thus, some level of approximation can be introduced in the reliability calculations. The sub-optimal decoder approximates this by considering only the candidate codewords generated.

Now we assume equiprobable codewords and normalize the soft-output by dividing through by $2/\sigma^2$. Using the approximation $\log(e^x + e^y) \approx \max(x, y)$ of [96] and some algebra, the soft-output can be approximated by:

$$\Lambda'(\boldsymbol{d}_j) = \left(\frac{|\boldsymbol{y} - S(\boldsymbol{c}_c)|^2 - |\boldsymbol{y} - S(\boldsymbol{d})|^2}{4}\right) S(d_j) \approx \Lambda(d_j), \tag{6.22}$$

where $\mathbf{c}_c$ is the concurrent codeword defined as the codeword closest to the received vector given that $d_j \neq c_j$. With some algebra the soft output of (6.22) can be expressed as:

$$\Lambda'(d_j) = S(d_j)\sum_{l=1}^{n}\frac{y_l}{2}\left(S(d_l)-S(c_l)\right) = S(d_j)\sum_{l=1,d_l\neq c_l}^{n} S(d_l)y_l \,. \tag{6.23}$$

The extrinsic information gained by the decoder for the *j*th bit is defined as:

$$\begin{aligned} w_j &= \Lambda'_j - y_j \\ &= S(d_j)\sum_{l=1,l\neq j,d_l\neq c_l}^{n} S(d_l)y_l \\ &= S(d_j)\beta_j \end{aligned} \tag{6.24}$$

where

$$\beta_j = \sum_{l=1,l\neq j,d_l\neq c_l}^{n} S(d_l)y_l \,. \tag{6.25}$$

For $1 \leq j \leq n$, and each position j, the value w_j can be sent to the next decoder in an iterative decoding scheme as extrinsic LLR, with a scaling factor α_c, so that

$$y'_j = y_j + \alpha_c w_j \tag{6.26}$$

is computed as the soft-input to the next decoder. The factor α_c is used to compensate for the difference in the variance of the Gaussian random variables y_j and y'_j.

Equation (6.25) is the theoretical value for β_j and is used when the decoder has found a competing codeword for the *j*th position ($d_j \neq c_j$). This will be called case 1. When a competing codeword, $\mathbf{c}_c$, is not available an approximation to β_j is required. Two main cases need to be considered as follows:

- The decoder has found at least 2 distinct codewords, but no competing codeword is produced with $d_j \neq c_j$. This will be called case 2.
- Only one codeword is found by the decoder, so there are no competing codewords for any positions in the current received component codeword. This will be called case 3.

In case 2, β_j is calculated using (6.25) for some positions in the codeword. It is important not to overestimate the extrinsic information when using iterative decoding as it may cause significantly slower average convergence in the decoding. So the following conservative approximation to (6.25) is used in case 2 [86], [89]

$$\beta_j = \min_l (\beta_l > 0), \tag{6.27}$$

where the minimum is over all positive values of β_j calculated using (6.25).

In case 3, (6.25) cannot be used to calculate β_j for any position in the decoded codeword $\boldsymbol{d}$. Since it is important not to overestimate the extrinsic information, an average value can be used. We use the mean of the absolute value of the soft input to the decoder.

In summary, β_j can be written as:

$$\beta_j = \begin{cases} \sum\limits_{l=1, l\neq j, d_l \neq c_l}^{n} S(d_l) y_l, & \boldsymbol{c}_c \text{ found with } c_j \neq d_j \\ \min\limits_l (\beta_l > 0), & \boldsymbol{c}_c \text{ found, but } c_j = d_j \\ \operatorname*{mean}_l (|y_l|), & \text{no } \boldsymbol{c} \neq \boldsymbol{d} \text{ found.} \end{cases} \tag{6.28}$$

The extrinsic information is then given for all instances by (6.24).

Now that the soft-output of the block decoder has been defined, in the next section the iterative decoding of GLD codes using the above algorithm shall be considered.

6.3.4 Iterative Decoding of GLD Codes

GLD codes can be effectively decoded using the following decoding scheme. For each bit, we compute its probability given its received sample considering that it belongs to the super-code C^1. We use N/n SISO decoders working in parallel on the N/n independent constituent codes of C^1. Each decoder is implemented using the above algorithm. This step generates for each coded bit an *a posteriori* probability and *extrinsic* probability. The later one is fed through the interleaver to the second step as a *priori* information for the N/n SISO decoders working on the N/n constituent codes of C^2. This process is iterated on each super-code: $C^1 \rightarrow C^2 \rightarrow C^1 \rightarrow C^2 \rightarrow C^1 \rightarrow \ldots$ until the preset maximum iteration number is reached or a stopping criterion is satisfied. The q-th iteration of the decoding scheme is presented in Figure 5.2.

For each component code, the soft-input corresponding to the i^{th} bit position is defined as:

$$y_i'^{(1)}(q) = y_i + \alpha_c^{(1)}(q)\, w_i^{(2)}(q\text{-}1), \tag{6.29}$$

and

$$y_i'^{(2)}(q) = y_i + \alpha_c^{(2)}(q)\, w_i^{(1)}(q), \tag{6.30}$$

where y_i is the ith received sample from the channel, $w_i^{(2)}(q-1)$ is the extrinsic information for the ith bit position from the previous decoding of super-code C^2 and $w_i^{(1)}(q)$ is the extrinsic information for the ith bit position from the current decoding of super-code C^1.

In (6.29) and (6.30) $\alpha_c^{(j)}(q), j = 1, 2$, is a damping or scaling factor used to compensate for the difference in the variances of the Gaussian random variables y_i and y_i'. In [86], [89] it is shown that the correction factors $\alpha_c^{(j)}(q), j = 1, 2$, can be computed adaptively based on the statistics of the processed codewords as:

$$\alpha_c^{(1)}(q) = \frac{\mu_{w^{(2)}(q-1)}}{\mu_y} \frac{\sigma_y^2}{\sigma_{w^{(2)}(q-1)}^2}, \tag{6.31}$$

and

$$\alpha_c^{(2)}(q) = \frac{\mu_{w^{(1)}(q)}}{\mu_y} \frac{\sigma_y^2}{\sigma_{w^{(1)}(q)}^2}, \tag{6.32}$$

where $\mu_{w^{(2)}(q-1)}$ is the mean and $\sigma_{w^{(2)}(q-1)}^2$ the variance of the absolute value of $w^{(2)}(q-1)$, $\mu_{w^{(1)}(q)}$ is the mean and $\sigma_{w^{(1)}(q)}^2$ the variance of the absolute value of $w^{(1)}(q)$ and μ_y is the mean and σ_y^2 the variance of the absolute value of y.

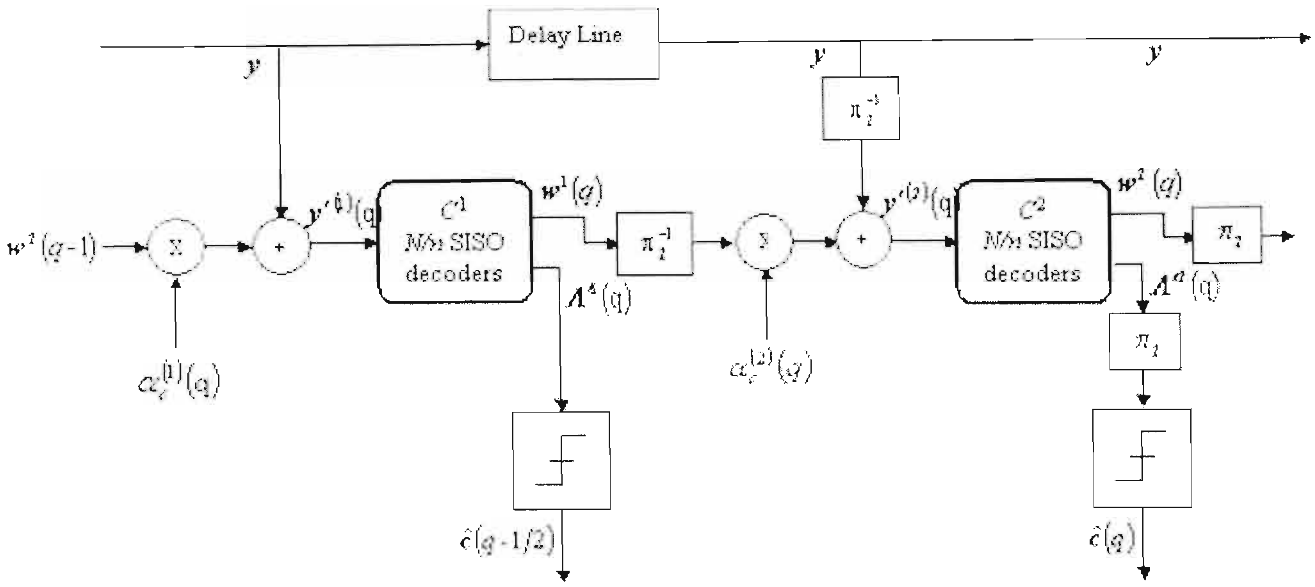

Figure 6.2: The q-th GLD decoding iteration.

6.3.5 Decoding Complexity

Now the complexity of the decoding algorithm is discussed. Most of the complexity required by decoding algorithms in the class that uses algebraic decoding to generate a set of candidate codewords is the computational complexity performed by the algebraic decoder. Thus the decoding complexity is defined to be the number of times one applies bounded-distance or the number of $e^{(r)}$'s generated during the decoding of y. Denote the number of times to apply bounded distance decoding as N_{BDD}, the number of $e^{(r)}$'s generated during the decoding of y as N_r, and their averages as $\overline{N}_{\mathrm{BDD}}$ and $\overline{N}_r$ respectively. Note that for the Chase-based algorithm of [85], [86], [88], [89] and the algorithm of [92], $\overline{N}_r = \overline{N}_{\mathrm{BDD}}$ as each test sequence is algebraically decoded to produce a possible codeword. For the proposed algorithm, however, $\overline{N}_{\mathrm{BDD}} < \overline{N}_r$. This is mainly due to the preprocessing test, which is used to avoid algebraic decoding of test sequences that would result in a codewords already found. This claim will be substantiated by computer simulation results in Section 6.5.

6.4 Stopping Criteria for GLD Decoding

Iterative decoding is a key feature of GLD codes. As the number of iteration increases, the bit error rate (BER) of the decoder decreases and the incremental improvement gradually diminishes. Each decoding iteration results in additional computations and decoding delay. Often, a fixed number M is chosen and each frame is decoded for M iterations (called "FIXED" in the following). Usually M is set with the worst corrupted frames in mind. Most frames need fewer iterations to converge. It would reduce the average computation and decoding delay substantially without performance degradation if the decoder terminated the iterations for each individual frame immediately after the bits are correctly estimated. For each decoded frame, the number of iterations performed is determined by the number of passes before a certain condition or rule for stopping is satisfied. The stopping condition attempts to determine when a frame can be reliably decoded with no further iterations, and it is computed based on the data available to the decoder during the decoding of each specific frame. More explicitly, at the end of each iteration, the decoder performs a check on the condition for stopping. If the condition is true, the iterative process on the frame is terminated, and the decoded sequence from the current iteration is sent to the output; otherwise, the iterative process continues to the next iteration. To prevent an endless loop should the stopping rule never be satisfied, we require that the decoding ceases after the maximum number of iterations, M.

Although iterative decoding improves the log-likelihood-ratio (LLR) value for each received bit through iterations, the hard decision of the received bit is ultimately made based on the sign of its LLR value. The hard decisions of the received sequence at the end of each iteration provide information on the convergence of the iterative decoding process. At each iteration the hard decisions of the received sequence at the output of each super-code are compared and the iterative process is terminated if they agree with each other for the entire block. This stopping criterion, called the hard-decision-aided (HDA) criterion, has been used in [97] for turbo decoding. Unlike the HDA proposed in [98], it requires no storage from the previous iteration and is very simple to implement. At each iteration, it requires N binary additions of sign bits and a counter not greater than N to check the sign changes. Whenever a sign change happens, the criterion is violated and the iterative process continues.

To save more iteration, at the cost of some degradation in performance, the number of bit difference in the hard decisions at the output of each super-code can be counted and the iterative process is terminated if this number is below a predefined threshold. We call this stopping criterion the improved hard-decision-aided (IHDA) criterion. Let D_{12} be the number of sign differences between the LLR at the output of each super-code, the IHDA terminating scheme is:

$$D_{12}\begin{cases}\geq p\times N, & \text{continue the iteration}\\ < p\times N, & \text{stop the iteration}\end{cases}, \qquad (6.33)$$

where p is the sign difference ratio, and N is the block length. It should be noted that this is different from the technique used in [99] where D_{12} is the number of sign differences between the extrinsic of the two SISO modules. Flexibility in performance versus computational complexity and decoding delay can be achieved simply by changing the value of p. Generally speaking, the smaller p is, the smaller the degradation in BER, but the larger the average number of required iterations.

6.5 Simulation Model and Results

Iterative decoding of two GLD codes for an AWGN channel with binary input using the above algorithm was simulated. The interleaver was chosen such that two component codes had not more than one digit in common. The algebraic decoder used by the improved Kaneko's algorithm was implemented using the Berlekamp-Massey algorithm as described in [100], [24]. The first code has length $N = 3969$, and rate $R = 0.6190$. Its constituent code is the (63, 51, 5) BCH code. It should be noted that the trellis complexity of this constituent code makes it impractical to be decoded with a trellis-based algorithm. The second code has length $N = 3969$, and rate $R = 0.8095$. Its constituent code is the (63, 57, 3) BCH code. These codes are referred to as code 1 and code 2 respectively.

The performance of the proposed algorithm for code 1 is shown in Figure 6.3 for different iteration steps, where E_b is the energy per coded bit and N_0 is the noise spectral density. The maximum number of times to apply bounded distance decoding for each iteration was set to 16. It is clearly seen that performance improves with each iteration, with diminishing returns. The Shannon limit for rate $R = 0.6190$ is $\cong$ 0.8 dB. It is observed that for this code, the E_b/N_0 for a bit error rate of 10^{-5} is at less than 1.8 dB of its Shannon's limit after 8 iterations. In Figure 6.4 the BER performance of the proposed algorithm is compared to that of the Chase-based algorithm with N_r = 16 in both cases. The results show that both algorithms exhibit similar performances. The complexity of both algorithms in terms of the average number of bounded-distance decoding per component code per iteration is shown in Figure 6.5. This number is closely related to the BER after decoding, and therefore, it is plotted against the BER after decoding rather than against the signal-to-noise ratio (SNR). It is observed that the proposed algorithm reduces $\overline{N}_{\mathrm{BDD}}$ by about 50% at a BER of 10^{-4}. This is of practical meaning since less decoding delay may be required to achieve satisfactory performance for a given application, thereby allowing for higher data rates.

Next, let consider the case where the maximum number of times to apply bounded-distance decoding for our decoding algorithm is the same as that for Chase-based algorithm. The performances of both algorithms are shown in Figure 6.6 for $\overline{N}_{\mathrm{BDD}}$ = 16. The results show that the proposed algorithm performs better than the Chase-based algorithm. At a BER of 10^{-5}, the improvement in performance is about 0.13 dB. This improvement can be explained by the fact our decoding algorithm can remove wasteful trials in Chase-based algorithm and can utilize part of the complexity to generate other codewords. Thus, our decoding algorithm increases the probability that the set of codewords generated will contain the codeword that is most likely. In addition, because our decoding algorithm generates more distinct codewords, the percentage of computing the βs using the theoretical value of (6.25) and hence, the extrinsic information is increased. Figure 6.7 shows the percentage of positions using the theoretical value of β for different BERs after decoding for both algorithms. This percentage is also related to the BER after decoding. It can be observed that, as the BER decreases, the Chase-based algorithm uses the theoretical value less and less compared to our algorithm. It should be noted, however, that this improvement in performance observed is at the cost of an increase in $\overline{N}_r$. However, since the complexity of generating an error pattern is far less than that of bounded-distance decoding, this increase in $\overline{N}_r$ is not an issue. Flexibility in performance versus computational complexity can be achieved simply by changing the maximum number of error patterns to be generated or the maximum number of bounded-distance decoding to be used. It should be noted, however, that this improvement in performance observed is at the cost of an increase in $\overline{N}_r$ as demonstrated in Figure 6.8. However, since the complexity of generating an error pattern is far less than that of bounded-distance decoding, this increase in $\overline{N}_r$ is not an issue.

Let now evaluate the efficiency of the stopping criteria proposed to improve the average decoding speed of the GLD decoder using the proposed algorithm. In Figure 6.9, the performance of the BER using the two terminating schemes is shown for code 1. It is observed that the HDA technique exhibit similar performance to the IHDA technique when $q = 0.001$. The performance of the IHDA degrades as the value of q increases. Compared to the fixed method, both the HDA and the IHDA (with $q = 0.001$) suffer a degradation in performance of less than 0.1 dB at a BER of $2.0*10^{-5}$. In Figure 6.10, the average number of iterations versus the signal-to-noise ratio is shown. It is observed that the IHDA technique is as efficient as the HDA method in terms of the average number of iteration when $q = 0.001$. As the value of q increases, the IHDA method saves more and more iterations but at the cost of some degradation in performance. These two figures show that the proper range of q, for which flexibility in performance versus computational complexity and decoding delay can be achieved, is $0.001 \leq q \leq 0.01$. The number of sign inconsistency pN increases as N grows. For a given N, p should be smaller for higher E_b/N_o. At BER $< 10^{-6}$ region, the performance is not degraded when $pN \leq 1$.

The proposed algorithm is now compared with the trellis-based MAP algorithm as used in [56] and [66]. To this end, we use code 2. The performances of both algorithms are shown in Figure 6.11 after 8 iterations in each case. The maximum number of times to apply bounded distance decoding for each iteration was set to 16. We can observe that the MAP algorithm converges earlier than the proposed algorithm, but that the slope of the latter is much steeper. As a result, both algorithms exhibit similar performance at a BER of 10^{-5}.

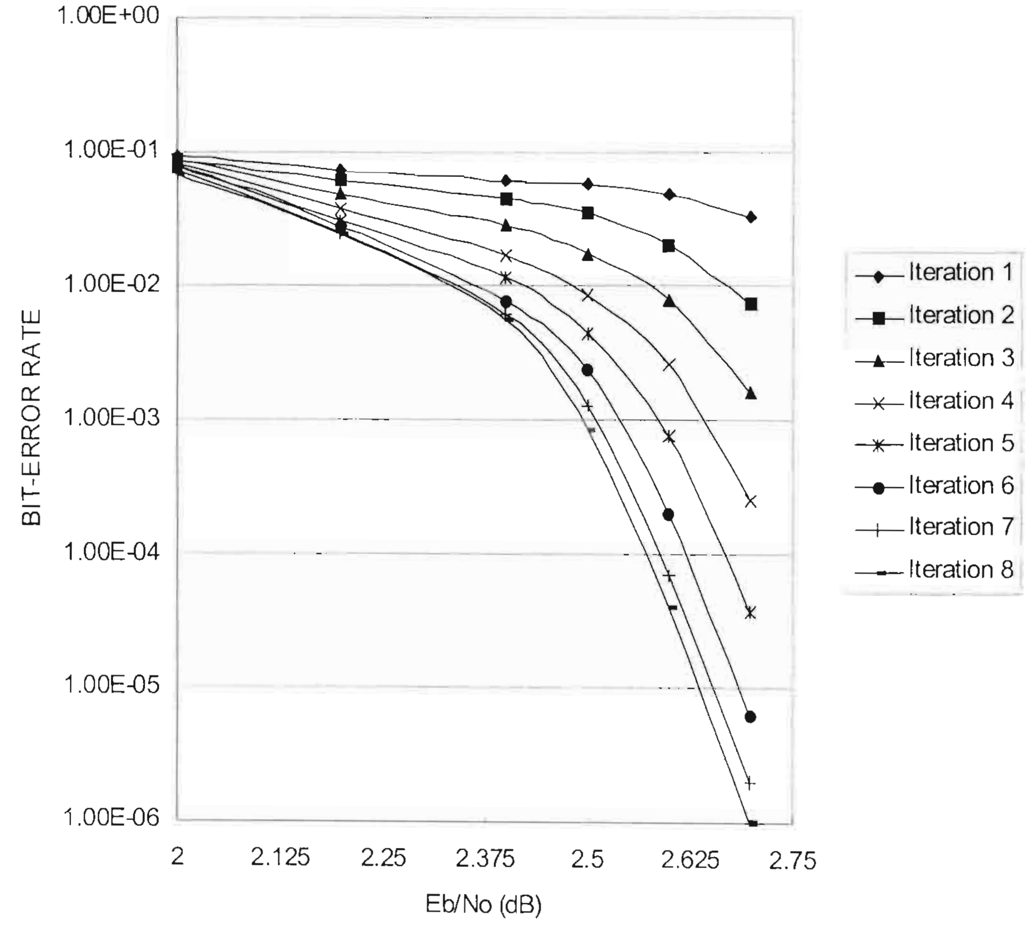

Figure 6.3: Performance of the proposed decoding algorithm for different number of iteration (for Code 1). Maximum N_{BDD} for each iteration = 16.

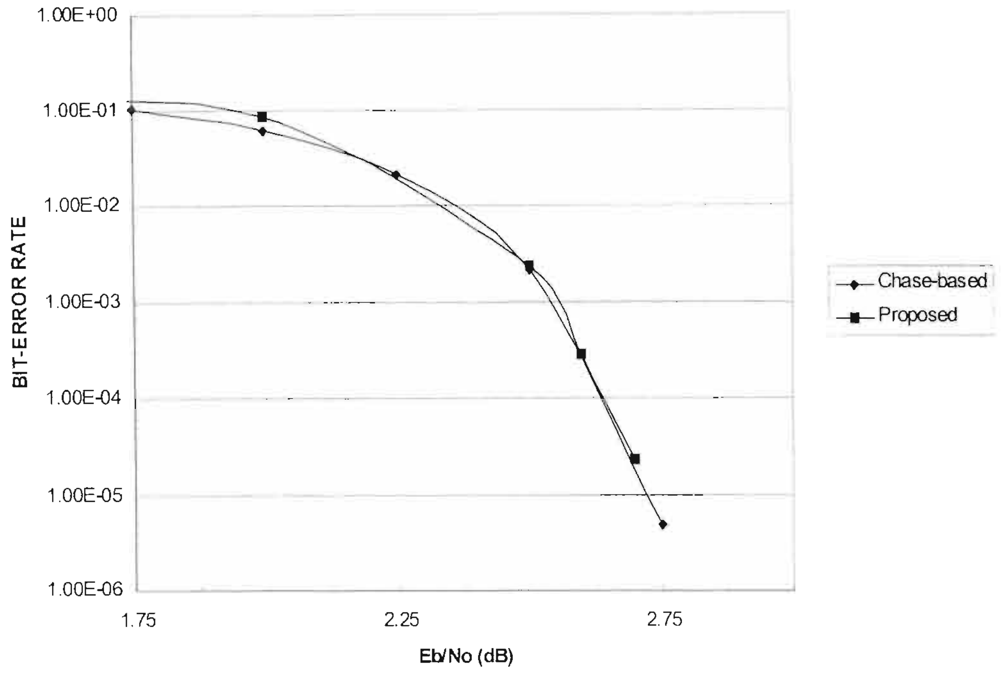

Figure 6.4: Performance of code 1 using the proposed algorithm and the chase-based algorithm after 8 iterations in both cases. The number of error pattern generated in both algorithms was made equal to 16.

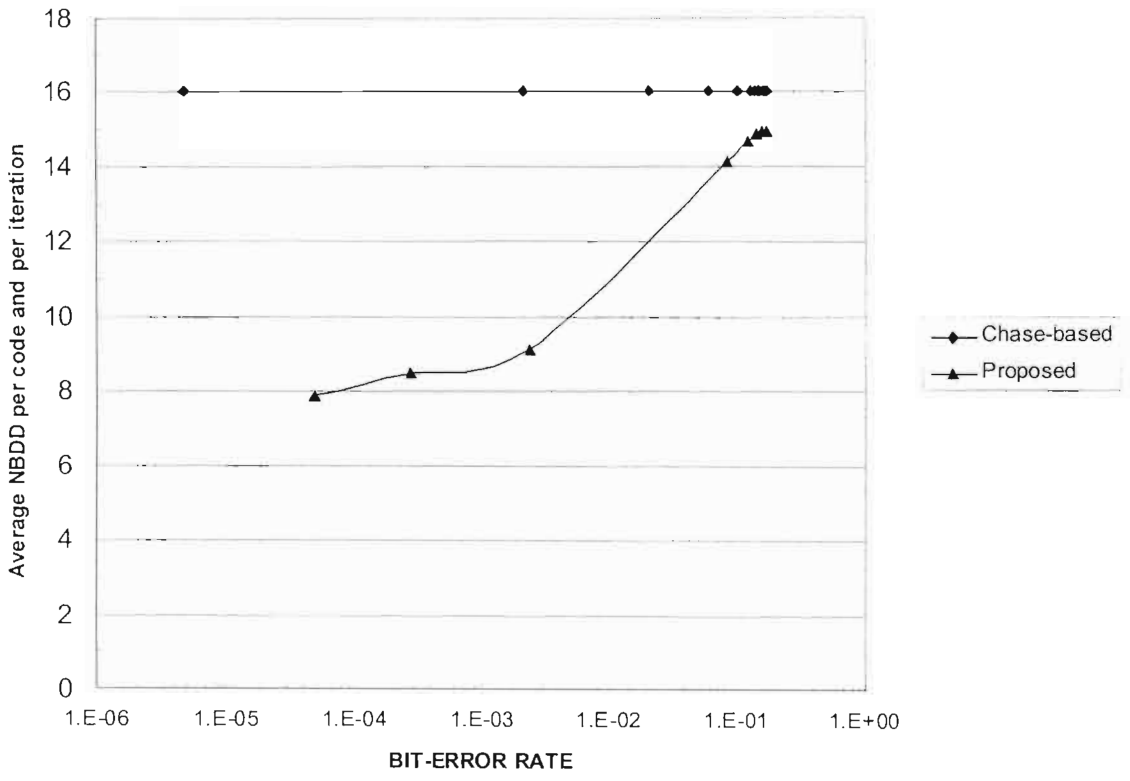

Figure 6.5: Comparison of the average number of bounded-distance decoding used by the proposed algorithm and the Chase-based algorithm for the same number of error patterns generated ($N_r = 16$). Code 1.

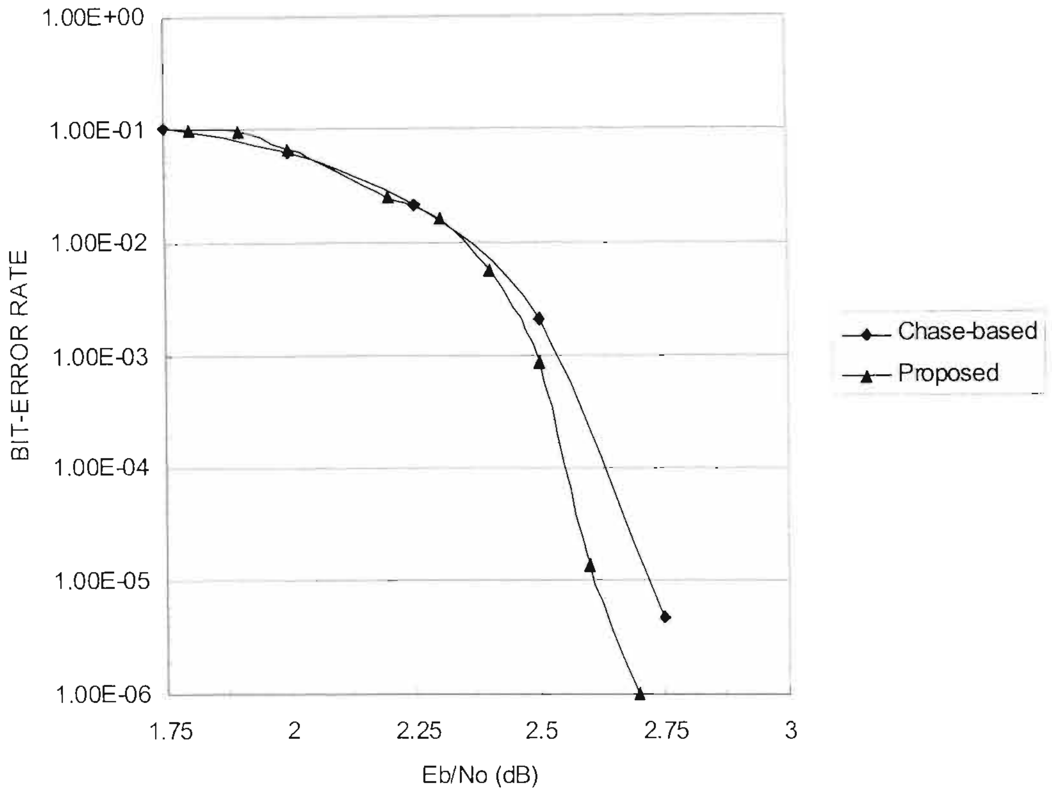

Figure 6.6: Performance of code 1 using the proposed algorithm and the chase-based algorithm after 8 iterations in both cases. The maximum number bounded-distance decoding for each iteration in both algorithms was made equal to 16.

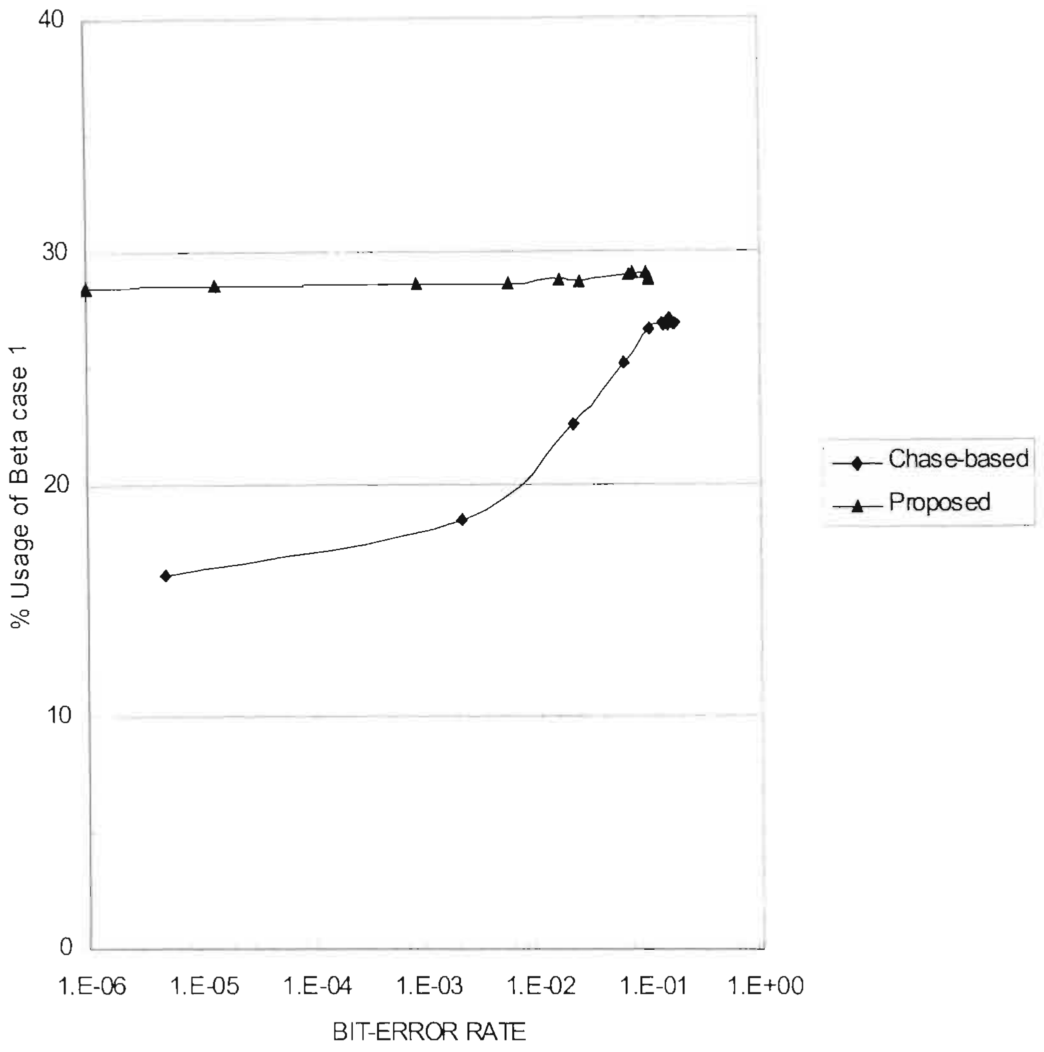

Figure 6.7: Percentage of positions using the theoretical value of beta (case 1) for different BERs after decoding for the proposed algorithm and the Chase-based algorithm. Code 1.

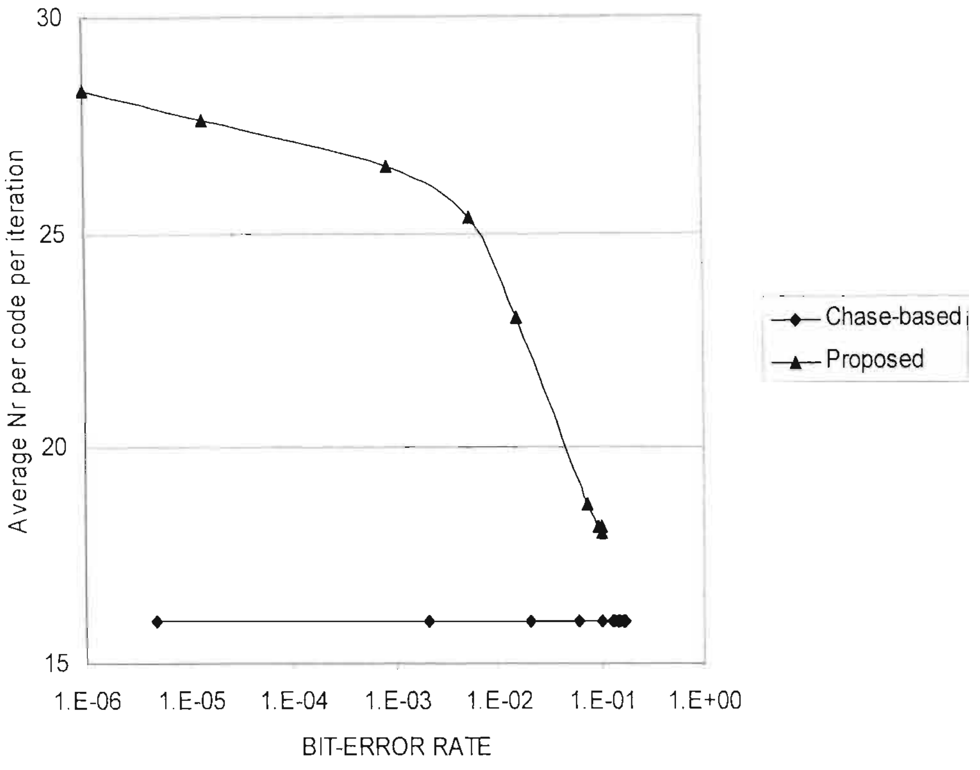

Figure 6.8: Comparison of the number of error patterns generated by the two decoding algorithms for the same number of bounded-distance decoding ($\overline{N}_{BDD} = 16$). Code 1.

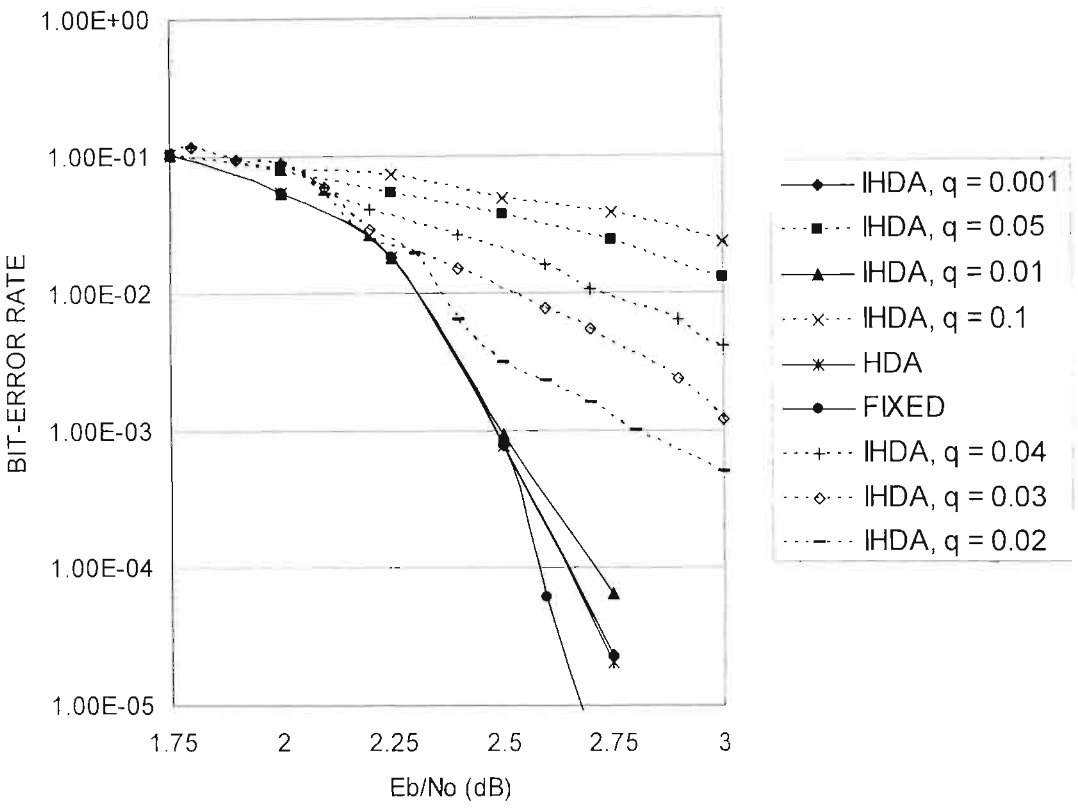

Figure 6.9: Simulated BER performance for the two terminating schemes and the fixed method using the proposed algorithm. Maximum $N_{\mathrm{BDD}} = 16$ for each iteration. Code 1.

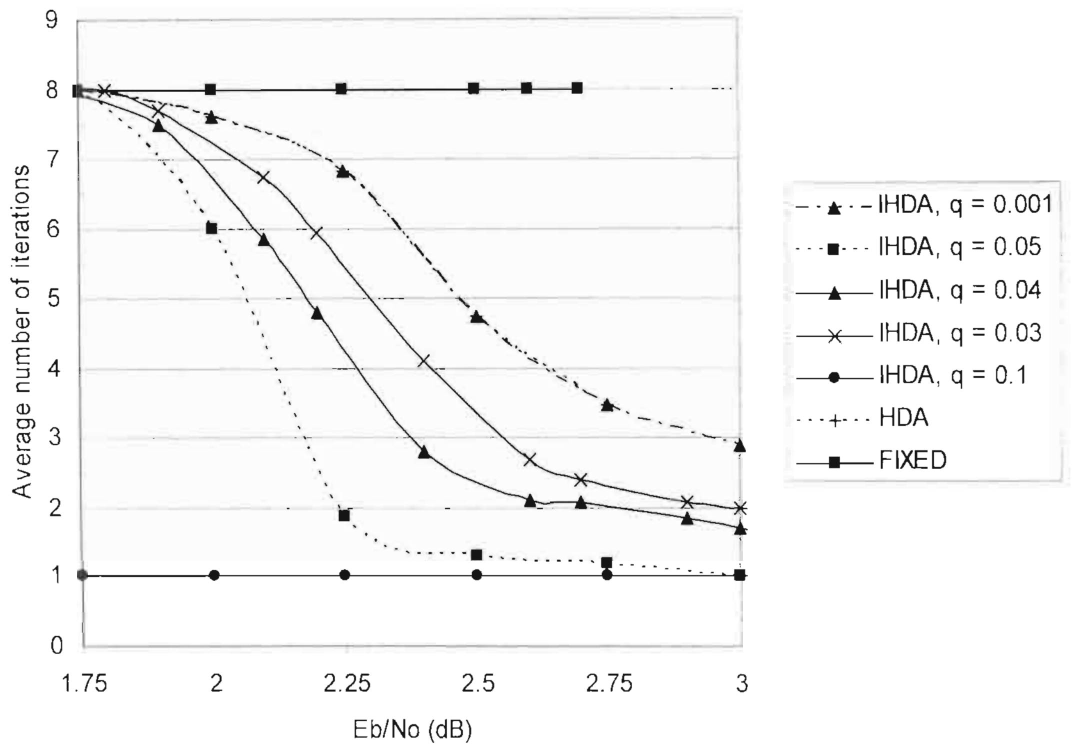

Figure 6.10: Simulated average number of iterations for the two terminating schemes and the fixed method using the proposed algorithm. Maximum N_{BDD} = 16 for each iteration. Code 1.

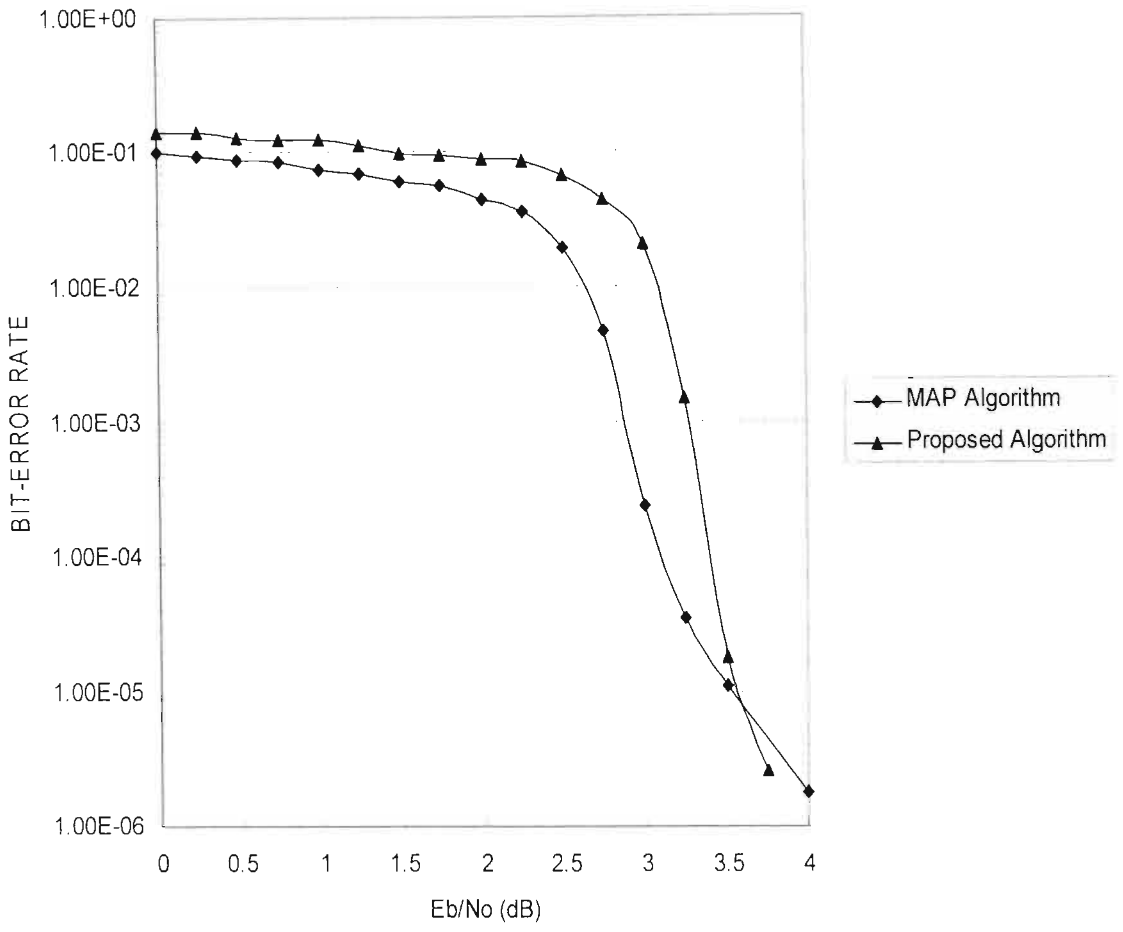

Figure 6.11: Performance comparison of the trellis-based MAP algorithm and the proposed algorithm for code 2. 8 iterations.

6.6 Conclusions

In this chapter, an efficient list-based decoding algorithm for GLD codes is proposed. The decoding algorithm modifies and utilizes improved Kaneko's algorithm for soft-input hard-output decoding. In contrast to the improved Kaneko algorithm, three modifications are introduced. First, list decoding is performed, i.e., all codewords generated are stored. Secondly, a simple preprocessing test to avoid algebraic decoding resulting in a codeword already found is proposed. This preprocessing test greatly reduces the complexity of the proposed algorithm when compare to those used in [85], [88], [89], [86]. Moreover, a method to generate soft-outputs is presented. To improve the average decoding speed of the GLD decoder, two simple criteria for stopping the iterative process for each frame immediately after the bits can be reliably decoded with no further iterations are proposed.

Compared to the trellis-based MAP decoding algorithm, the proposed algorithm suffers no degradation in performance at low bit-error rate, but has the additional advantages that it can be used in cases where the trellis-based MAP algorithm would be prohibitively complex and impractical. Compared to the Chase-based algorithm, the proposed algorithm is more efficient, has lesser computational complexity for the same performance, and provides an effective tradeoff between performance and computational complexity to facilitate its usage in practical applications.

The proposed algorithm can easily be applied to different concatenated or compound codes.

CHAPTER 7

UNION BOUND FOR BINARY LINEAR CODES ON THE GILBERT-ELLIOTT CHANNEL MODEL WITH APPLICATION TO TURBO-LIKE CODES

In this chapter, an analytical expression for the pairwise error probability of maximum likelihood decoding of a binary linear code on the Gilbert-Elliott (GE) channel model is derived. This expression is used to obtain the union bound on the bit error probability of linear codes on the GE channel. Comparisons between the results obtained by this analytical expression and results obtained through computer simulations in the case of turbo codes and generalized irregular low density (GILD) codes show that the analytical results are accurate in establishing the decoder performance in the range where obtaining sufficient data from simulation is impractical.

This chapter is organized as follows: A brief introduction to motivate the problem is first presented. In Section 7.2, the GE channel model is described; some known properties of the channel model are recapitulated, and useful statistical properties of the channel are derived. Section 7.3 describes how the GE model can be matched to the land mobile channel. In Section 7.4, the pairwise error probability on the GE channel is derived, and this expression is used in Section 7.5 and Section 7.6 to obtain union bound on the bit error probability of turbo codes and generalized irregular low density (GILD) codes respectively. The simulation model and results are presented in Section 7.7, and finally conclusions are drawn in Section 7.8.

7.1 Introduction

It is well known that the real-world communication channel has memory, often introducing noise distortion in a bursty fashion. In order to study the performance of error correcting codes for such a channel, it is sometimes practical to use a model whose properties are both complex enough to closely capture the real channel statistical characteristics and simple enough to allow mathematically tractable system analysis.

For a channel with memory, the Gilbert-Elliott (GE) channel, which emerges from the early 1960's and is due to Gilbert [101] and Elliott [102], is a useful and one of the simplest discrete models that has been studied in considerable detail in the literature. In this model, for a slowly varying channel, the channel is assumed to either be in a good state, where the probability of error is small, or in a bad state, where the probability of error is significantly larger. The dynamics of the channel are modelled as a first-order Markov chain, a model which Wang and Moayeri [103] and Wang and Chang [104], in spite of its simplicity, showed to be very accurate for a Rayleigh fading channel. In [105], Ahlin presented a way to match the parameters of the GE model to the land mobile channel, an approach that was generalized in [103] to a Markov model with more than two states. This approach was also used by Sharma *et al.* in [106] and [107] where error trapping decoders are studied.

Elliott [102] first analyzed the performance of error-correcting codes on a GE channel by establishing a series of recursions for $P(m,n)$, the probability of m transmission errors in a block of n symbols. Recently Yee and Weldon [108] presented a combinatorial analysis for a simplified GE channel that replaced the recursions with closed-form expressions. An alternate nonrecursive technique for approximate evaluation of $P(m,n)$ on simplified GE channels has also been presented by Wilhelmsson and Milstein [109] and by Wong and Leung [110]. These analyses considered block-coded transmission and are useful when the error correcting capability of the code is known.

In this chapter, an expression for the pairwise error probability on the GE channel is derived. This expression can be used to obtain union bounds of the block and bit error probabilities of a linear code with maximum likelihood (ML) decoding if the spectrum of the code is known. Comparisons between the results obtained by this analytical expression and results obtained through computer simulations in the case of turbo codes and generalized irregular low density (GILD) codes show that the analytical results are accurate in establishing the decoder performance in the range where obtaining sufficient data from simulation is impractical.

7.2 The Gilbert-Elliott Channel Model

The GE channel is a first-order, discrete-time, stationary, Markov chain with two states, one good and one bad, appropriately denoted by G and B. In the good state errors occur with low probability $P_e(G)$ while in the bad state they occur with high probability $P_e(B)$. The probabilities that the channel state changes from G to B and from B to G are denoted by b and g, respectively. The model is shown in Figure 7.1. The steady state probabilities of being in states G and B are $\pi_G = \frac{g}{b+g}$ and $\pi_B = \frac{b}{b+g}$, respectively.

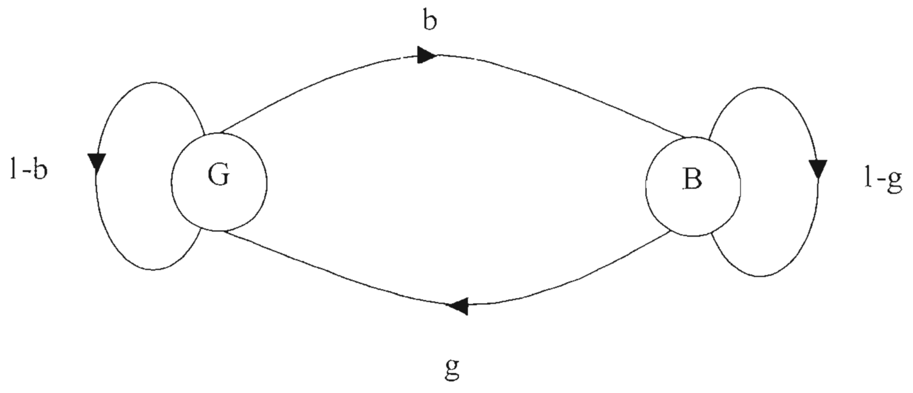

Figure 7.1: The Gilbert-Elliott channel model.

In the good state at time k, the noise is assumed to be additive white Gaussian noise (AWGN) with power spectral density $N_G/2$ where N_G typically has low magnitude.

Similarly, in the bad state, the noise is white Gaussian with power spectral density $N_B/2$ where $N_B > N_G$.

Figure 7.2 is useful in interpreting the significance of the two states. It shows the received SNR, γ, vs. time. If the SNR drops below some pre-determined threshold γ_T, then the channel goes into the bad state else it resides in the good state. In either state, the channel exhibits the properties of a binary symmetric channel. Figures 7.3 and 7.4 show the channel bit error characteristics in the two states respectively. In the bad state, the probability of error is $P_e(B)$ and in the good state the probability of error is $P_e(G)$. Normally, $P_e(G) \ll P_e(B)$.

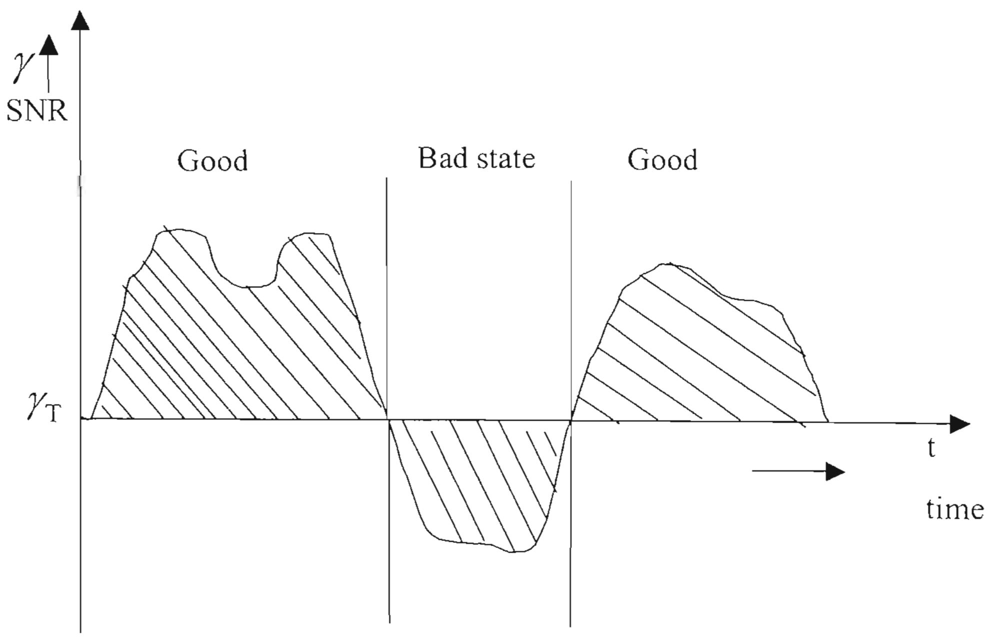

Figure 7.2: Physical interpretation of the states in the Gilbert-Elliott model.

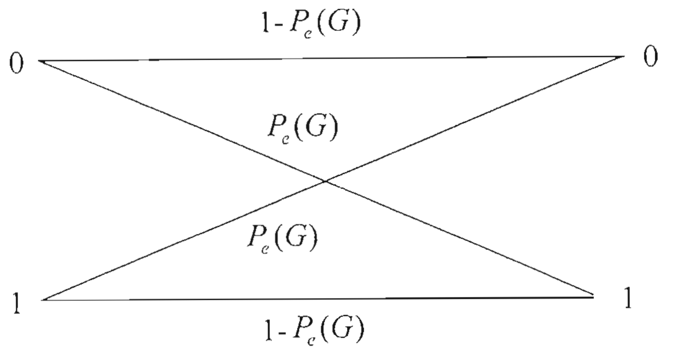

Figure 7.3: Good state binary symmetric channel .

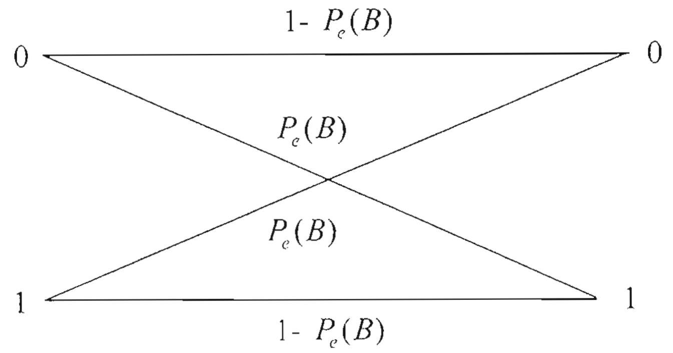

Figure 7.4: Bad state binary symmetric channel.

Let *T(G)* and *T(B)* be the number of time units the channel spends in the good and bad state respectively. Their averages, $\bar{T}(G)$ and $\bar{T}(B)$, can be obtain as follows:

Assume the channel is in the good state and let

$$Y = \begin{cases} 1, & \text{if the next transition is from } G \text{ to } B \\ 0, & \text{if the next transition is from } G \text{ to } G \end{cases}, \qquad (7.1)$$

$$\begin{aligned}\bar{T}(G) = E\left[T(G)\right] &= E\left[E\left[T(G)/Y\right]\right] \\ &= E\left[T(G)/Y=1\right]\Pr\{Y=1\} + E\left[T(G)/Y=0\right]\Pr\{Y=0\} \\ &= bE\left[T(G)/Y=1\right] + (1-b)E\left[T(G)/Y=0\right] \\ &= b + (1-b)\left(1 + E\left[T(G)\right]\right).\end{aligned}$$

Therefore

$$\bar{T}(G) = E\left[T(G)\right] = \frac{1}{b}. \tag{7.2}$$

Likewise, assume the channel is in the bad state and let

$$X = \begin{cases} 1, & \text{if the next transition is from } B \text{ to } G \\ 0, & \text{if the next transition is from } B \text{ to } B \end{cases}, \tag{7.3}$$

$$\begin{aligned}\bar{T}(B) = E\left[T(B)\right] &= E\left[E\left[T(B)/X\right]\right] \\ &= E\left[T(B)/X=1\right]\Pr\{X=1\} + E\left[T(B)/X=0\right]\Pr\{X=0\} \\ &= gE\left[T(B)/X=1\right] + (1-g)E\left[T(B)/X=0\right] \\ &= g + (1-g)\left(1+E\left[T(B)\right]\right).\end{aligned}$$

Therefore

$$\bar{T}(B) = E\left[T(B)\right] = \frac{1}{g}. \tag{7.4}$$

In words *T(G)* and *T(B)* are geometric random variables with parameter b and g respectively.

7.3 Matching the Gilbert-Elliott Channel Model to the Land Mobile Channel

In this section the generative GE model is related to the analogue Rayleigh fading model, where the correlation function is given by the zeroth-order Bessel function, a model commonly used for the land mobile channel, e.g. [111]. To do this, some results concerning the memory of the analogue model are needed. As can be seen in Figure 7.5, the signal envelope only occasionally experiences very deep fades. Shallow fades are more likely to occur. A quantitative expression of this property is the level crossing rate, *h(.)*, which is defined as the expected rate at which the envelope crosses a specified threshold, γ_T, in the positive direction. If the SNR, γ, is used instead of the signal envelope level, R, the following expression for the level crossing rate is obtained [111],

$$h(\gamma_T) = f_D\sqrt{2\pi\frac{\gamma_T}{\bar{\gamma}}}\exp(-\gamma_T/\bar{\gamma}), \tag{7.5}$$

where γ_T is the specified SNR threshold and $\bar{\gamma}$ is the average SNR of the received signal. The maximum Doppler shift, f_D, is given by

$$f_D = v/\lambda, \tag{7.6}$$

where v is the velocity of the moving vehicle and λ is the carrier wavelength. Related to the level crossing rate is the expected duration of the fade below the specified threshold, γ_T, which is given by:

$$E[\tau_h] = \frac{1}{h(\gamma_T)} \mathrm{pr}[\gamma \le \gamma_T]. \tag{7.7}$$

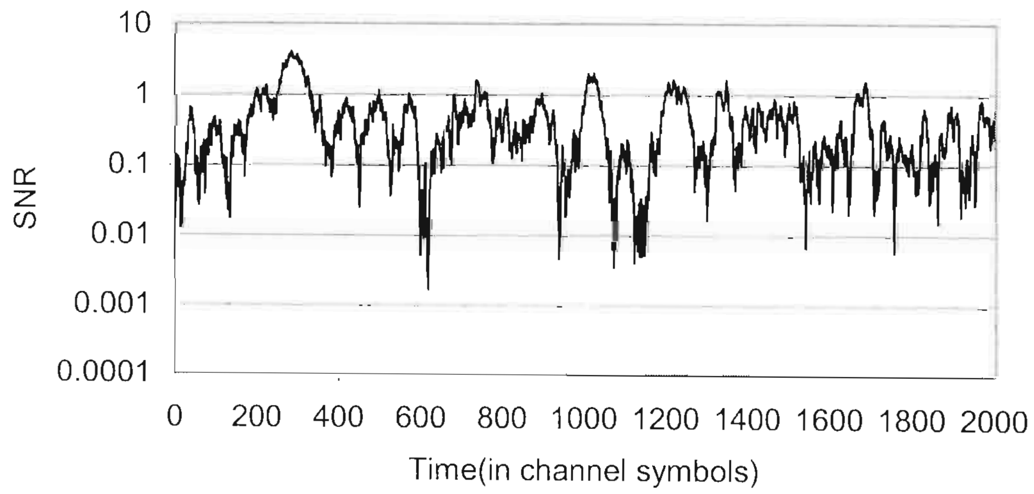

Figure 7.5: An example of the received signal envelope on a typical simulated Rayleigh fading (f_m =0.001).

In (7.7), $\mathrm{pr}[\gamma \le \gamma_T]$ represents the fraction of time the Rayleigh fading channel is below γ_T and is given by:

$$\mathrm{pr}[\gamma \le \gamma_T] = \int_0^{\gamma_T} f(\gamma) d\gamma, \tag{7.8}$$

where $f(\gamma)$ is the distribution of the received SNR. It should be noted that the Rayleigh fading results in an exponentially distributed multiplicative distortion of the signal. As a result, the probability density function of the signal-to-noise ration (SNR), γ, is given by:

$$f(\gamma)=\frac{1}{\bar{\gamma}}\exp(-\gamma/\bar{\gamma}),\ \gamma\geq 0. \tag{7.9}$$

Substituting (7.9) into (7.8) we obtain

$$\mathrm{pr}\left[\gamma\leq\gamma_{\mathrm{T}}\right]=1-\exp(-\rho), \tag{7.10}$$

where $\rho=\gamma_{\mathrm{T}}/\bar{\gamma}$. So the expected fade duration in a Rayleigh fading environment is

$$E[\tau_b]=\frac{\exp(\rho)-1}{f_D\sqrt{2\pi\rho}}. \tag{7.11}$$

In the same way, the average non-fade duration $E[\tau_g]$ above the threshold is given by

$$E[\tau_g]=\frac{1}{h(\gamma_T)}\mathrm{pr}\left[\gamma\geq\gamma_T\right]$$

$$=\frac{1}{f_{\mathrm{D}}\sqrt{2\pi\rho}}. \tag{7.12}$$

We want to relate the discrete channel model to the analogue one in such a way that the discrete model should generate approximately the same error distribution as the analogue channel (including the modulator and the demodulator). A natural way to match the two models is to relate the state sequence to the fading signal envelope. Let the bad state represents the situation when the signal envelope is below some threshold and let the good state represents the situation when the signal envelope is above the threshold. We then let the average number of time units the channel spends in the Good (Bad) states be equal to the expected non-fade (fade) duration, normalized with the symbol time-interval T_S,

$$\bar{T}(G)=E\left[\tau_g\right]\frac{1}{T_S}, \tag{7.13}$$

$$\bar{T}(B)=E\left[\tau_b\right]\frac{1}{T_S}. \tag{7.14}$$

Hence, the transition probabilities are given by:

$$b=f_D T_S\sqrt{2\pi\rho}, \tag{7.15}$$

and

$$g=\frac{f_D T_S\sqrt{2\pi\rho}}{\exp(\rho)-1}. \tag{7.16}$$

The next step is to calculate the error probabilities in each state. They are taken to be the conditional error probabilities of the Rayleigh fading channel, conditioned on being in the respective state, i.e.

$$P_e(B)=\frac{1}{\pi_B}\int_0^{\gamma_T} f(\gamma)P(\gamma)d\gamma, \tag{7.17}$$

and

$$P_e(G)=\frac{1}{\pi_G}\int_{\gamma_T}^{\infty} f(\gamma)P(\gamma)d\gamma, \tag{7.18}$$

where $P(\gamma)$ is the symbol error probability for a given value of γ, which depends on the modulation format used. For BPSK modulation with coherent demodulation, the conditional probability of a code symbol error, conditioned on the received SNR, is given by [112]

$$P_e(\gamma)=\frac{1}{2}\text{erfc}\left(\sqrt{\gamma}\right), \tag{7.19}$$

where $\text{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_x^{\infty}\exp(-t^2)dt$ is the complementary error function.

Substituting (7.9) and (7.19) into (7.17), the probability of error in the bad state, $P_e(B)$ is

$$P_e(B)=\frac{\int_0^{\gamma_T}\frac{1}{2}\text{erfc}\left(\sqrt{\gamma}\right)\frac{1}{\bar{\gamma}}\exp(-\gamma/\bar{\gamma})d\gamma}{\int_0^{\gamma_T}\frac{1}{\bar{\gamma}}\exp(-\gamma/\bar{\gamma})d\gamma}$$

$$=\frac{\int_0^{\gamma_T}\frac{1}{2}\left(\int_{\sqrt{\gamma}}^{\infty}\frac{2}{\sqrt{\pi}}\exp\left(\text{-t}^2\right)\,\text{dt}\right)\frac{1}{\bar{\gamma}}\exp\left(\text{-}\gamma/\bar{\gamma}\right)\,\text{d}\gamma}{\exp\left(\text{-}\gamma,\bar{\gamma}\right)|_{\gamma_T}^{0}}$$

$$=\frac{\frac{1}{2}I}{1\text{-}\exp\left(\text{-}\gamma_T/\bar{\gamma}\right)}. \tag{7.20}$$

The integral, I, in the numerator defined as

$$I=\int_0^{\gamma_T}\left(\int_{\sqrt{\gamma}}^{\infty}\frac{2}{\sqrt{\pi}}\exp\left(\text{-t}^2\right)\,\text{dt}\right)\frac{1}{\bar{\gamma}}\exp\left(\text{-}\gamma/\bar{\gamma}\right)\,\text{d}\gamma \tag{7.21}$$

can be evaluated by resorting to integration by parts.

We obtain

$$I = \text{-erfc}\sqrt{\gamma}\ \exp\left(-\gamma/\bar{\gamma}\right)\Big|_0^{\gamma_T} - \int_0^{\gamma_T} \frac{1}{\sqrt{\pi}} \exp\left[-\left(1+\frac{1}{\bar{\gamma}}\right)\gamma\right]\frac{\mathrm{d}\gamma}{\sqrt{\gamma}}$$

$$= 1\text{-erfc}\sqrt{\gamma_T}\ \exp\left(-\gamma_T/\bar{\gamma}\right) - I_1. \tag{7.22}$$

To evaluate I_1 in equation (7.22), we use the substitution

$$u^2 = \left(1+\frac{1}{\bar{\gamma}}\right)\gamma, \tag{7.23}$$

which gives

$$I_1 = \int_0^{\gamma_T} \frac{1}{\sqrt{\pi}} \exp\left[-\left(1+\frac{1}{\bar{\gamma}}\right)\gamma\right]\frac{d\gamma}{\sqrt{\gamma}}$$

$$= \frac{1}{\sqrt{1+\frac{1}{\bar{\gamma}}}} \int_0^{\sqrt{(1+1/\bar{\gamma})\gamma_T}} \frac{2}{\sqrt{\pi}} \exp\left(-u^2\right) du$$

$$= \frac{1}{\sqrt{1+\frac{1}{\bar{\gamma}}}}\left[1\text{-erfc}\left(\sqrt{\gamma_T\left(1+\tfrac{1}{\bar{\gamma}}\right)}\right)\right]. \tag{7.24}$$

Therefore,

$$I = 1 - \text{erfc}\sqrt{\gamma_T}\ \exp\left(-\gamma_T/\bar{\gamma}\right) - \frac{1}{\sqrt{1+\frac{1}{\bar{\gamma}}}}\left[1-\text{erfc}\left(\sqrt{\gamma_T\left(1+\tfrac{1}{\bar{\gamma}}\right)}\right)\right]. \tag{7.25}$$

Substituting the value of I in equation (7.20), the simplified expression of $P_e(B)$ is

$$P_e(B) = \frac{1-\text{erfc}\left(\sqrt{\gamma_T}\right)\exp(-\rho) - \frac{1}{\sqrt{1+\frac{1}{\bar{\gamma}}}}\left[1-\text{erfc}\left(\sqrt{\gamma_T+\rho}\right)\right]}{2\left[1\text{-exp}(-\rho)\right]}. \tag{7.26}$$

The probability of error in the good state, $P_e(G)$, can similarly be expressed as

$$P_e(G) = \frac{\int_{\gamma_T}^{\infty} f(\gamma)P(\gamma)d\gamma}{\int_{\gamma_T}^{\infty} f(\gamma)d\gamma}$$

$$= \frac{\int_{\gamma_T}^{\infty} \frac{1}{2}\left(\int_{\sqrt{\gamma}}^{\infty} \frac{2}{\sqrt{\pi}} \exp\left(-t^2\right) \mathrm{d}t\right)\frac{1}{\bar{\gamma}}\exp\left(-\gamma/\bar{\gamma}\right)\mathrm{d}\gamma}{\int_{\gamma_T}^{\infty} \frac{1}{\bar{\gamma}}\exp\left(-\gamma/\bar{\gamma}\right)\mathrm{d}\gamma}$$

$$= \frac{\frac{1}{2} I_2}{\exp(-\gamma_T / \bar{\gamma})}, \tag{7.27}$$

where

$$I_2 = \int_{\gamma_T}^{\infty} \left(\int_{\sqrt{\gamma}}^{\infty} \frac{2}{\sqrt{\pi}} \exp(-t^2)\, dt \right) \frac{1}{\bar{\gamma}} \exp(-\gamma/\bar{\gamma})\, d\gamma . \tag{7.28}$$

I_2 can also be integrated by parts as was done for I yielding

$$I_2 = -\operatorname{erfc}\sqrt{\gamma}\ \exp(-\gamma/\bar{\gamma})\Big|_{\gamma_T}^{\infty} - \int_{\gamma_T}^{\infty} \frac{1}{\sqrt{\pi}} \exp\left[-\left(1+\tfrac{1}{\bar{\gamma}}\right)\gamma\right] \frac{d\gamma}{\sqrt{\gamma}}$$

$$= \operatorname{erfc}\sqrt{\gamma_T}\, \exp(-\gamma_T/\bar{\gamma}) - \frac{1}{\sqrt{1+\frac{1}{\bar{\gamma}}}} \int_{\sqrt{(1+1/\bar{\gamma})\gamma_T}}^{\infty} \frac{2}{\sqrt{\pi}} \exp(-u^2) du$$

$$= \operatorname{erfc}\sqrt{\gamma_T}\, \exp(-\gamma_T/\bar{\gamma}) - \int_{\sqrt{(1+1/\bar{\gamma})\gamma_T}}^{\infty} \frac{2}{\sqrt{\pi}} \exp(-u^2) du \tag{7.29}$$

Hence, $P_e(G)$ can be expressed in the following form:

$$P_e(G) = \frac{1}{2} \left[\operatorname{erfc}\left(\sqrt{\gamma_T}\right) - \frac{\exp(\rho)}{\sqrt{1+\frac{1}{\bar{\gamma}}}} \operatorname{erfc}\left(\sqrt{\gamma_T + \rho}\right) \right]. \tag{7.30}$$

7.4 Pairwise Error Probability on the Gilbert-Elliott Channel

In this section we derive the expression of the pairwise ML decoding error probability on the GE channel for two codewords/paths in a code trellis which differ in d symbols when the channel state is known exactly to the decoder. If the channel state is known exactly to the decoder we assume that amongst the d bits in which the wrong path and the correct path differ, there are d_B in the bad state and $d_G = d - d_B$ in state G. Amongst the bits in the bad state, there are e_B bits in error and amongst the "good" bits e_G are in error. Let $CM^{(1)}$ and $CM^{(0)}$ be the likelihood metric of the wrong path and the correct path respectively, restricted over the d positions in which the two paths differ;

$$\begin{aligned} CM^{(1)} &= e_B \log(1-P_e(B)) + (d_B - e_B)\log P_e(B) \\ &+ e_G \log(1-P_e(G)) + (d_G - e_G)\log P_e(G) \end{aligned}, \tag{7.31}$$

and

$$\begin{aligned} CM^{(0)} &= (d_B - e_B)\log(1-P_e(B)) + e_B \log P_e(B) \\ &+ (d_G - e_G)\log(1-P_e(G)) + e_G \log P_e(G) \end{aligned}. \tag{7.32}$$

The probability of error in the pairwise comparison of the likelihood metrics $CM^{(1)}$ and $CM^{(0)}$ is:

$$P_2(d) = \Pr\left(CM^{(1)} > CM^{(0)}\right). \tag{7.33}$$

If both metric values are equal, a random choice is made with probability 1/2. If equations (7.31) and (7.32) are substituted into (7.33) we obtain:

$$P_2(d) = \Pr\left[(d_G + Cd_B) < 2(e_G + Ce_B)\right], \tag{7.34}$$

where C is the metric ratio defined as

$$C = \frac{\log\left[(1-P_e(B))/P_e(B)\right]}{\log\left[(1-P_e(G))/P_e(G)\right]}. \tag{7.35}$$

To evaluate $P_2(d)$, the probability distribution for being in the bad state d_B times out of d and the distribution for being in the good state d_G times out of d is needed. It is shown in Appendix C that

$$P_d(d_B) = \begin{cases} (1-b)^{d-1}\pi_G, & d_B = 0 \\ \left[P_d(d_B/GG) + P_d(d_B/GB)\right]\pi_G + \left[P_d(d_B/BG) + P_d(d_B/BB)\right]\pi_B, & 1 \le d_B < \mathrm{d} \\ (1-g)^{d-1}\pi_B, & d_B = d \end{cases} \tag{7.36}$$

where

$$P_d(d_B/GG) = \sum_{i=2}^{\min(d_B+1,d-d_B)} \binom{d-d_B-1}{i-1}\binom{d_B-1}{i-2}(1-b)^{d-d_B-i} b^{i-1}(1-g)^{d_B-i+1} g^{i-1}, \tag{7.37}$$

$$P_d(d_B/GB) = \sum_{i=1}^{\min(d_B,d-d_B)} \binom{d-d_B-1}{i-1}\binom{d_B-1}{i-1}(1-b)^{d-d_B-i} b^{i}(1-g)^{d_B-i} g^{i-1}, \tag{7.38}$$

$$P_d(d_B/BG) = \sum_{i=1}^{\min(d_B,d-d_B)} \binom{d-d_B-1}{i-1}\binom{d_B-1}{i-1}(1-b)^{d-d_B-i} b^{i-1}(1-g)^{d_B-i} g^{i}, \tag{7.39}$$

$$P_d(d_B/BB) = \sum_{i=2}^{\min(d_B,d-d_B+1)} \binom{d-d_B-1}{i-2}\binom{d_B-1}{i-1}(1-b)^{d-d_B-i+1} b^{i-1}(1-g)^{d_B-i} g^{i-1}. \tag{7.40}$$

Here, $P_d(d_B/GG)$ is the conditional probability of being d_B times in the bad state, conditioned on being in the good state both the first and the last instants of time. The other probabilities are defined accordingly.

Following the same reasoning, it is easy to show that,

$$P_d(d_G)=\begin{cases}(1-g)^{d-1}\pi_B, & d_G=0\\ \left[P_d(d_G/GG)+P_d(d_G/GB)\right]\pi_G+\left[P_d(d_G/BG)+P_d(d_G/BB)\right]\pi_B, & 1\le d_G<d\\ (1-b)^{d-1}\pi_G, & d_G=d\end{cases}\tag{7.41}$$

where

$$P_d(d_G/GG)=\sum_{i=2}^{\min(d_G,d-d_G+1)}\binom{d-d_G-1}{i-2}\binom{d_G-1}{i-1}(1-b)^{d_G-i}b^{i-1}(1-g)^{d-d_G+1}g^{i-1},\tag{7.42}$$

$$P_d(d_G/GB)=\sum_{i=1}^{\min(d_G,d-d_G)}\binom{d-d_G-1}{i-1}\binom{d_G-1}{i-1}(1-b)^{d_G-i}b^{i}(1-g)^{d-d_G-i}g^{i-1},\tag{7.43}$$

$$P_d(d_G/BG)=\sum_{i=1}^{\min(d_G,d-d_G)}\binom{d-d_G-1}{i-1}\binom{d_G-1}{i-1}(1-b)^{d_G-i}b^{i-1}(1-g)^{d-d_G-i}g^{i},\tag{7.44}$$

$$P_d(d_G/BB)=\sum_{i=2}^{\min(d_G+1,d-d_G)}\binom{d-d_G-1}{i-1}\binom{d_G-1}{i-2}(1-b)^{d_G-i+1}b^{i-1}(1-g)^{d-d_G-i}g^{i-1}.\tag{7.45}$$

Thus

$$P_2(d)=\sum_{d_B+d_G=d}\sum_{e_B}\sum_{e_G}\left\{\begin{array}{l}\binom{d_B}{e_B}P_c(B)^{e_B}\left(1-P_c(B)\right)^{d_B-e_B}P_d(d_B)\\ \times\binom{d_G}{e_G}P_c(G)^{e_G}\left(1-P_c(G)\right)^{d_G-e_G}P_d(d_G)\end{array}\right. .\tag{7.46}$$

In (7.46), the summation over e_B and e_G is restricted to those values e_B and e_G which fulfil the inequality in (7.34) and the factor 1/2 is introduced in case of equality. This expression can be used to compute the union bound on the word and bit-error probability of linear codes with maximum-likelihood decoding on the GE channel. In the next two sections, it is applied to turbo codes and generalized irregular low density (GILD) codes respectively.

7.5 Application to Turbo Codes

Since its introduction in 1993 by Berrou et *al.* [4], [5] turbo coding has raised great interest in the communication community. A turbo encoder is formed by two parallel concatenated recursive convolutional encoders connected by an interleaver. The interleaver performs permutation of the input sequence. The decoder consists of two iterative maximum-a-posteriori (MAP) decoders connected by an interleaver and deinterleaver. Turbo codes have excellent bit error rate (BER) performance at low SNRs. Performance near Shannon capacity limit can be obtained with large interleavers, and comparatively good performance is possible for any interleaver size. However, at high SNR, a flattening phenomenon of the BER curves has been observed [113]. This is called the "error floor" of turbo codes. It is believed that the "error floor" is caused by a small minimum distance. In general, it is very difficult to characterize the behaviour of the "error floor" using simulation (the computational cost is too high). Instead, one resorts to theoretical bounds. Many researchers have studied performance bounds for turbo codes [113]–[118], [73], [78]. Although these works are studied on AWGN channels, performance analysis of turbo codes over fading channels have also been investigated [119], [120]. More recently, Kang et *al.* [121] and Garcias-Frias and Villasenor [122] considered the design and performance of turbo codes on a GE channel and proposed the necessary modifications to the turbo decoder. No analytical work on the BER was done in these papers.

In this section, using the expression of the pairwise error probability derived in the previous section, an expression for the union bound on the bit error probability of turbo codes on the GE channel is obtained. With the assumption of *uniform interleaver* (a collection of deterministic interleavers, each of which occurs with equal probability), the key of union bound analysis of turbo codes largely depends on the weight enumerating function (WEF) of the constituent codes. Although Benedetto et *al.* were the first to introduce the concept of a uniform interleaver, their method of calculating the WEF is not accurate and not straightforward to comprehend. Divsalar et *al.* [116] presented a recursive method, based on the transfer function method [123], to obtain the WEF. Consider the traditional union upper bound for the ML decoding of a (N,K)

block code. Without loss of generality, it is assumed that the all-zeros codeword was sent. The expression for the average bit error probability for the two constituent rate 1/2 convolutional codes is given by [116]

$$P_b = \sum_{d=d_{\min}}^{M} \sum_{i} \sum_{d_1} \sum_{d_2} \frac{i}{N} \binom{N}{i} P(d_1/i) P(d_2/i) P_2(d). \qquad (7.47)$$

In (7.47), $P_2(d)$ is the probability of incorrectly decoding a codeword with weight d, $M = 3N$, N being the input block length and $d = i + d_1 + d_2$. To calculate the expression of equation (7.47), the distribution of the parity sequences d_1 and d_2 is required. This distribution can be given by [116]

$$P(d_p/i) = \frac{t(N,i,d_p)}{\sum_{d_p} t(l,i,d_p)} = \frac{t(N,i,d_p)}{\binom{N}{i}}, p \in \{1,2\} \qquad (7.48)$$

where $t(l,i,d_p)$, which can be found from the code's transfer function, is the number of paths of length l, input weight i, and output weight d_p, starting and ending in the all zero state. With $P(d_p/i)$, the performance of turbo codes can be studied on various statistical channels by formulating the two-codeword probability $P_2(d)$ for the channel of interest and using (7.47). For the GE channel, $P_2(d)$ is given by equation (7.46).

7.6 Application to GLD Codes

GLD codes independently introduced by Lentmaier [55] and Boutros [56] were presented in the previous chapter. In this section, using the expression of the pairwise error probability derived in Section 7.4, an expression for the union bound on the bit error probability of GLD codes on the GE channel is obtained.

7.6.1 Average Weight Distribution of GLD Codes

The direct computation of the exact weight distribution of a GLD code becomes rapidly intractable when the length of the code N increases. The average weight coefficients of a GLD code can be easily obtained by averaging over all the possible interleavers π_l.

Let us denote by $g(s)$ the moment generating function of the component code, which is the exponential polynomial whose coefficient g_i of degree i is the normalized number of codewords with weight i. As the J super-codes C_j of length N are the direct sum of N/n independent constituent codes C_0, their moment generating function $G_{C_j}(s)$ are simply a power of $g(s)$:

$$G_{C_j}(s) = g(s)^{N/n} = \sum_l Q(l) e^{ls}, \tag{7.49}$$

where $Q(l)$ is the probability that a codeword of C_j has weight l. We assume without loss of generality that all super-codes are built from the same constituent code. Since the total number of codewords in C_j is $(2^k)^{N/n}$, the number of codewords of C_j having weight l is:

$$N_j(l) = 2^{(kN/n)} Q(l). \tag{7.50}$$

Thanks to the fact that $C_1, \cdots, C_J$ are randomly permuted versions of the same super-code, and thus independent, the probability $P(l)$ that a vector of weight l belongs to $C = C_1 \cap \cdots \cap C_J$, is the product of the probabilities that it belongs to each code:

$$P(l) = \left(N_1(l) \Big/ \binom{N}{l} \right)^J. \tag{7.51}$$

Finally, the average number of codewords in C having weight l is:

$$\overline{N(l)} = \binom{N}{l} \times P(l) = \frac{2^{(JkN/n)} Q(l)^J}{\binom{N}{l}^{J-1}}. \tag{7.52}$$

Only GLD codes with two levels are considered for three major reasons. First, it was shown that GLD codes with only $J = 2$ are asymptotically good [55], [56], [66]. Second, two levels is the best choice in terms of rate, as it decreases with J (6.1). Third, the structure, graphical representation and decoding are simpler.

7.6.2 Union Bound on the Bit-Error Probability of GLD Codes on the GE Channel

The average weight distribution of GLD codes (7.52) can be used for computing the average bit error probability of ML decoding of GLD codes on various statistical channels by formulating the two-codeword probability for the channel of interest. Actually, the interleaver acts on all coded bits, so that they are equally protected. Thus we obtain the following Union-Bound (UB) for transmission over the GE channel:

$$\overline{P_{eb}} \leq \sum_{d=1}^{N} \frac{d}{N} \times \overline{N(d)} \times P_2(d), \tag{7.53}$$

where $P_2(d)$ is given by equation (7.46).

7.6.3 GLD Decoder for the Gilbert-Elliott Channel

The decoding of GLD codes is based on iterative soft-input soft-output (SISO) decoding of individual constituent codes. In this chapter, only low rate GLD codes are considered. Hence, exploiting the fact that the constituent code usually has a small code length and high code rate, a trellis-based algorithm can be used to obtain high error-correcting performance with reasonable decoding complexity. In our implementation, we used the trellis-based MAP (maximum a posteriori probability) algorithm [40], also referred to as BCJR, on the syndrome trellis of the constituent code. The algorithm was summarized in Chapter 4.

The modification of the GLD decoding algorithm for the Gilbert-Elliott channel is dependent on what information is available to the GLD decoder. In this chapter, we only consider the case of known channel state. If the state, G or B, is known, then the modification to the GLD decoder is straightforward. Equations (4.87), (4.88), and (4.89) still apply for the decoding of the component codes, where the relevant channel reliability factor is given by:

$$L_c = \frac{4E_s}{N_i}, \quad i \in \{G, B\}. \tag{7.54}$$

7.7 Simulation Model and Numerical Results

For all simulation and numerical results on turbo code, the component encoders are rate one-half, recursive systematic convolutional encoders with memory two (four states) and octal generators $(7,5)$. The frame size is $N = 1024$ bits unless otherwise specified. Perfect side information is assumed and the MAP algorithm is used as decoding algorithm with the branch transition probabilities modified accordingly [121]. A two-level GLD code of length $N = 420$, rate $R = 0.467$, built from the (15, 11, 3) Hamming code is considered. In order to represent a wide range of mobile communication environments, the product $f_D T_s$ was considered as an independent parameter f_m and numerical analysis and simulations were performed for $f_m = 0.1$, and 0.03. Unless otherwise specified, the signal-to-noise ratio threshold on the GE channel was set to 0.1 so that a SNR of 10 dB below the average SNR represents the transition to the bad state.

In Figure 7.6, the bound on the bit error rate of turbo code is evaluated for the above values of the normalized Doppler frequency and for various block lengths. It can be noticed that the bound diverges at low SNR. This behaviour mimics that of similar bounds applied to totally random codes, which turbo codes resemble. This divergence is an artifact of the bound, as the actual performance of the system does not diverge at low SNR. In computing these bounds, we realized that only a handful of terms $i \leq 10, d_p \leq 20$ are needed for convergence, and that this is almost independent of both the values of the frame size N and the normalized Doppler frequency f_m. In Figure 7.7, the effect of the threshold on the accuracy of the bound is assessed. It can be observed that a minor variation in this parameter does not dramatically affect the accuracy of the bound. A comparison of the bound with simulated results in Figures 7.8 and 7.9 confirms the accuracy of the former at medium to high SNR. Thus the bound can be used to predict the system performance in the range where obtaining sufficient data from simulations is impractical and hence to determine the coding gain.

In Figures 7.10 and 7.11, we compare the bound and the simulation results for the GLD code. It can also be observed again that the bound is accurate at medium to high SNR.

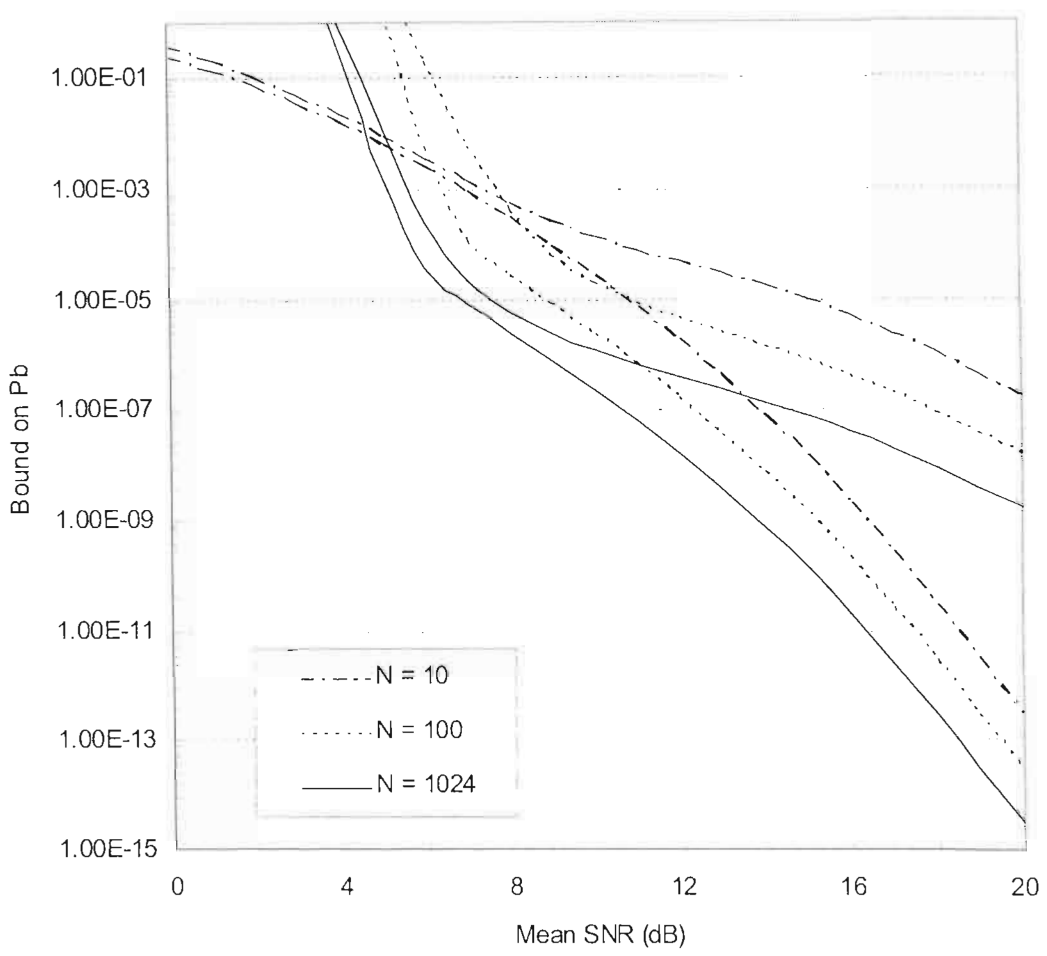

Figure 7.6: Bounds on the bit error rate of 1/3 rate turbo code for various block lengths N, $f_m = 0.1$ (lower group), and $f_m = 0.03$ (upper group).

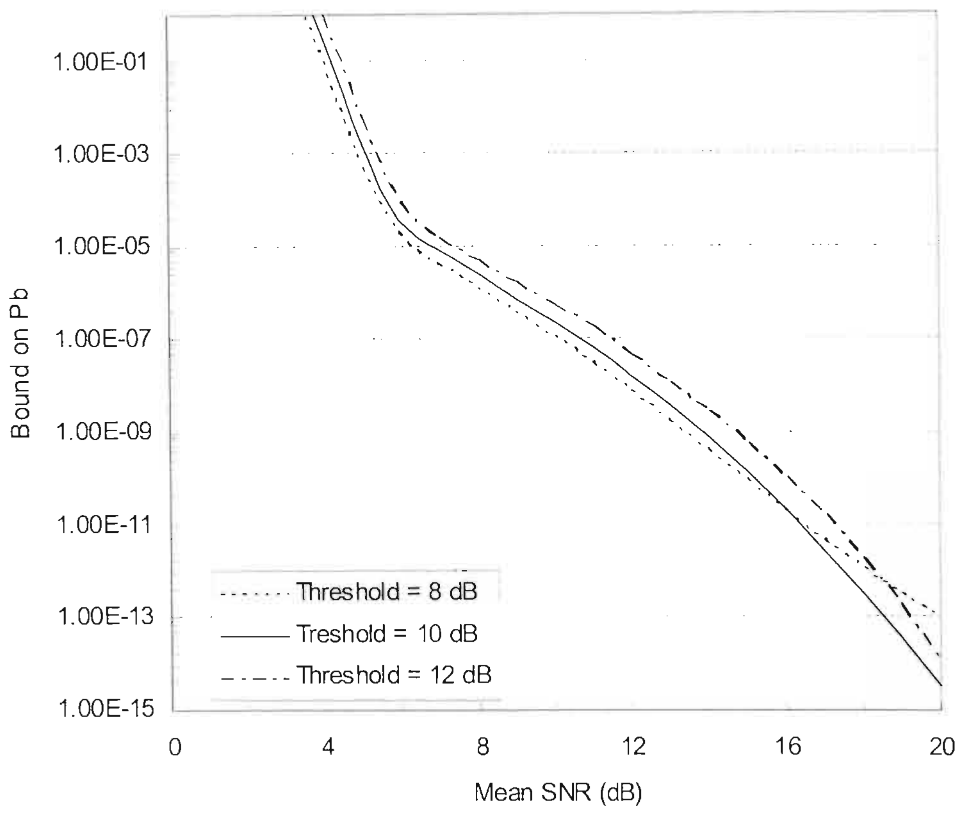

Figure 7.7: Bounds on the bit error rate of 1/3 turbo code for various values of the threshold, $f_m = 0.1$.

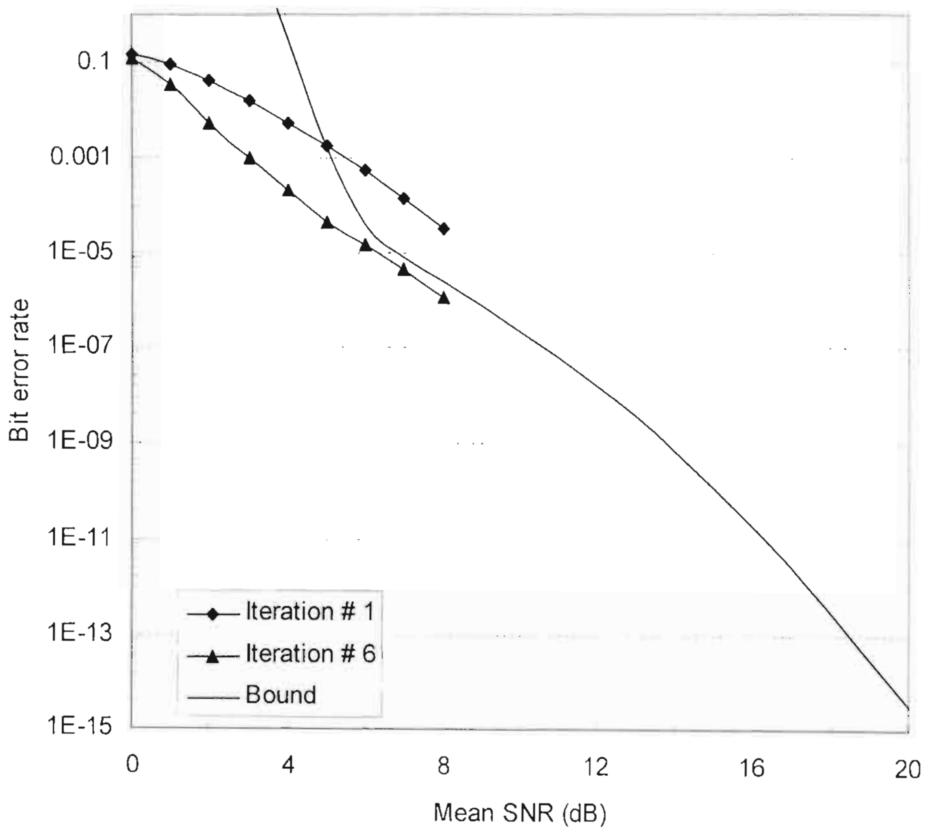

Figure 7.8: Transfer function bound versus simulations results for 1/3 rate turbo code, $f_m = 0.1$.

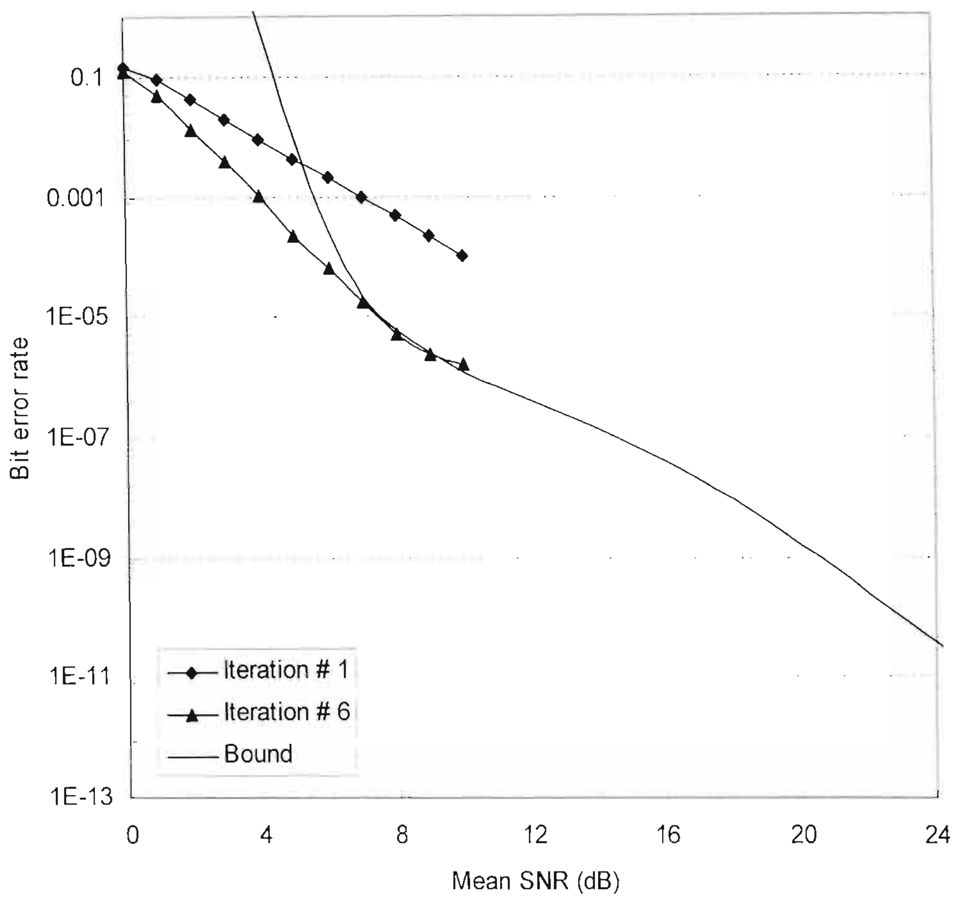

Figure 7.9: Transfer function bound versus simulation results for 1/3 rate turbo code, $f_m = 0.03$.

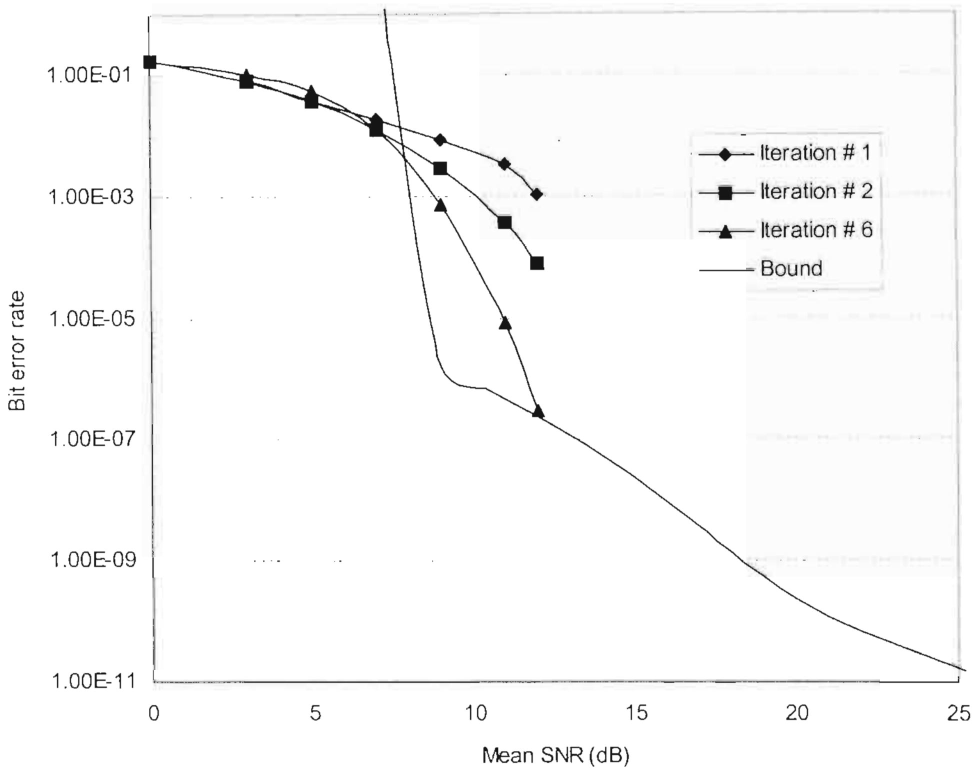

Figure 7.10: Transfer function bound versus simulation results for $J = 2$ GLD code with Hamming (15, 11, 3) constituent code. $N = 420$, $f_m = 0.1$.

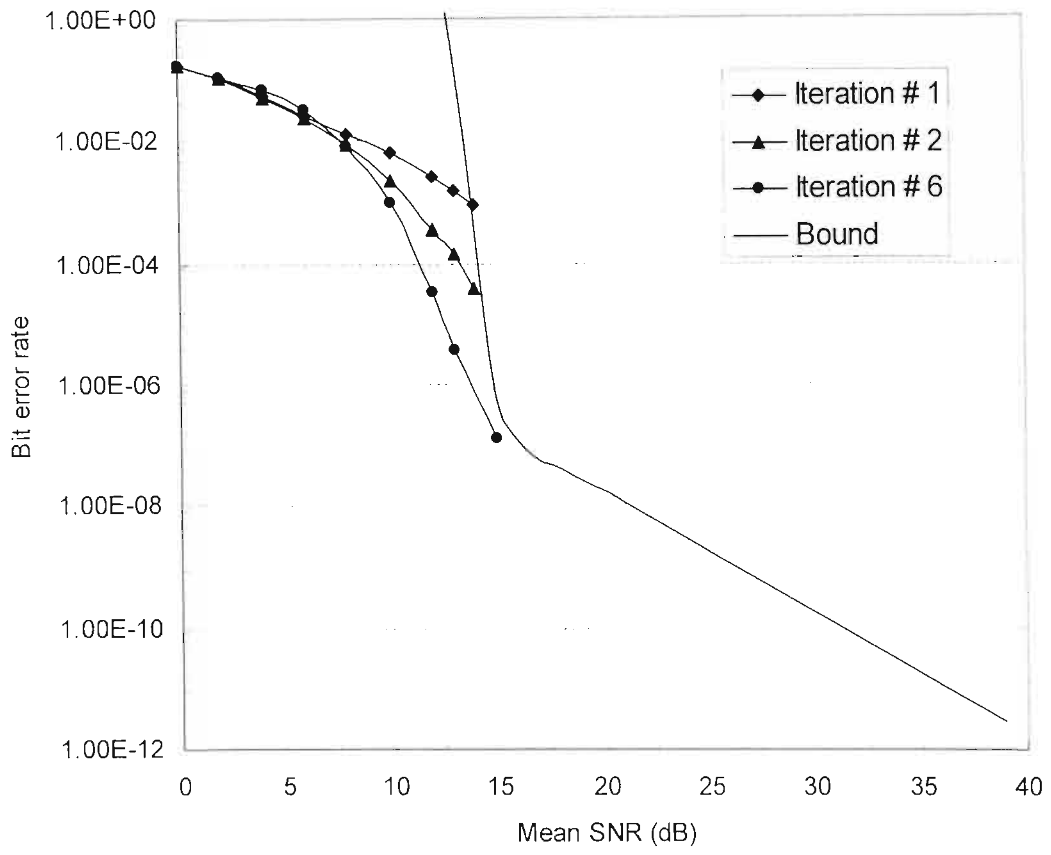

Figure 7.11: Transfer function bound versus simulation results for $J = 2$ GLD code with Hamming (15, 11, 3) constituent code. $N = 420$, $f_m = 0.03$.

7.8 Conclusion

In this chapter, an analytical expression for the pairwise error probability of maximum likelihood decoding of a binary linear code on the GE channel model is derived. This expression can be used to obtain the union bound on the bit error probability of linear codes with maximum-likelihood decoding on the GE channel. The expression is used to obtain union bound for turbo codes and GLD codes on the GE channel. Comparisons between the results obtained by this analytical expression and results obtained through computer simulation showed that, in both cases, the analytical results are accurate in establishing the decoder performance in the range where obtaining sufficient data from simulation is impractical.

CHAPTER 8

CONCLUSION

8.1 Summary

This thesis deals with the design of concatenated codes and decoding algorithms as well as their characterization in fading channel.

Inspired by recent results showing that irregular structure improves performance, the generalization of irregular LDPC codes, called GILD codes, is introduced. GILD codes are both an adaptation of, and an attractive alternative to GLD codes. The high flexibility in selecting the parameters of GILD codes and their better performances and higher rates make them more attractive than GLD codes and hence suitable for small and large block length forward error correcting schemes.

Two new decoding algorithms for LDPC codes are proposed. The first algorithm is a hard-decision method, and the second one is a modification of the first to include reliability information of the received symbols. In principle and in complexity, the algorithms belong to the class of so called bit flipping algorithms. The defining attribute of the proposed algorithms is the bit selection criterion which is based on the fact that, for low density matrices, the syndrome weight increases with the number of errors in average until error weights much larger than half the minimum distance. A loop detection procedure with minimal computational overhead is also proposed that protects the decoding from falling into infinite loop traps. Simulation results show that the proposed algorithms offer an appealing performance/cost trade-offs.

Two new soft-input soft-output iterative decoding algorithms for compound codes based on linear block codes are also proposed. The first algorithm is based on MAP decoding of low-weight subtrellis centered around a generated candidate codeword.

Simulation results indicate that the proposed algorithm provides a significant improvement in error performance over Chase-based algorithm and achieves practically optimal performance with a significant reduction in decoding complexity. The second algorithm is a list based decoding algorithm. It modifies and utilizes the improved Kaneko's decoding algorithm for soft-input hard-output decoding. These hard outputs are converted to soft-decisions using reliability calculations. An important feature of this algorithm is the derivation of a condition to rule out useless test error patterns in the generation of candidate codewords. This rule-out condition reduces many unnecessary decoding iterations and computations. Compared to the Chase-based algorithm, this algorithm is more efficient, has lesser computational complexity for the same performance and provides effective tradeoff between performance and computational complexity.

The characterization of linear codes in channel with memory is also investigated. An analytical expression for the pairwise error probability of maximum likelihood decoding of a binary linear code on a channel with memory modelled by the regenerative GE channel model is derived. This expression is used to obtain the union bound on the bit error probability of linear codes on the GE channel. Comparisons between the results obtained by this analytical expression and results obtained through computer simulations in the case of turbo codes, and GLD codes show that the analytical results are accurate in establishing the decoder performance in the range where obtaining sufficient data from simulation is impractical.

8.2 Future Work

Further investigations need to be carried out. It has been observed in Chapter 4 that, on AWGN channel, there is a loss in a large range of SNR between the simulation results for a fixed GILD code and the upper bound on the ML decoding error of the average ensemble of GILD codes of the same length. It is well known that the interleaver is a key component, which plays an important role in enhancing the performance of compound codes. Future work should aim at designing appropriate interleavers for GILD codes.

Another point that has not been studied is the encoding algorithm of GILD codes. As with GLD codes, the parity check matrix is systematized to produce the associated generator matrix. Even if the systematization is done offline once and that the encoding complexity is not too high for small block length, this method is a drawback of GILD codes and GLD codes compared to other compound codes for medium length. An efficient encoding process taking account the graphical representation of GILD codes may be found, to avoid the explicit construction of the generator matrix.

In Chapter 5, a new low-weight subtrellis based soft-input soft-output decoding algorithm for linear block code suitable for iterative decoding is presented. This algorithm can be improved in several ways. Furthermore a theoretical analysis of the proposed algorithm and its complexity is to be studied. The two new decoding algorithms for LDPC codes introduced in Chapter 3 can also be improved in several ways.

Future research should also extend the work presented in Chapter 7 to derive improved bounds that are accurate at low signal-to-noise ratio.

Appendix A

The MAP Algorithm for SPC Codes

Single parity-check (SPC) code is the simplest code together with the repetition code. If it has length n, the code dimension is $n-1$, and it its rate is $R=(n-1)/n$. The nth bit is parity bit, and it is computed by taking modulo 2 sum of all information bits. Maximum *a posteriori* (MAP) decoding of an SPC code operates on log-likelihood ratios (LLRs) of each received bit. Based on the received value corresponding to each bit, channel LLRs are computed for each code bit position i, with $0 \le i < n$:

$$L_c(c_i) = \ln \frac{P(c_i = 1 \mid y_i)}{P(c_i = 0 \mid y_i)} \tag{A.1}$$

MAP decoding algorithm computes the so-called *extrinsic LLR* for each bit. Expression for the extrinsic LLR for code bit at position i is similar to the one for the channel LLR. The difference is that probabilities are conditioned on all components of the received sequence $\boldsymbol{y}$, except the component y_i that corresponds to the ith code bit itself.

Consider first the case when we have only two information bits, c_0 and c_1, and parity bit c_2. Then, $c_2 = c_0 \oplus c_1$, where $\oplus$ indicates addition modulo 2. The extrinsic LLR for c_0 is obtained as follows:

$$\begin{aligned} L_{ex}(c_0) &= \ln \frac{P(c_1 \oplus c_2 = 1 \mid y_1, y_2)}{P(c_1 \oplus c_2 = 0 \mid y_1, y_2)} \\ &= \ln \frac{e^{L_c(c_1)} + e^{L_c(c_2)}}{1 + e^{L_c(c_1)} e^{L_c(c_2)}} \\ &= -2 \tanh^{-1}\left(\tanh\left(\tfrac{L_c(c_1)}{2}\right) \cdot \tanh\left(\tfrac{L_c(c_2)}{2}\right)\right) \end{aligned} \tag{A.2}$$

The two last lines of (A.2) are obtained after some mathematical manipulation.

Extrinsic LLRs for bits c_1 and c_2 are obtained in the same way. For each bit, channel LLR is improved by adding extrinsic LLR to it. Based on the sum

$L(c_i) = L_c(c_i) + L_{ex}(c_i)$ decision is made for bit c_i. If $L(c_i) > 0$, the hard decision for the ith bit is 1, $z_i = 1$, and if $L(c_i) < 0$, $z_i = 0$.

Good approximation of the extrinsic LLR in (A.2) is given by the following expression:

$$L_{ex}(c_0) \approx (-1) \cdot \text{sign}(L_c(c_1)) \cdot \text{sign}(L_c(c_2)) \cdot \min\{|L_c(c_1)|, |L_c(c_2)|\}. \qquad \text{(A.3)}$$

The exact extrinsic LLR in (A.2) can be generalized to an SPC code of arbitrary length n:

$$L_{ex}(c_i) = 2 \cdot (-1)^n \cdot \tanh^{-1}\left(\prod_{j=0, j \neq i}^{n-1} \tanh\left(\frac{L_c(c_j)}{2}\right)\right). \qquad \text{(A.4)}$$

The generalization of the approximation for extrinsic LLR given in (A.3) to an SPC of length n is:

$$L_{ex}(c_i) \approx (-1)^n \cdot \prod_{j=0, j \neq i}^{n-1} \text{sign}(\mathrm{L}_c(c_j)) \cdot \min_{0 \leq j < n, j \neq i} \{|L_c(c_j)|\}. \qquad \text{(A.5)}$$

Appendix B

This appendix derives an upper bound on the Hamming distance between two distinct codewords. For any two non equivalent error patterns, $\boldsymbol{e}^{(r)}$ and $\boldsymbol{e}^{(j)}$, define $\boldsymbol{z}^{(r)} = \boldsymbol{y}^{H} \oplus \boldsymbol{c}^{(r)} \oplus \boldsymbol{e}^{(r)}$ and $\boldsymbol{z}^{(j)} = \boldsymbol{y}^{H} \oplus \boldsymbol{c}^{(j)} \oplus \boldsymbol{e}^{(j)}$. First we note that $w_H\left(\boldsymbol{z}^{(r)}\right) \leq t$ and $w_H\left(\boldsymbol{z}^{(j)}\right) \leq t$ because of bounded distance decoding.

$$\begin{aligned}
d_H\left(\boldsymbol{c}^{(r)},\boldsymbol{c}^{(j)}\right) &= d_H\left(\boldsymbol{z}^{(r)} \oplus \boldsymbol{e}^{(r)}, \boldsymbol{z}^{(j)} \oplus \boldsymbol{e}^{(j)}\right) \\
&= w_H\left(\boldsymbol{z}^{(r)} \oplus \boldsymbol{e}^{(r)}, \boldsymbol{z}^{(j)} \oplus \boldsymbol{e}^{(j)}\right) \\
&\leq w_H\left(\boldsymbol{e}^{(r)} \oplus \boldsymbol{e}^{(j)}\right) + w_H\left(\boldsymbol{z}^{(r)}\right) + w_H\left(\boldsymbol{z}^{(j)}\right) \\
&\leq d_H\left(\boldsymbol{e}^{(r)},\boldsymbol{e}^{(j)}\right) + 2t\,. \qquad \text{(B.1)}
\end{aligned}$$

Appendix C

Derivation of $P_d(d_B)$ and $P_d(d_G)$ for the Pairwise Error Probability

Recall that we want to find the probability of being in state B exactly d_B out of d times, $P_d(d_B)$, and the probability of being in state G exactly d_G out of d times, $P_d(d_G)$.

Consider $P_d(d_B)$. For the cases $d = 0$ and $d = n$, the result is trivial, since $d = 0$ means that the channel starts in the good state and never leaves it, which will happen with probability $(1-b)^{d-1}\pi_G$, and $d = n$ means that the channel starts in the bad state and remains there, which will happen with probability $(1-g)^{d-1}\pi_B$. Henceforth, we may therefore assume that $1 \le d \le n$.

The channel behaviour can be depicted by one of the following four cases:

- The channel starts in the good state and ends in the good state, which will happen with probability $P_d(d_B/GG)$.
- The channel starts in the good state and ends in the bad state, which will happen with probability $P_d(d_B/GB)$.
- The channel starts in the bad state and ends in the bad state, which will happen with probability $P_d(d_B/BB)$.
- The channel starts in the bad state and ends in the good state, which will happen with probability $P_d(d_B/BG)$.

$P_d(d_B)$ is found by summing the conditional probabilities, weighted appropriately, i.e.

$$P_d(d_B) = \left[P_d(d_B/GG) + P_d(d_B/GB)\right]\pi_G + \left[P_d(d_B/BG) + P_d(d_B/BB)\right]\pi_B. \quad \text{(C.1)}$$

Now, consider $P_d(d_B/GG)$, and let i be the number of sojourns in the good state. The number of sojourns in the bad state is then i-1. If the channel is in the bad state exactly d_B out of d times, then it is in the good state exactly d-d_B. Clearly $i \le d - d_B$, otherwise the channel would be in the good state too many times, and $i - 1 \le d_B$, otherwise the channel would be in the bad state too many times. The probability, apart from d_B and d, depends on the number of times the channel state changes, not on the exact behaviour of the channel. In this case, we will have:

- i-1 transitions from a good state to a bad state, each of which will happen with probability b.
- d-d_B-i transitions from a good state to a good state, each of which will happen with probability 1-b.
- i-1 transitions from a bad state to a good state, each of which will happen with probability g.
- d_B-i+1 transitions from a bad state to a bad state, each of which will happen with probability 1-g.

The probability for this specific channel behaviour is given by:

$$(1-b)^{d-d_B-i}\,b^{i-1}\,(1-g)^{d_B-i+1}\,g^{i-1}. \tag{C.2}$$

Now, since the number of ways d_B can be expressed as a sum of i-1 positive intergers is

$$\binom{d_B-1}{i-2}, \tag{C.3}$$

and the number of ways that d-d_B can be expressed as a sum of i positive integers is

$$\binom{d-d_B-1}{i-1}, \tag{C.4}$$

we have that

$$P_d\left(d_B/GG\right)=\sum_{i=2}^{\min(d_B+1,d-d_B)}\binom{d-d_B-1}{i-1}\binom{d_B-1}{i-2}(1-b)^{d-d_B-i}\,b^{i-1}\,(1-g)^{d_B-i+1}\,g^{i-1}. \tag{C.5}$$

By using similar arguments, it is straightforward to derive the other conditional probabilities, and hence $P_d\left(d_B\right)$, which concludes the proof.

$P_d\left(d_G\right)$ is also derived similarly.

References

[1] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal, 1948.*

[2] R. G. Gallager, Low-density parity-check codes, Cambridge, MA USA, MIT Press, 1963. Also available online at http://justice.mit.edu/people/pubs/ldpc.ps.

[3] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. On Information Theory*, vol. 8, no. 1, pp. 21-28, Jan. 1962.

[4] C. Berrou, A. Glavieux, and P. Thitimajishima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. 1993 Int. Conf. on Communications*, pp. 1064 –1070.

[5] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261-1271, Oct. 1996.

[6] V. Zyablov and M. Pinsker, "Estimation of the error-correction complexity of Gallager low-density codes," *Problemy Peradachi Informatsii*, vol. 11, pp. 23-26, Jan. 1975.

[7] G. A. Margulis, "Explicit construction of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, pp. 71-78, 1982.

[8] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 553-547, Sept. 1981.

[9] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, vol. 46, no. 6, pp. 1710-1722, Nov. 1996.

[10] D. J. C MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electronics Letters*, vol. 32, pp. 1645-1646, Aug. 1996.

[11] N. Wiberg, *Codes and Decoding on General graphs*, PhD thesis, Linköping University, S-581 83, Linköping Sweden, 1996.

[12] N. Sourlas, "Spin-glass models as error-correcting codes," *Nature*, vol. 339, pp. 693-695, June 1989.

[13] I. Kantor and D. Saad, "Error-correcting codes that nearly saturate Shannon's bound," *Phys. Rev. Lett.*, vol. 83, pp. 2660-2663, 1999.

[14] D. J. C. MacKay, "Good Error-Correcting Codes based on Very Sparse Matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399-431, Mar. 1999.

[15] J. Peral, "Fusion, propagation, and structuring in belief networks," *Artif. Intell.*, vol. 29, pp. 241-288, 1986.

[16] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation," in *Proc. International Symposium on Information Theory* (ISIT'98), p. 117, Aug. 16-21, 1998. Also available online at http://www.icsi.berkerly.edu/luby/PAPERS/belerr.ps.

[17] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, "Analysis of low-density codes and improved designs using irregular graphs," Proc. Of the 30th ACM STOC, May 23-26, 1998. Also available online at http://www.icsi.berkerly.edu/luby/PAPERS/anaerr.ps.

[18] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 585-598, Feb. 2001.

[19] T. Richardson, A. Shokrollahi and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.

[20] D. J. C. MacKay, S. T. Wilson, M. C. Davey, "Comparison of constructions of irregular Gallager codes," *IEEE Trans. Comm*, vol. 47, pp. 1449-1454, Oct. 1999.

[21] S. –Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Comm. Letters*, vol. 5, no. 2, Feb. 2001.

[22] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp. 2711-2736, Nov. 2001.

[23] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed., MIT Press, Cambridge, MA., 1972.

[24] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice Hall, Englewood Cliffs, New Jersey, 1983.

[25] H. Tang, J. Xu, Y. Kou, S. Lin, and K. Abdel-Ghaffar, "On algebraic construction of Gallager and circulant low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 1269-1279, June. 2004.

[26] H. Tang, J. Xu, S. Lin, K. Abdel-Ghaffar, "Codes on finite geometries," accepted for publication in *IEEE Trans. Inform. Theory*, 2004.

[27] R. M. Tanner, D. Srkdhara, and T. Fuja, "A class of group-structured LDPC codes," *Proc. of ISTA 2001*, Ambleside, England, 2001.

[28] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "Construction of low-density parity-check codes based on Reed-Solomon codes with two information symbols," *IEEE Commun. Lett.*, vol. 7, no. 7, pp. 317-319, July 2003.

[29] V. D. Kolesnik, "Probability decoding of majority codes," *Prob. Peredachi Inform.*, vol. 7, pp. 3-12, July 1971.

[30] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711-2736, Nov. 2001.

[31] Z. Liu and D. A. Pados, "A decoding algorithm for finite geometry LDPC codes," submitted to *IEEE Trans. Commun.*

[32] F. R. Kschischang, B. J. Frey and H. –A. Loeliger, "Factor Graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498-519, Feb. 2001.

[33] R. Lucas, M. Fossorier, Y. Kou, and S. Lin, "Iterative decoding of one-step majority logic decodable codes based on belief propagation," *IEEE Trans. Commun.*, vol. 48, pp. 931-937, June 2000.

[34] J. Pearl, *Probabilistic Reasoning in Intelligence Systems: Netwroks of Plausible Inference*, Morgan Kaufmann, San Mateo, 1988.

[35] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity-check codes," *IEEE Trans. Commun.*, vol. 47, pp. 673-680, May 1999.

[36] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, pp. 406-414, March 2002.

[37] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IRE Trans.*, IT-4, pp. 38-49, Sept. 1954.

[38] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *IRE Trans.*, EC-3, pp. 6-12, Sept. 1954.

[39] J. L. Massey, *Threshold Decoding*, Cambridge, MA: MIT Press, 1963.

[40] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284-287, March 1974.

[41] T. Beth, D. Jungnickel, and H. Lenz, Design Theory, Cambridge, U.K: Cambridge Univ. Press, 1986.

[42] B. Vasic and O. Milenkovic, "Combinatorial construction of low-density parity-check codes for iterative decoding," *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 1156-1176, June 2004.

[43] B. Ammar, B. Honary, Y. Kou, J. Xu, and S. Lin "Construction of low-density parity-check codes based on balanced incomplete block designs," *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 1257-1268, June 2004.

[44] S. Lin, L. Chen, I. Djurdjevic, J. Xu, "Near Shannon limit quasi-cyclic low-density parity-check codes," *in Proc. IEEE GlobeCom'2003*, pp. 2030-2035, San Francisco, CA, Dec. 2003.

[45] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near Shannon limit quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 52, no. 7, July 2004.

[46] G. A. Margulis, "Explicit construction of graphs without short cycles and low-density codes," *Combinatorica*, 2(1): 71-78, 1982.

[47] J. Rosenthal and P. O. Vontobel, "Construction of LDPC codes using Ramanujan graphs and ideas from Margulis," *in Proc. The 38th Allerton Conf. On Commun., Control and Computing*, pp. 248-257, Monticello, IL, Oct. 4-6, 2001.

[48] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan graphs," *Combinatorica*, 8(3):261-277, 1988.

[49] P. Sarnak, *Some applications of modular forms*, Cambridge University Press, Cambridge, 1990.

[50] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressive edge-growth Tanner graph," *IBM Research, Zurich Research Laboratory*, 8803, Rüschlikon, Switzerland.

[51] P. Erdös and H. Sachs, Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl, *Wiss. Z. Univ. Hall Martin Luther Univ. Halle-Wittenberg Math. – Natur. Reine*, 12:251-257, 1963.

[52] D. A. Spielman, "Finding good LDPC codes," *36th Annual Allerton Conference on Communication, Control, and Computing*, 1998. http://www-math.mit.edu/spielman/theory/research_ecc.html.

[53] S. Y. Chung, T. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638-656, Jan. 2000.

[54] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity-check codes, tech. Rep., Bell Labs, Lucent Technologies, Murray Hill, NJ 07947, Apr. 1999.

[55] M. Lentmaier and K. Sh. Zigangirov, "On generalized low-density check codes based on Hamming component codes," *IEEE Commun. Lett.,* vol. 3, no. 8, pp. 248-250, August 1999.

[56] J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (Tanner) codes," in *Proc. ICC'99*, Houston, July 1999.

[57] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638-656, Feb. 2000.

[58] J. –F. Cheng and R. J. McEliece, "Some high-rate near capacity codecs for the Gaussian channel," in *34th Allerton Conf. Communications, Controls and Computing*, 1996.

[59] M. C. Davey and D. J. C. Mackay, "Low-density parity-check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, pp. 165-167, June 1998.

[60] T. Richardson and R. Urbanke, "The capcity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599-618, Feb. 2001.

[61] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann "Practical loss-resilient codes," in *Proc. 29th Annu. Symp. Theory of Computing,* 1957, pp. 150-159.

[62] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, "Analysis of random processes via and-or trees," in *Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms*, 1998, pp. 364-373.

[63] N. Wiberg, H. –A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecommun.*, vol. 6, pp. 513-526, Sept. 1995.

[64] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569-584, Feb. 2001.

[65] D. J. C. MacKay. (1998) Turbo codes are low-density parity-check codes. Available on line at: http://www.cs.toronto.edu/~mackay/abstracts/-turbo-ldpc.html.

[66] O. Pothier, *Compound codes based on graphs and their iterative decoding*, Ph.D. thesis, Ecole Nationale Superieure des Telecommunications, Jan. 2000.

[67] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann "Practical loss-resilient codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569-584, Feb. 2001.

[68] F.J. MacWilliams, N.J.A. Sloane: The theory of error-correcting codes, North-Holland, Englewood Cliffs, eighth impression, 1993.

[69] E. R. Berlekamp: Algebraic Coding Theory, Aegean Park Press, Revised 1984 Edition.

[70] M. Abramowitz and I. Stegun, *Handbook of mathematical functions*, Dover publications Inc, New York, ninth printing, 1972, p. 257.

[71] A. J. Viterbi, J. K. Omura, *Principles of digital communications and coding*, McGraw-Hill, 1979.

[72] R. G. Gallager, *Information theory and reliable communication*, John Wiley & Sons, 1968.

[73] T. M. Duman, and M. Saheli, "New performance bounds for turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 6, June 1998, pp. 717-723.

[74] E. R. Berlekamp, "The technology of error-correcting codes," *Proceeding of the IEEE*, vol. 68, no. 5, pp. 564-593, May 1980.

[75] G. Poltyrev, "Bounds on the decoding error probability of binary linear codes via their spectra," *IEEE Trans. On Inf. Theory*, vol. 40, no. 4, July 1994, pp. 1284-1292.

[76] H. Herzberg and G. Poltyrev, "The error probability of m-ary PSK block coded modulation schemes," *IEEE Trans. Commun.*, vol. 44, no. 4, April 1996, pp. 427-433.

[77] I. Sason and S. Shamai (Shitz), "Bounds on the error probability of ML decoding of block and turbo-block codes," *Annales des Telecommunications*, vol. 5, no. 3-4, Mars-April 1999, pp. 183-200.

[78] I. Sason and S. Shamai (Shitz), "Improved upper bounds on the ML decoding error probability of parallel and serial concatenated turbo codes via their ensemble distance spectrum," *IEEE Trans. On Inf. Theory*, vol. 46, no. 1, January 2000, pp. 24-47.

[79] A. Lafourcade and A. Vardy, "Asymptotically good codes have infinite trellis complexity," *IEEE Trans. on Inf. Theory*, vol. 41, pp. 555-559, Mar. 1995.

[80] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. on Inf. Theory*, vol. 18, pp. 170-182, Jan. 1972.

[81] G. D. Forney Jr., "Generalized minimum distance decoding," *IEEE Trans. on Inf. Theory*, vol. 12, pp. 125-131, Apr. 1966.

[82] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. on Inf. Theory*, vol. 41, pp. 1379-1396, Sept. 1995.

[83] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa, "An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder," *IEEE Trans. Inform. Theory*, vol. 40, pp. 320-327, Mar. 1994.

[84] M. P. C. Fossorier and S. Lin, "Soft-input soft-output decoding of linear block codes based on ordered statistics," *Globecom*, 1998.

[85] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 1003-1010, Aug. 1998.

[86] P. A. Martin and D. P. Taylor, "On adaptive reduced-complexity iterative decoding," *Proc. 2000 IEEE Global Telecomm. Conf.* (GLOBECOM'00), pp. 772-776, 2000.

[87] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellisses and trellis-based Decoding Algorithm for Linear Block Codes*, Kluwer Academic Publishers, 1998.

[88] A. Picart and R. Pyndiah, "Adapted iterative decoding of product codes," in *Proc. Globecom,* 1999, pp. 2357-2362.

[89] P. A. Martin and D. P. Taylor, "On multilevel codes and iterative multistage decoding," *IEEE Trans. Commun.*, vol. 49, No. 11, pp. 1916-1925, Nov. 2001.

[90] T. Johanson and K. Zigangirov, "A simple one sweep algorithm for optimal APP symbol decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. 44, pp. 3124-3129, Nov. 1998.

[91] T. Kaneko, T. Nishijima, and S. Hirasawa, "An improvement of soft-decision likelihood decoding algorithm using hard-decision bounded-distance decoding" *IEEE Trans. Inform. Theory*, vol. 43, pp. 1314-1319, July. 1997.

[92] S. Dave, J. Kim and S. C. Kwatra, "An efficient decoding algorithm for block turbo codes," *IEEE Trans. Commun.*, vol. 49, pp. 41-46, Jan. 2001.

[93] T. Y. Hwang, "Decoding linear block codes for minimizing word error rate," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 733-737, Nov. 1979.

[94] G. D. Forney, Jr., *Concatenated codes* (MIT research monograph no. 37). Cambridge, MA: MIT Press, 1966.

[95] H. T. Moorthy, S. Lin, and T. Kasami, "Soft-decision decoding of binary linear block codes based on an iterative search algorithm," IEEE *Trans. Inform. Theory*, vol. 43, pp. 1030-1040, May 1997.

[96] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. on Inf. Theory*, vol. 44, pp. 429-445, Mar. 1996.

[97] T. M. N. Ngatched and F. Takawira, "A simple stopping criterion for turbo decoding," *IEE Electronic Letters*, vol. 37, no. 22, pp. 1350-1351, Oct. 25, 2001.

[98] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, pp. 1117-1120, Aug. 1999.

[99] Yufei Wu, B. D. Woerner and W. J. Ebel, "A simple stopping criterion for turbo decoding" *IEEE Communications Letters*, vol. 4, No. 8, pp. 258-260, August 2000.

[100] R. H. Morelos-Zaragoza, *The art of error correcting coding*, First edition, John Wiley & Sons, 2002.

[101] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253-1265, Sept. 1960.

[102] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *Bell Syst. Tech. J.*, vol. 42, pp. 1977- 1997, Sept. 1963.

[103] H. S. Wang and N. Moayeri, "Finite-state Markov channel—A useful model for radio communication channels," *IEEE. Trans. Veh. Technol.*, vol. 44, pp. 163-171, Feb. 1995.

[104] H. S. Wang and P. C. Chang, "On verifying the first-order Markovian assumption for a Rayleigh fading channel model," *IEEE trans. Veh. Technol.*, vol. 45, pp. 353-357, May 1996.

[105] L. Ahlin, "Coding methods for the mobile radio channel," presented at *Nordic Seminar on Digital Land Mobile Communications*, Espoo, Findland, Feb. 1985.

[106] G. Sharma, A. Dholakia, and A. A. Hassan, "Simulation of error trapping decoders on a fading channel," in *Proc. 1996 IEEE Vehicular Technology Conf.*, Atlanta, GA, Apr. 28–May 1, 1996, pp. 1361-1365.

[107] G. Sharma, A. A. Hassan and A. Dholakia, "Performance evaluation of burst-error correcting codes on a Gilbert-Elliott channel," *IEEE Trans. Comm.* Vol. 46, No. 7, July 1998.

[108] J. R. Yee and E. J. Weldon, "Evaluation of the performance of error-correcting codes on a Gilbert Channel," *IEEE Trans. Comm.* Vol. 43, No. 8, August 1995.

[109] L. Wilhelsson and L. B. Milstein, "On the effect of imperfect interleaving for the Gilbert-Elliott channel," *IEEE Trans. Comm.* Vol. 47, No. 5, pp. 681-688, May 1999.

[110] B. Wong and C. Leung, "On computing undetected error probabilities on the Gilbert channel," *IEEE Trans. Commun.*, vol. 43, pp. 2657-2661, Nov. 1995.

[111] W. C. Jakes, Ed., *Microwave Mobile Communications*. New York: IEEE Press, 1974.

[112] J.G Proakis, *Digital Communications*. New York: McGraw-Hill, 1989.

[113] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proc. GLOBECOM* 94, pp. 1298-1303, Dec. 1994.

[114] S. Benedetto and G. Montorsi, "Unveilling turbo codes: some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, No. 2, pp. 409-428, March 1996.

[115] S. Benedetto and G. Montorsi, "Performance evaluation of turbo-codes," *Electronics Letters*, vol. 31, No. 3, pp. 163-165, Feb. 1995.

[116] D. Divsalar, S. Dolinear, F. Pollara and R. J. McEliece, "Transfer function bounds on the performance of turbo codes," *JPL TDA progress Reports*, 42-122, pp. 44-45, Aug. 1995.

[117] I. Sason and S. Shamai (Shitz), "Improved upper bounds on the decoding error probability of parallel and serial concatenated turbo codes via their ensemble

distance spectrum," *Proceeding of IEEE International Synposium on Information Theory*, p. 30, MIT, Cambridge, MA, USA, Aug. 1998.

[118] Y. V. Sirid, "Weight distributions and bounds for turbo codes," *European Transaction on Telecommunication and Related Technologies*, vol. 6, No. 5, pp. 543-555, Sept. – Oct. 1995.

[119] E. K. Hall and S. G. Wilson, "Designing and analysis of turbo codes on Rayleigh fading channels," *IEEE Journal of Selected Area in Communications*, pp. 160-174, Feb. 1998.

[120] J. Y. Kim, "Performance of OFDM/CDMA system with turbo coding in a multipath fading channel," *IEEE Trans. On Consumer Electronics*, vol. 45, No. 2, pp. 372-379, May 1999.

[121] J. H. Kang, W. E. Stark and A. O. Hero, "Turbo codes for fading and burst channels," *Proceedings of the Communications Theory Miniconference*, Globecom'99, pp. 40-45, Nov. 1998.

[122] J. Garcias-Frias and J. D. Villasenor, "Turbo decoding of Gilbert-Elliott channels," *IEEE Trans. Commun.*, vol. 50, No. 3, pp. 357-363, Mar. 2002.

[123] R. J. McEliece, "How to compute weight enumerators for convolutional codes," *Communications and coding*, pp. 121-141, M. Darnell and B. Honary, eds. Tauton Sumerest, England: Research Studies Press Ltd., 1998.

[124] S. A. Barbulescu, *Iterative decoding of turbo and other concatenated codes*, Ph.D. thesis, School of Electrical Engineering, Faculty of Engineering, University of South Australia, Feb. 1996.

[125] E. K. Hall and S. G. Wilson, "Design and analysis of turbo codes for noncoherent channels," in *Proc. IEEE Communication Theory Mini-conference*, Phoenix, USA, Nov. 1997, pp. 66-70.

[126] P. Frenger, "Turbo decoding on Rayleigh fading channel with noisy channel estimates," in *Proc. IEEE VTC'99, Houston*, TX, May 1999, pp. 884-888.

[127] O. Edfors, M. Sandell, JJ. Van de Beek, S.K. Wilson, and P.O. Brjesson, "OFDM channel estimation by singular value decomposition," in *Proc. IEEE Vehicular technology Conf., Atlanta, GA*, Apr. 28-May 1, 1996, pp. 924-927.

[128] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 76-80, Jan. 1978.

[129] G. D. Forney, Jr., "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 1152-1187, Sept. 1988.

[130] A. Vardy and Y. Be'ery, "Maximum likelihood soft-decision decoding of BCH codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 546-554, Mar. 1994.

[131] D. J. Muder, "Minimal trellises for block codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1049-1053, Sept. 1988.

[132] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On the optimum bit orders with respect to the state complexity of trellis diagrams of binary linear codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 242-245, Jan. 1993.

[133] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On complexity of trellis structure of linear block codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1057-1064, May. 1993.

[134] Y. Berger and Y. Be'ery, "Bounds on the trellis size of linear block codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 203-209, Jan. 1993.

[135] V. V. Zyablov and V. R. Sideroko, "Bounds on the complexity of trellis decoding of linear block codes," *Probl. Pred. Inform.*, vol. 29, July/Sept. 1993.

[136] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On structural complexity of the L-section minimal trellis diagrams for binary linear block codes," *IEICE Trans. Fundamentals of Elec., Comm. Comp. Sci.*, vol. E76-A, no. 9, pp. 1411-1421, Sept. 1993.

[137] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On branch labels of parallel components of the L-section minimal trellis diagrams for binary linear block codes," *IEICE Trans. Fundamentals of Elec., Comm. Comp. Sci.*, vol. E77-A, no. 9, pp. 1058-1068, June. 1994.

[138] G. D. Forney, Jr., "Dimension/length profiles and trellis complexity of linear block codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1741-1772, Nov. 1994.

[139] G. D. Forney, Jr. and M. D. Trott, "The dynamics of group codes: State spaces, trellis diagrams and canonical encoders," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1491-1513, 1993.

[140] A. Barg, E. Krouk, and H. C. A. van Tilborg, "On the complexity of minimum distance decoding of long linear codes," *IEEE Trans. Inform. Theory*, vol. IT-45, pp. 1392-1405, July 1999.

[141] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal a posteriori algorithms suitable for turbo decoding," *European Trans. On Telecommunications*, vol. 8, no. 2, Mar-Apr. 1997, pp. 119-125.

[142] J. Hagenauer, E. Offer, L. Papke, "A viterbi algorithm with soft-decision outputs and its applications," Proceedings of GLOBECOM'89, Dallas, Texas, pp. 47.1.1-47.1.7, Nov. 1989.

[143] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, 1993.

[144] A. Papoulis, *Probability, Random variables, and stochastic Processes*, McGraw-Hill, 3rd edition, 1991.

[145] S. K. Wilson, *Digital audio broadcasting in a fading and dispersive channel*, PhD thesis, Dept. of Electrical Engineering, Stanford University, Aug. 1994.

[146] A. Vardy and Y. Be'ery, "Maximum likelihood soft-decision decoding of BCH codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 546-554, Mar. 1994.

[147] Y. Berger and Y. Be'ery, "Soft trellis-based decoder for linear block codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 764-773, May 1994.

[148] L. Ekroot and S. Dolinear, "A decoding of block codes," *IEEE Trans. Commun.*, vol. 50, No. 9, pp. 1052-1056, Sep. 1996.

[149] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, "Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. 39, no. 5, pp. 1541-1523, Sep. 1993.

[150] A. Lafourcade and V. Vardy, "Optimum sectionalization of a trellis," *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 689-702, May 1996.

[151] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 4, pp. 1072-1092, Jul. 1996.

[152] T. Koumoto, T. Takata, T. Kasami and S. Lin, "A low-weight trellis based iterative soft-decision decoding algorithm for binary linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 1072-1092, Mar. 1999.

[153] T. M. N. Ngatched and F. Takawira, "Decoding of GLD Codes in a Rayleigh Fading Channel With Imperfect Channel Estimates," in *Proc. South African Telecommunication Networks & Applications Conference*, Drakensberg, South Africa, 1-4 Sept. 2002.

[154] T. M. N. Ngatched and F. Takawira, "Generalized Irregular low-Density (Tanner) Codes based on Hamming Component Codes," in Proc. *IEEE AFRICON 2002*, pp. 193-196, Oct. 2002.

[155] T. M. N. Ngatched and F. Takawira, "Efficient Decoding of Generalized Low-Density Parity-Check Codes Based on Long Component codes," in Proc. *WCNC 2003, IEEE Wireless Communications and Networking Conference*, vol. 4, no. 1, pp. 705 - 710, Mar. 2003.

[156] T. M. N. Ngatched and F. Takawira, "Error-Correcting Codes Based on Irregular Graphs," in *Proc. South African Telecommunication Networks & Applications Conference*, George, South Africa, 8-10 Sept. 2003.

[157] T. M. N. Ngatched and F. Takawira, "A low-weight trellis based soft-input soft-output decoding algorithm for binary linear block codes with application to generalized Irregular low-density codes," in Proc. *GLOBECOM* 2004, *IEEE Global Telecommunications Conference*, vol. 4, no. 1, pp. 705-710, Nov. 29 - Dec. 3, 2004.

[158] T. M. N. Ngatched and F. Takawira, "An Iterative Soft-Input Soft-Output Decoding Algorithm for Linear Block Codes," in *Proc. South African Telecommunication Networks & Applications Conference*, Spier Wine Estate, Western Cape, South Africa, 6-8 Sept. 2004.

[159] T. M. N. Ngatched and F. Takawira, "Pairwise error probability on the Gilbert-Elliott channel with application to turbo codes," in proc. *IEEE AFRICON* 2004, pp. 279-284, Sept. 2004.

[160] T. M. N. Ngatched, M. Bossert, and A. fahrner, "Two decoding algorithms for low-density parity-check codes," accepted for presentation at *IEEE International Conference on Communications (ICC)* 2005, 16 – 20 May 2005, Seoul, Korea.

[161] T. M. N. Ngatched and F. Takawira, "Improved generalized low-density parity-check codes using irregular graphs," *SAIEEE Transactions*, vol. 94: pp. 43-49.

[162] T. M. N. Ngatched and F. Takawira, "High-performance low-complexity decoding of generalized low-density (GLD) parity-check codes," submitted to *IEEE Trans. Commun.*, Aug. 2003.

[163] T. M. N. Ngatched and F. Takawira, "Generalization of irregular low-density parity-check codes," submitted to *IEEE Trans. Commun.*, Dec. 2003.

[164] T. M. N. Ngatched and F. Takawira, "Union bound for binary linear codes on the Gilbert-Elliott channel with application to turbo-like codes," submitted to *IEEE Trans. Veh. Technol.*, June. 2004.

[165] T. M. N. Ngatched, M. Bossert, and A. fahrner, "Hard- and soft-decision decoding algorithms for low-density parity-check codes," in preparation, to be submitted to *IEEE Trans. Commun.*

[166] M. Bossert and F. Hergert, "Hard- and soft-decision decoding beyong the half minimum distance — An algorithm for linear codes," *IEEE Trans. Inform. Theory*, vol. IT-32, no. 5, Sept. 1986.

[167] M. Bossert, *Channel coding for Telecommunications*, Wiley, New York, 1999, pp. 165-167.

[168] R. Lucas, M. Bossert, and M. Breitbach, "On Iterative soft-decision decoding of linear binary block codes and product codes" *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 276-296, Feb. 1998.

[169] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386-398, Jan. 2005.

[170] D.J. C. Mackay, "http://www.inference.phy.cam.ac.uk/mackay/PEG_ECC.html."