Single and Joint Iterative Decoding for Higher Order Modulation Schemes

Juan Carlos Serrato Vital

Submitted in accordance with the requirements

for the degree of Doctor of Philosophy

The University of Leeds

School of Electronic and Electrical Engineering

January 2008

Acknowledgements

The time I have spent as a student in the University of Leeds has had a dramatic impact on my knowledge, my personality and my perception of life. Therefore, I can only be grateful to all the people that have made it possible for me to be here and that have helped me to make the best of this opportunity.

I want to thank my supervisor, Professor O'Farrell for all his advice and dedication. I also want to thank my family for preparing and supporting me for and through this stage, that started since my childhood, especially my mother and grandparents.

Many thanks to all my friends and colleagues both in Mexico and the United Kingdom for their invaluable advice provided, especially to Nimbe Ewald.

Finally, I want to express my gratitude to the Science and Technology National Council (CONACYT) for the funds awarded.

Abstract

The research project described in this thesis concentrates on the study, and application of specific channel coding techniques, in particular, low-density parity-check (LDPC) codes, iterative decoding on Tanner graphs, and their application on joint iterative receivers based on the turbo principle, previously proposed.

The construction of random LDPC codes that fulfil certain desirable characteristics, such as large girth, specific ρ and γ values, and acceptable BER and FER performance for short code lengths, traditionally requires a high degree of processing power (i.e. CPU cycles) to run stochastic routines that firstly search within all the possible combinations for those ones that match the desired characteristics of the LDPC matrix, and secondly determines the bit-error rate (BER) and frame-error rate (FER) performance.

The construction of well structured LDPC codes by means of algebraic methods has provided LDPC codes that achieve excellent performance, with desirable structure on their LDPC matrices. However, from the universe of LDPC matrices, those ones created through well structured procedures are a small group. Multiple procedures to modify their characteristics such as length and rate have assisted to increase the pool of LDPC codes based on

well structured procedures.

This thesis study the problem of constructing random LDPC codes with particular length, girth, and column weight as design parameters, with reduced processing power, while providing, at the same time, a desirable structure to allow efficient use of the memory and of the parallel processing capacity to reduce delay through efficient encoding and decoding.

Based on previous studies that analysed the same problem, an algorithm is introduced to construct the Girth-Partition and Shift (GPS) LDPC codes, which are half-rate quasi-cyclic (QC) LDPC codes. Several GPS constructions are analysed over the AWGN channel and the flat-fading channel. The effect on the BER and FER performance from variations on their design parameters, is included in this study.

This work also includes the BER and FER performance of the concatenation in parallel of different LDPC codes, some of which are based on well structured procedures, such as Euclidean Geometries (EG) and Projective Geomtries (PG), and Margulis constructions based on the Cayley graph, while the rest are based on random procedures, such as Graphical Models (GM) and GPS-LDPC codes. The aim of the analysis of this scheme, combined with the referred LDPC code constructions, include the improvement of the BER and FER performance for short code lengths and the reduction of the encoding complexity.

The BER and FER performance achieved by the parallel concatenation of the previously mentioned LDPC codes, is further analysed in a joint demapping, parallel channel decoding and source decoding system. The impact of each component on the overall system performance is also examined.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BCJR	Bahl, Cocke, Jelinek and Raviv algorithm
BER	Bit Error Rate
CDB	Cycle Database
CDF	Cumulative Distribution Function
CDMA	Code Division Multiple Access
DE	Density evolution
EG	Euclidean Geometry
EXIT	extrinsic information transfer
FER	frame error rate
GA	gaussian approximation
GF	Galois field
GPS	girth-partition and shift
I	circulant identifier
ISI	Inter-symbol interference
LDPC	Low-density parity-check
LLR	Log-likelihood ratio

MAP	Maximum A-Posteriori
MCW	mean column weight
MG	Margulis graph
MUD	Multi-user detection
MUI	Multi-user interference
N	nulls
PDF	probability density function
PG	Projective Geometry
PMF	probability mass function
RSC	Recursive systematic codes
S	slopes
SDB	shift database
SISO	Soft-input soft-output
SOVA	Soft-output Viterbi algorithm
SNR	Signal-to-Noise ratio
TC	Turbo codes
TP	Transition Probability

Chapter 1

Introduction

Channel coding has become an indispensable tool in modern communication and digital storage systems that demand efficient and reliable digital data transmission. The emergence of large scale, high speed data networks that exchange, process and store digital information for commercial, governmental and military applications related to an ever growing number of services and users, requires channel capacity-approaching coding and decoding techniques. The conception of channel coding and transmission schemes dedicated to a particular application is based on a complex set of design factors such as the coding gain, coding rate, bit error rate (BER), delay, implementational complexity and channel characteristics [1].

The evolution of channel coding is full of occasional breakthroughs that gave way to Turbo codes (TC) [2] and Low Density Parity Check (LDPC) codes [3], and the Maximum A-Posteriori (MAP) decoding algorithm [4] that applied to factor graphs [5] is known as sum-product or belief propagation algorithm. Once the potential of such coding and decoding techniques was

recognized, new architectural designs were required to efficiently implement these coding and decoding techniques.

An LDPC matrix understood by means of its factor graph, also known as Tanner graph [6], is useful to derive procedures that modify the origial structure of the parity-check matrix in such a way that improvements on the BER and FER performance are achieved. Factor graphs have made possible the derivation of unified receiver designs that incorporate previously independent sections, as well as channel models, achieving excellent performance [7].

Techniques considered to determine converge thresholds and performance boundaries on LDPC codes and iterative subsystems, have atracted much atention lately, especially Density Evolution [8], Gaussian Approximation [9] and Extrinsic Information [10].

The construction of LDPC codes through structured procedures is still under research. LDPC codes belonging to this family, achieve good BER and FER performance, but the number of codes that fulfil the requirements of length and rate demanded by operating and new communication systems, is still limited.

The theory that allows the design of iterative receivers by means of factor graphs, is presented in [7], but performance characterisation of such joint iterative receivers is still under research. An important contribution to accomplish such task, is presented in [11] and [12]. The work presented in [13], [14], and [15] makes use of all the previously mentioned contributions to derive a joint iterative LDPC receiver.

1.1 Summary of previous research results

In [16], a complete and detailed presentation of structured LDPC codes is presented, including the basic characterisation of the codes for each construction. The performance of LDPC codes whose construction is based on Finite Geometries, Balanced Incomplete Block Designs, Geometry-based Designs, and Turbo-structures, is analysed comparatively. Most of these codes are regular half-rate LDPC codes. It is concluded that codes with large girth converge faster under iterative decoding and improved the performance in the high E_b/N_0 region, by slowing down the onsetting of the error floor.

In [17], three new classes of structure LDPC codes with large girth are introduced. These codes are regular LDPC codes with column weight $\rho = 2$. The codes included under these construction are firstly the rate 1/2 with girth 16, secondly, the rate 1/3 with girth 20, and thirdly, a group of LDPC codes with variable rate and girth 12. These codes perform better than randomly created LDPC codes with similar coding rate.

In [7], factor graphs are used to derive joint iterative receivers for a broad range of channel models and codes. Using the model structure, the joint likelihood function is factorised to later generate the corresponding factor graph, and derive the updates. The channels modeled in this work include the a block fading channel, noncoherent Rayleigh channel with pilots every symbol and every third symbol, and multi-path fading channel.

In [13], [14], and [15], the concatenation in parallel of two randomly created LDPC codes is proposed. BER and FER performance curves are derived through simulations over the AWGN channel and the flat Rayleigh

fading channel, after decoding with the sum-product algorithm. This study includes EXIT chart assuming a Gaussian distribution of the extrinsic information and of the a-priori information, and concludes that EXIT graphs assist on the design of LDPC codes, since the conditions for convergence of the individual (constituent) LDPC codes must met.

1.2 Thesis outline

Based on the comments from the previous section, and a review of existing code construction publications, the work presented is directed towards the following objectives:

1. Develop new algorithms for the construction of LDPC codes, or ensembles of LDPC codes, to overcome the limitations present in state-of-the-art LDPC codes, especially performance, length, and rate.

2. Contribute to the characterisation of the existing types LDPC codes.

3. Derive new applications for existing constructions of LDPC codes and for existing types of LDPC codes.

4. Contribute to the characterization of existing joint iterative subsystems.

5. Analise the performance of state-of-the-art LDPC codes applied to existing joint iterative subsystems.

Chapter 2

In this chapter, complementary background material is provided; this material is needed for the full understanding of our research results. A literature review of some low-density parity-check (LDPC) code constructions, the techniques for code characterisation, and some applications as stand-alone channel decoders, and as joint-iterative receivers.

Chapter 3

This chapter is devoted to the introduction of a new algorithm for the construction of random LDPC codes.

This research complements the work presented in [18], where the construction of LDPC based on Graphic Models (GM) is considered as an option to construct powerful LDPC (n,2,k) LDPC codes. The novelty of this study is based firstly, on the presentation of a new algorithm to create LDPC codes which are quasi-cyclic, in order to increase the universe of LDPC codes constructed by this means, secondly, the reduction on the number of operations required to search for solutions that satisfy the design parameters of the LDPC code, when compared to the Graphic Model procedure, and finally, the characterisation of the LDPC codes constructed through this algorithm.

Chapter 4

This chapter presents the performance of different types of LDPC codes when concatenated in parallel.

This research complements the work presented in [15], where the per-

formance of the parallel concatenation of random LDPC codes, designed to contain specific parameters, is analysed. The novelty of this work relies on the performance analysis of different types of LDPC codes, random and well structured, that fulfil the general requirements established by previous work. This analysis also includes further conditions according to which, the application of this LDPC codes in a parallel concatenated scheme, prove benefitial in terms of BER and FER. Performance improvement for short code lengths can be achieved with the LDPC codes proposed under parallel concatenation. Although efficient encoding and decoding techniques is not part of this research, it is important to mention how the types of LDPC codes analysed in this chapter can benefit from such state-of-the-art encoding and decoding techniques, due to their cyclic and quasi-cyclic structure. One type of LDPC codes included in this Chapter, are the randomly created Girth-Partition and Shift (GPS) LDPC codes, whose construction is introduced in Chapter 3.

Chapter 5

This chapter presents the analysis of a joint iterative receiver consisting of a demapper, a parallel LDPC decoder, and a source decoder. The modifications required to make it feasible the exchange of soft-information for each one of the contituent sections, is already well documented. The novelty of this work relies on the analysis of the constituent subsystems, to understand the individual impact on the overall performance, and possibly determine which design parameters are preferable for this joint scheme.

Chapter 6

This final chapter includes overall conclusions, statements of originality and proposed directions for future work.

1.2.1 Appendix A

This appendix includes the generator polynomial vectors and the circulant vector that define the Euclidean Geomtry (EG) and Projective Geometry (PG) codes, used in various chapters of this dissertation.

1.3 Publications

1. J.C. Serrato, and T. O'Farrell,"Parallel Concatenated Gallager Codes using Euclidean and Projective Geometry LDPC Codes", The Annual London Conference on communications (LCS 2004), University College London, UK, 13-14 Sept. 2004.

2. J.C. Serrato, and T. O'Farrell,"Joint Demapping and Source Decoding for Multilevel Modulation", IEEE Wireless Communications and Networking Conference (WCNC 2006), Las Vegas, Nevada, USA, 3-6 Apr. 2006.

3. J.C. Serrato, and T. O'Farrell, "Structured Parallel Concatenated LDPC codes", The Annual London Conference on communications (LCS 2006), University College London, UK, 14-15 Sept. 2004.

4. J.C. Serrato, and T. O'Farrell, "Girth-Partition and Shift LDPC Codes"

under submission to IET Communications.

Chapter 2

Background and Literature review

2.1 Purpose

This chapter has the following objectives. Firstly, to present a detailed explanation of Turbo Codes (TC), Low-Density Parity-Check (LDPC) codes, Factor Graphs, Iterative Decoding and the Design of Iterative Subsystems. Secondly, to present the state-of-the-art in the previous mentioned topics. Thirdly, to highlight the shortcomings of these coding approaches. Finally the chapter presents a general framework for the studies to be undertaken in this thesis.

2.2 Turbo Codes

Turbo codes have been studied in dept since their discovery in 1973 [2] and have been included for some time in communication systems standards. The turbo principle has also been applied to a variety of receivers, after making modifications to the original encoding and decoding processes. The performance of a joint demapper and source decoder as well as the parallel concatenation of well structured LDPC codes is analised, making use of the turbo principle. Therefore, understanding the characteristics of the turbo codes is essential to this work.

2.2.1 Encoding of Turbo Codes

A turbo code is the parallel concatenation of two codes. The first design considered a subclass of convolutional codes known as Recursive Systematic Convolutional (RSC) codes, where two rate $r = 1/2$ systematic codes were concatenated in parallel, having the input data interleaved before being fed into one of the two encoders. As the encoders are systematic, meaning that one of the outputs is the input itself, and considering that both are fed with the same input, the systematic output of one of the encoders does not need to be transmitted. The overall code rate is $r = 1/3$, although higher code rates can be obtained by puncturing of the parity bits. Fig. 2.1 shows the block diagram of the turbo encoder, where z information bits are encoded to produce the codeword of length $D + P$.

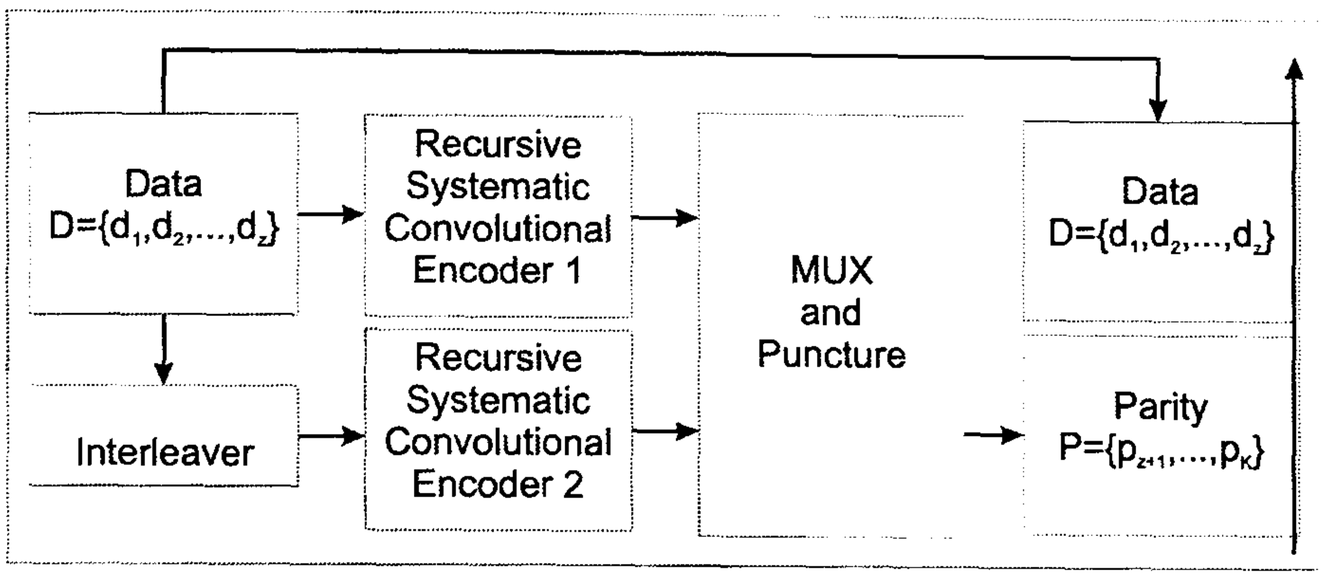

Figure 2.1: Turbo Encoder block diagram

2.2.2 Decoding of Trellis structured codes

Any decoding algorithm is based on either the idea of finding the most likely transmitted information sequence (e.g. the Viterbi algorithm) or on finding the most likely transmitted information bit given the coded sequence (e.g. the Bahl, Cocke, Jelinek and Raviv (BCJR) algorithm [19]). The first type of decoder is called Maximum Likelihood algorithm, while the second is called a Maximum a-posteriori probability (APP) algorithm.

The error performance of both, the Viterbi and the BCJR algorithms, is the same in the case of equally likely information bits, and therefore the first one is preferred due to its lower complexity. However, when the information bits are not equally likely, better performance is achieved with the BCJR algorithm [20].

The BCJR algorithm used in a turbo like structure can deliver superior performance because the a-priori probabilities of the information bits change from iteration to iteration. Also, its inherently Soft-Input Soft-Output nature is very well suited for iterative decoding. On the other hand, the Viterbi algorithm is inherently a Hard-Output algorithm that can be modified to de-

liver a Soft-Output known as the Soft Output Viterbi Algorithm (SOVA). which is more complex and has better performance than the Viterbi algorithm, but is simpler and has a poorer performance when compared to the iterative BCJR decoding algorithm.

Optimal decoding of turbo codes is impractical, as the complexity is too large. The suboptimal iterative decoding algorithm presented in [2], which breaks the overall decoding problem by decoding each of the constituent codes with locally optimal solutions sharing information in an iterative fashion, offers good performance at much lower complexity. Each decoder includes soft-information at the input and produces soft-information at the output which is exchanged and used as a-priori information by the other decoder. During the first iteration no a-priori information is considered. The decoding process continues until some previously setup number of iterations is completed. Iterative decoding obeys a law of diminishing returns such that the incremental gain of each additional iteration is less than that of the previous iteration. The decoding process may not always converge.

A drawback of turbo codes, when compared to LDPC codes, is the difference in the number of decoding iterations spent for each frame, since the turbo decoder has no way to check if the estimated codeword is a valid one, and therefore it is necessary for each received codeword to go through all the iterations previously setup.

It is because of the small number of low weight codewords that turbo codes perform well at low E_b/N_0 values; however, the performance of turbo codes at higher E_b/N_0 values becomes limited by the relatively small minimum distance of the code. The goal of turbo code design is to reduce the

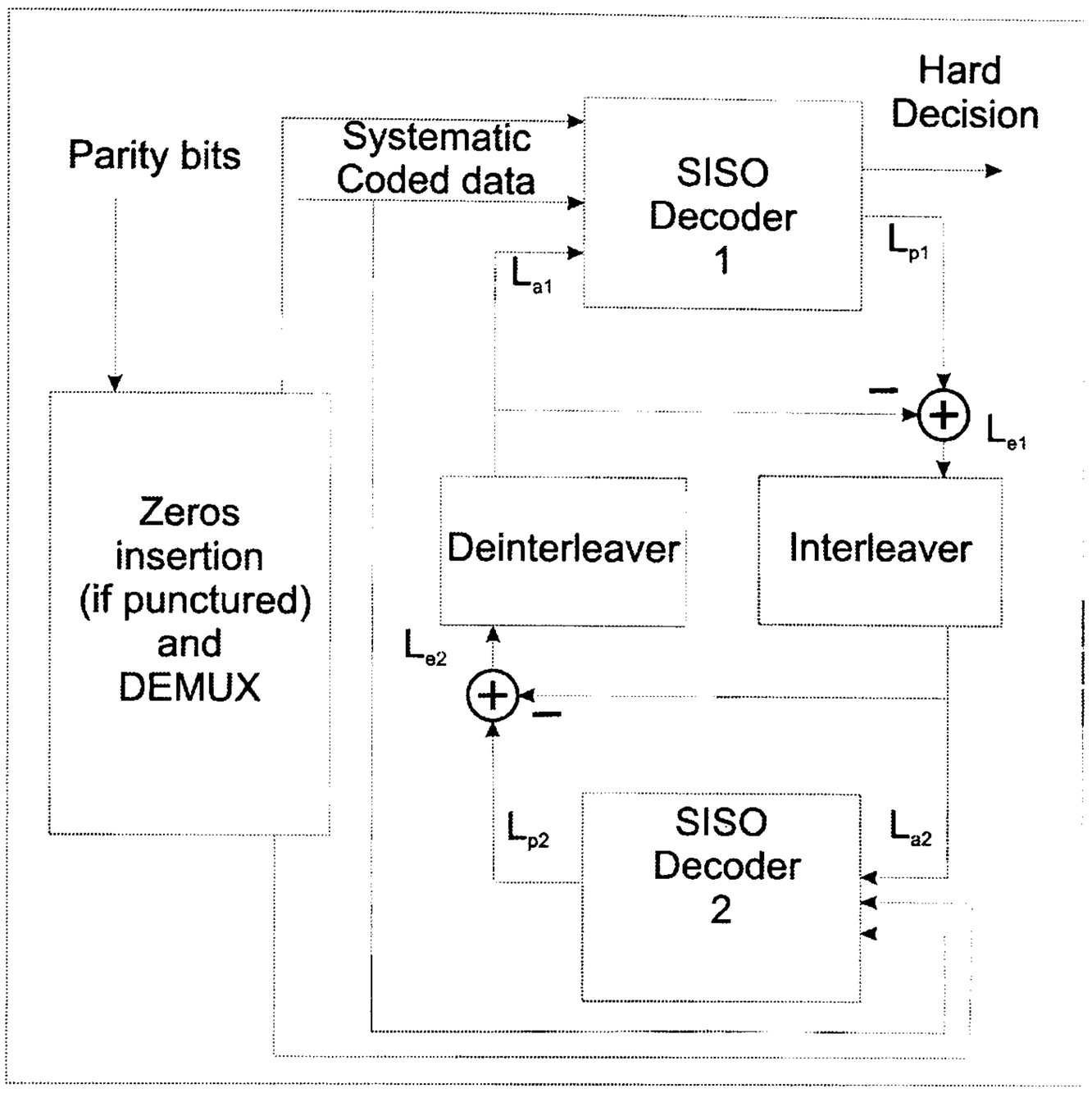

Figure 2.2: Turbo Decoder block diagram

multiplicity of low weight codewords. The BER performance improves as the block size increases at the cost of increased latency. The interleaver design is also a factor only at high E_b/N_0 values, as long as the inputs at the two encoders are sufficiently uncorrelated to avoid a negative effect on the low E_b/N_0 region.

The choice of the constituent Recursive Systematic Convolutional (RSC) encoder and the constraint length do not significantly influence the BER performance. Fig. 2.2 shows the structure of the turbo decoder.

Given the received sequence $\mathbf{r}$, the BCJR algorithm minimizes the BER, by maximizing the a-posteriori probability $P(\hat{u}_l = u_l|\mathbf{r})$ that the information bit u_l is correctly decoded, having the estimate $\hat{u}_l$. The following description

of the algorithm is based on the log-likelihood ratios, or L-values. The decoder inputs the received sequence $\mathbf{r}$ and the a-priori L-values of the information bits $L_a(u_l), l = 0, 1, ..., h-1$. The algorithm calculates the a-posteriori L-values (APP L-values) of each information bit through eq. 2.1,

$$L(u_l) \equiv ln\left[\frac{P(u_l = +1|\mathbf{r})}{P(u_l = -1|\mathbf{r})}\right] \tag{2.1}$$

and the decoder output is given by eq. 2.2

$$\hat{u}_l = \begin{cases} +1 \ \ \forall \ \ L(u_l) > 0 \\ -1 \ \ \forall \ \ L(u_l) < 0 \end{cases} \tag{2.2}$$

In iterative decoding, the APP L-values can be taken as the decoder outputs, resulting in a SISO decoding algorithm. The a-posteriori probability equation is then modified to get eq. 2.3

$$P(u_l = +1|\mathbf{r}) = \frac{p(u_l = +1, \mathbf{r})}{P(r)} = \frac{\sum_{(s',s)\in\sum_l^{u_l=+1}} p(s_l = s', s_{l+1} = s, \mathbf{r})}{P(r)} \tag{2.3}$$

where $\sum_l^+$ is the set of all state pairs $s_l = s'$ and $s_{l+1} = s$ on the trellis diagram, that correspond to the input bit $u_l = +1$ at time $t = l$. Reformulating the expression $P(u_l = -1, \mathbf{r})$ in the same way, the APP L-value can be written as eq. 2.4

$$L(u_l) = ln\left\{\frac{\sum_{(s',s)\in\sum_l^{u_l=+1}} p(s_l = s', s_{l+1} = s, \mathbf{r})}{\sum_{(s',s)\in\sum_l^{u_l=-1}} p(s_l = s', s_{l+1} = s, \mathbf{r})}\right\} \tag{2.4}$$

with $p(s', s, \mathbf{r})$ defined by eq. 2.5,

$$p(s', s, \mathbf{r}) \begin{array}{l} = p(s', s, \mathbf{r}_{t<l}, \mathbf{r}_l, \mathbf{r}_{t>l}) \\ = p(\mathbf{r}_{t>l}|s)\, p(s, \mathbf{r}_l|s')\, p(s', \mathbf{r}_{t<l}) \end{array} \tag{2.5}$$

where after applying the Bayes' rule and considering that the probability of the received branch at time l depends only on the state and input bit at time l. Next, defining α, β, and γ through eq. 2.6,

$$\begin{aligned} \alpha_l(s') &\equiv p(s', \mathbf{r}_{t<l}) \\ \gamma_l(s', s) &\equiv p(s, \mathbf{r}_l|s') \\ \beta_{l+1}(s) &\equiv p(\mathbf{r}_{t>l}|s) \end{aligned} \tag{2.6}$$

the expression for the probability $\alpha_{l+1}(s)$, is rewritten as eq. 2.7

$$\alpha_{l+1}(s) = p(s', \mathbf{r}_{t<l+1}) = \sum_{s' \in \sigma_l} \gamma_l(s', s)\, \alpha_l(s') \tag{2.7}$$

and similarly, the expression for the probability $\beta_{l+1}(s)$, is rewritten as eq. 2.8

$$\beta_l(s') = \sum_{s \in \sigma_{l+1}} \gamma_l(s', s)\, \beta_{l+1}(s) \tag{2.8}$$

where σ_l is the set of all states at time l.

Equation 2.7 represents a forward recursion, while 2.8 represents a backward recursion. The initial values for both recursions respectively are defined in eq. 2.9,

$$\alpha_0(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \quad \beta_K(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \tag{2.9}$$

where K is the length of the input sequence. Finally, the branch metric is defined by 2.10,

$$\gamma_l(s', s) = p(s, \mathbf{r}_l|s') = P(u_l)\, p(\mathbf{r}_l|\mathbf{v}_l) \tag{2.10}$$

where $\mathbf{v}_l$ represents the output bits corresponding to the state transition $s' \rightarrow s$ at time l.

Overall, the conditions to have a successful BER performance with turbo coding are (1) the presence of the two concatenated codes, (2) the independence between the generated codewords through a pseudo-random interleaving process, and (3) the alternate decoding that incorporates a-priori L-values to improve its estimates, and generates a-posteriori L-values to exchange between decoders. These conditions are also present on the LDPC codes.

2.3 Low Density Parity Check Codes

LDPC codes were first discovered by Gallager [3] in the early 1960s. LDPC codes are linear block codes of length n and, just as any other block code, are uniquely specified by either a generator matrix $\mathbf{G}$ of a parity-check matrix $\mathbf{H}$. If it is specified by a parity-check matrix, the code is the null space of $\mathbf{H}$. An n-tuple $\mathbf{v}=\{v_0, v_1, ..., v_{n-1}\}$ over $GF(2)$ is a codeword if and only if $\mathbf{vH}^T=0$.

In particular, an LDPC code has the following structural properties: (1) each row consists of ρ 1's (row weight); (2) each columns consists of γ 1's

(column weight); (3) the number of 1's in common between any two columns, denoted by λ, is no greater than 1 (i.e. for any set of four points with coordinates $P_1(x_1, y_1), P_2(x_1, y_2), P_3(x_2, y_1)$, and $P_4(x_2, y_2)$, at least one of them must have value 0); and (4) both ρ and γ are small compared with the length of the code and the number of rows in $\mathbf{H}$. The density d of the parity-check matrix $\mathbf{H}$ is defined as the ratio between the total number of 1's in $\mathbf{H}$ and the total number of entries in $\mathbf{H}$.

The previous properties were stated in [3] and belong to regular LDPC codes. Irregular LDPC codes follow properties (3) and (4), but have a variable number of 1's per row and/or per column. Irregular LDPC codes have a better BER and FER performance compared to their regular counterparts at the cost of higher encoding complexity when compared to a certain type of regular LDPC codes known as cyclic and quasi-cyclic; however, they may exhibit an error floor at a lower BER, in particular those ones built to achieve a performance very close to their capacity limit. In [21] a procedure to lower the error floor and allow an even better performance of irregular LDPC codes is discussed.

A *path* is defined as a finite alternating sequence of imaginary vertical and horizontal lines that join any two non-zero entries in $\mathbf{H}$. Each non-zero should not be considered more than once. A *cycle* is a *path* with the same beginning and ending non-zero entry. The total number of non-zero entries included in a *cycle* is called the *length*. The *girth* is the *cycle* with the shortest *length* contained in $\mathbf{H}$, which is always a positive even integer. An increase on the *girth* has a positive impact on the performance of the code when decoded with any type of belief-propagation algorithm [22] (message-

passing on a bipartite graph [4]). Gallager's third structural property implies that the LDPC matrix should not contain cycles of $length$=4.

In [23] it was proved that cycle-free codes cannot support good codes, since their cycle-free bipartite graphs provide extremely poor trade off between rate and distance for each fixed length, even though they converge to the optimal solution. An optimal solution refers to the case where the decoding process takes place on a cycle-free LDPC code using any type of belief propagation algorithm; the number of updates on the bipartite graph of the variable and check node L-values is a finite number; it is said that codes whose bipartite graph present cycles cannot come to an optimal solution, since the node updating operation continues until infinity. An upper bound on the minimum distance of cycle-free codes indicates that for codes whose rate is higher than 0.5, the minimum distance is equal to or less than 2. The performance of a code at signal-to-noise ratios of practical interest is not necessarily related to the minimum distance, just as in the case of turbo codes, nevertheless, the same paper concludes that cycle-free codes are defined on parity-check matrices that are much too sparse to allow for a reasonable performance.

A Tanner graph [6], is a graphical representation of an LDPC code where the variables are lined up on one side and the parity checks on the other (bipartite graph), having edges between variables and parity checks that share a non-zero entry on the parity-check matrix $\mathbf{H}$. There are no edges linking variables to variables, nor parity-checks to parity-checks. A stopping set S in a code $\mathbf{C}$ is a subset of the variable nodes in a Tanner graph for $\mathbf{C}$ such that all the neighbours of S are connected to S at least twice. The size

s of the smallest stopping set is termed the stopping distance s or minimum stopping set s_{min} of $\mathbf{C}$ and is defined as a stopping set that does not contain a smaller stopping set within it. An LDPC code may have more than a single s_{min}. [24] proved that iterative decoding cannot determine the variable nodes contained in S when they are incorrect, since even if all the other bits were known, every neighbour of S has at least two connections to the set S and so all messages to S will be erasure messages.

Parameter s plays an important role in understanding the performance of a code under iterative decoding over the binary erasure channel, as LDPC codes with higher s show a better performance [25], thus making s an important factor to consider during the design of LDPC codes. Also, it is clear that adding linearly dependent rows to $\mathbf{H}$ can be advantageous, as this can increase the stopping distance s at the cost of higher complexity and latency [26]. s also explains why some LDPC codes with small girth, maybe even some 4 cycles, have excellent BER and FER performance.

Based on methods of construction, LDPC codes can be classified into two categories: random like codes and structured codes. Random-like LDPC codes [27], [21] are constructed by computer search, based on certain guidelines and required structurel properties of their Tanner graphs, such as girth, degree distributions of variable nodes and check nodes, and stopping sets. Structured LDPC codes are constructed based on algebraic and combinatorial tools, such as Finite Geometries [28], [29], [30], [31], [32], [33], and [34].

The generator matrix of an linear LDPC code can be obtained in the same way as for a linear block code. After manipulating the parity check matrix $\mathbf{H}$, to have it in the form: $\mathbf{H} = [P|I]$ where I represents the identity matrix, the

generator matrix is defined as $\mathbf{G} = [I|P^T]$. Having the information sequence x, the codeword c is obtained by determining $c = x \cdot \mathbf{G} mod(2)$.

Since the rediscovery of LDPC codes decoded with the MAP, Log-MAP and Max-Log-MAP algorithms, different lines of research have been drawn towards finding not just good BER and FER performance LDPC codes, but also towards algorithms that produce structured LDPC codes that are either regular or irregular.

Some of these procedures include Kirkman Triple Systems [35], Balanced Incomplete Block Designs [36], and Lazebnik and Ustimenko (LU) q-regular bipartite graph [37], among many others. Bilayer (expurgated and lengthened) LDPC codes for Decode-and-Forward in relay channels [38], which are capable of working at two different channel parameters and two different rates (relay and destination), are among the state-of-the-art LDPC codes, which are particularly useful for networks. Here, some of them are described in detail.

Although binary LDPC codes can approach the capacity of a variety of channels for asymptotically long block lengths, non-binary $GF(q)$ LDPC codes decoded using belief propagation have better performance on AWGN and Binary Symmetric (BSC) channels and are appealing for use on bursty channels [39]. However, belief propagation decoding for such codes is complex and has received much attention lately [40].

Recently, some specific scenarios that include LDPC codes include [41] and [42]. In the first one, the problem of extracting information from a large sensor network in which sensors cooperatively deliver messages to a mobile access point using a common LDPC codebook is analyzed. In the

second one, it is proved that some LDPC codes can achieve secrecy and can therefore being applied to the wiretap channel. A wiretap channel is modeled as one transmitter t_1 sending k information bits coded into an n bit codeword, through two different channels ch_1 and ch_2, two receivers r_1 and r_2 respectively, where only r_1 should be capable of decoding the received codeword c_1 with negligibly small probability of error, while r_2 should receive the codeword c_2 with mutual information $I(n, c_2)$ approaching zero as the value n approaches infinity. Secrecy capacity of a wire tap channel is the largest k/n for which secure and reliable communication is achievable.

When compared against Turbo codes over the binary-input Additive white Gaussian noise channel, the performance of regular LDPC codes is slightly inferior [8], although irregular LDPC codes outperform Turbo codes.

2.3.1 The construction of LDPC codes based on Gallager's random procedure

The codes presented in [3] are regular and are defined by the following guidelines: Let k be a positive integer greater than 1. Choose any value for ρ and γ. Form a $k\rho$ x $k\gamma$ matrix $\mathbf{H}$ from γ submatrices $\mathbf{H}_1$, $\mathbf{H}_2$,..., $\mathbf{H}_\gamma$ of size k x $k\rho$. For $1 \leq i \leq k$, the ith row of $\mathbf{H}_1$ contains all its ρ 1's in columns $(i\text{-}1)\rho{+}1$ to $i\rho$. The remaining matrices are formed by doing column permutation of $\mathbf{H}_1$. The third structural property limits the number of valid permutations. Once a valid random permutation is obtained, the submatrices that constitute $\mathbf{H}$ are arranged in the form defined by eq. 2.11,

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_\gamma \end{bmatrix} \tag{2.11}$$

where the total number of ones is $k\rho\gamma$ and the total number of entries is $k^2\rho\gamma$ and therefore having a density $\frac{1}{k}$.

2.3.2 The construction of LDPC codes based on Finite Geometries

LDPC codes can be constructed algebraically based on the points and lines of finite geometries [43]. Let $\mathbf{Q}$ be a finite geometry with n points and J lines that has the following fundamental properties: (1) every line consists of ρ points; (2) every point is intersected by γ lines (i.e. every point lies on γ lines; (3) any two points are connected by one and only one line, and (4) two lines are either disjoint (i.e. parallel to each other) or they intersect at one and only one point. The points and lines in $\mathbf{Q}$ are denoted by $P_1, P_2, ..., P_n$ and $L_1, L_2, ..., L_n$, respectively. Let $\mathbf{v}=(v_1, v_2, ..., v_n)$ be an n-tuple over $GF(2)$ whose components correspond to the n points of geometry $\mathbf{Q}$, where the ith component v_i corresponds to the ith point p_i of $\mathbf{Q}$. Let L be a line in $\mathbf{Q}$. A vector is defined based on the points on L as: $\mathbf{v}_L=(v_1, v_2, ..., v_n)$, where v_i is 1 if p_i is a point on L and 0 otherwise.

Let $\mathbf{H_Q}=[h_{i,j}]$ be the $\mathbf{Q}$ geometry LDPC matrix with J rows and n columns (i.e. the rows and columns correspond to the lines and points of

the $\mathbf{Q}$ geometry respectively, where $h_{i,j} = 1$ if and only if the ith line of $\mathbf{Q}$ contains the jth point of $\mathbf{Q}$, and $h_{i,j} = 1$ otherwise.

The Euclidean and the Projective geometries over finite fields are two families of finite geometries that define LDPC matrices. These matrices are square, and for any particular length, the row weight is the same as the column weight. Based on the previous fact, the transpose of any finite geometry LDPC matrix defines in a different code with the same properties as the original one.

The construction of LDPC codes based on Projective Geometry (PG)

Consider the m-dimensional Projective Geometry over $GF(2^s)$, denoted PG$(m, 2^s)$. This geometry consists of $2^s + 1$ points, and each point is represented by an element of $GF(2^{(m+1)s})$, which is considered an extension field of $GF(2^s)$. The number of lines it contains is defined by eq. 2.13.

$$J = \frac{(1 + 2^s + ... + 2^{ms})(1 + 2^s + ... + 2^{(m-1)s})}{1 + 2^s} \tag{2.12}$$

lines, and each line consists of $2^s + 1$ points. Each point is intersected by

$$\gamma = \frac{2^{ms} - 1}{2^s - 1} \tag{2.13}$$

As an example, let m=2 and s=2. Let $GF(64)$ be the extension field of $GF(4) = \{0, 1, \beta, \beta^2\}$. Each element in $GF(64)$ is represented as a 3-tuple over $GF(4)$ as shown in Table 2.1.

The (0,0,0) 3-tuple is not included. To produce all the 3-tuples in the

num	x	y	z	num	x	y	z	num	x	y	z
0	1	0	0	21	w^2	0	0	42	w	0	0
1	0	1	0	22	0	w^2	0	43	0	w	0
2	0	0	1	23	0	0	w^2	44	0	0	w
3	w^2	1	w	24	w	w^2	1	45	1	w	w^2
4	1	1	w	25	w^2	w^2	1	46	w	w	w^2
5	1	w^2	w	26	w^2	w	1	47	w	1	w^2
6	1	w^2	0	27	w^2	w	0	48	w	1	0
7	0	1	w^2	28	0	w^2	w	49	0	w	1
8	w	w^2	0	29	1	w	0	50	w^2	1	0
9	0	w	w^2	30	0	1	w	51	0	w^2	1
10	w	w^2	w^2	31	1	w	w	52	w^2	1	1
11	w	1	w	32	1	w^2	1	53	w^2	w	w^2
12	1	0	w	33	w^2	0	1	54	w	0	w^2
13	1	w^2	w^2	34	w^2	w	w	55	w	1	1
14	w	w	w	35	1	1	1	56	w^2	w^2	w^2
15	1	0	1	36	w^2	0	w^2	57	w	0	w
16	w^2	0	w	37	w	0	1	58	1	0	w^2
17	1	1	w^2	38	w^2	w^2	w	59	w	w	1
18	w	w	0	39	1	1	0	60	w^2	w^2	0
19	0	w	w	40	0	1	1	61	0	w^2	w^2
20	1	w	1	41	w^2	1	w^2	62	w	w^2	w

Table 2.1: The 63 points in the PG$(2, 2^2)$, defined as 3-tuples (x,y,z) over $GF(2^2)$ excluding the 3-tuple (0-0-0)

appropriate order, any 3-tuple is multiplied by the matrix $\mathbf{O}$ portrayed in eq. 2.14, consisting on a zero vector of length m, an identity matrix of size m by m, and a 3-tuple as bottom row with elements in $GF(64)$.

$$\mathbf{O} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ w^2 & 1 & w \end{bmatrix} \tag{2.14}$$

Next, the creation of Table 2.1 is demonstrated in detail. Assign number num(0) to any 3-tuple representing a point in PG$(2, 2^2)$, which in this case

is the 3-tuple $(1,0,0)$. According to the previous procedure, the number num(1) will be assigned to the 3-tuple obtained from eq. 2.15

$$\text{num}(1) = (1,0,0) \cdot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ w^2 & 1 & w \end{bmatrix} \tag{2.15}$$

The inner product is obtained by using table 2.2 and ??, which is shown in eq. 2.16, which correspond to the second value in table 2.1, and represents the point in the second position in this field. Using point num(2), the point in the third position can be obtained, which is (0,0,1). Once all the 63 3-tuples representing the points within this field have been assigned a position, using the last point num(62) to get a further number will result in the point located in the first position num(0), which is the one arbitrarily chosen to start ordering the 3-tuples in the field. From this last statement, it is possible to decide which vector with elements in the field can be chose to represent the last row in matrix 2.14, since only a valid vector can be useful to assign positions to all the 3-tuples, while an invalid vector would only assign positions to a limited number of 3-tuples.

$$\begin{aligned} &= (1\cdot 0 + 0\cdot 0 + 0\cdot w^2, 1\cdot 1 + 0\cdot 0 + 0\cdot 1, 1\cdot 0 + 0\cdot 1 + 0\cdot w) \\ \text{num}(1) &= (0+0+0, 1+0+0, 0+0+0) \\ &= (0,1,0) \end{aligned} \tag{2.16}$$

Addition and multiplication are defined over $GF(2^2)$, using the primitive polynomial $w^2 = w + 1$, and are depicted in Table 2.2 and Table 2.3

respectively.

element 1	element 2	Addition
0	0	0
0	1	1
0	w	w
0	w^2	w^2
1	0	1
1	1	0
1	w	w^2
1	w^2	w
w	0	w
w	1	w^2
w	w	0
w	w^2	1
w^2	0	w^2
w^2	1	w
w^2	w	1
w^2	w^2	0

Table 2.2: Addition over $GF(2^2)$

Let $\alpha^i \ \forall(1 \leq i \leq 63)$ be a point in $PG(2,4)$, then, the line $(\eta_1\alpha^i + \eta_2\alpha^j)$ and the line $\left(\eta_1\alpha^k + \eta_2\alpha^j\right)$ with η over $GF(2^2)$ have α^j as a common point (the only common point).

Generalizing, the line passing through (or connecting) α_i and α_j consists of points in the form $(\eta_1\alpha^i + \eta_2\alpha^j)$, where both η cannot be equal to zero. In this case η_1 and η_2 take independently the values $0, 1, w, w^2$ to produce 15 lines, also represented as 3-tuples in Table 2.4 where only 5 points are independent (i.e. different points), since the elements $(\alpha^i, \beta\alpha^i, \beta^2\alpha^i)$ with β over $GF(2^2)$ are considered to be the same point in $PG(2,4)$. The last two columns at the right in Table 2.4 indicate the number of the point according to Table 2.1 and one among the three independent vectors respectively. The

element 1	element 2	Multiplication
0	0	0
0	1	0
0	w	0
0	w^2	0
1	0	0
1	1	1
1	w	w
1	w^2	w^2
w	0	0
w	1	w
w	w	w^2
w	w^2	1
w^2	0	0
w^2	1	w^2
w^2	w	1
w^2	w^2	w

Table 2.3: Multiplication over $GF(2^2)$

chosen vector from among the three, should be one with number equal of less than J with $J = 21$ in this particular case.

Here follows the creation the first row of the LDPC matrix $\mathbf{H_{PG(2,4)}}$, by obtaining the line that joins the five points, corresponding to the five values in such first row, whos value is 1 (i.e. the other values in the first row take value 0). To obtain all the points in table 2.4, values are assigned to $(\eta_1\alpha^i + \eta_2\alpha^j)$. α^i and α^j can take independently any of the 3-tuples with elements $0, 1, w, w^2$, where the 3-tuples chosen are $\alpha^i = (1, 0, 0)$ and $\alpha^j = (0, 1, 0)$. η_1 and η_2 can take independently any of the values $0, 1, w, w^2$ except both 0. The first point is obtained by making $\eta_1 = 0$ and $\eta_2 = 1$. The value of the first point is shown in eq. 2.17.

$$\begin{aligned}
\text{1st row} &= 0(1,0,0) + 1(0,1,0) = (0,0,0) + (0,1,0) = (0,1,0) \\
\text{2nd row} &= 0(1,0,0) + w(0,1,0) = (0,0,0) + (0,w,0) = (0,1,0) \\
\text{3rd row} &= 0(1,0,0) + w^2(0,1,0) = (0,0,0) + (0,w^2,0) = (0,w^2,0) \\
\text{4th row} &= 1(1,0,0) + 0(0,1,0) = (1,0,0) + (0,0,0) = (1,0,0) \\
\vdots &\quad \vdots \\
\text{15th row} &= w^2(1,0,0) + w^2(0,1,0) = (w^2,0,0) + (0,w^2,0) = (w^2,w^2,0)
\end{aligned} \tag{2.17}$$

By relating the obtained vectors in 2.17, with the order assigned to such vectors in table 2.1, table 2.4 is determined. It is required to know which vectors are independent; in order to do so, the first vector in table 2.4 is considered independent, and is multiplied by all the elements defined over $GF(2^2)$ (i.e. the vector (0,1,0) is multiplied by the elements $0, 1, w, w^2$) to obtain the dependent vectors (i.e. the vectors $(0,0,0), (0,w,0)$, and $(0,w^2,0)$).

From Table 2.4, the 3-tuples numbered 0, 1, 6, 8 and 18 represent the positions in the vector of length 21 whose value is set to one, while the rest of the positions take zero as value (i.e. from left to right, the first, second, fifth, seventh and seventeenth elements out of 21 elements take value 1, while the rest take value 0), as shown in eq. 2.18.

$$\text{first row} \quad 110000101000000000100 \tag{2.18}$$

The LDPC matrix $\mathbf{H_{PG(2,4)}}$ is formed by cyclic shift of vector 2.18, as depicted in expression 2.5.

x	y	z	number	dep/independent
0	1	0	1	independent
0	w	0	43	dep
0	w2	0	22	dep
1	0	0	0	independent
1	1	0	39	dep
1	w	0	29	dep
1	w2	0	6	independent
w	0	0	42	dep
w	1	0	48	dep
w	w	0	18	independent
w	w2	0	8	independent
w2	0	0	21	dep
w2	1	0	50	dep
w2	w	0	27	dep
w2	w2	0	60	dep

Table 2.4: Table with the 15 $(\eta_1\alpha^i + \eta_2\alpha^j)$ combinations using $GF(2^2)$, indicating which 3-tuples are independent

The construction of LDPC codes based on Euclidean Geometry (EG)

Consider the m-dimensional Euclidean geometry over $GF(2^s)$, EG$(m, 2^s)$. This geometry consists of 2^{ms} points, and each point is represented by an m-tuple over $GF(2^s)$. The point represented by the all-zero m-tuple, $\mathbf{0}$=(0,0,...,0), is called the *origin*. The number of lines it contains is defined by eq. 2.19, where each line consists of 2^s points. Also, each point is intersected by the number of lines defined by eq. 2.20.

$$J = \frac{2^{(m-1)2}(2^{ms}-1)}{2^s-1} \tag{2.19}$$

$$\left|\begin{array}{cccccccccccccccccccc}
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{array}\right|$$

Table 2.5: LDPC based on PG(2,4)

$$\gamma = \frac{2^{ms} - 1}{2^{s} - 1} \tag{2.20}$$

Let $m = 2$ and $s = 2$. Then $\mathrm{EG}(2, 2^2)$ is a 2-dimensional Euclidean geometry over the Galois field $GF(4)$, with 16 points, each one being a 2-tuple over $GF(4)$ as shown in Table 2.6.

From Table 2.7, the 2-tuples numbered 0, 4, 12, and 13 represent the positions in the vector of length 15 whose value is set to one; the rest of the positions take zero as value. The LDPC matrix $\mathbf{H}_{\mathbf{EG(2,4)}}$ is formed by cyclic shift of the such vector, as depicted in expression 2.8.

The following list contains the vectors representing the circulants $\mathbf{E}$ that

0	1	0
1	0	1
2	w1	1
3	w1	w2
4	1	1
5	w1	0
6	0	w1
7	w2	w1
8	w2	1
9	w1	w1
10	w2	0
11	0	w2
12	1	w2
13	1	w1
14	w2	w2
15	1	0

Table 2.6: The 16 points in the EG$(2, 2^2)$, defined as 2-tuples (x,y) over $GF(2^2)$ including the origin (0-0)

generate the LDPC matrices $\mathbf{H}$, as well as the corresponding generator polynomials ($\mathbf{GP}$) of $\mathbf{H}$ and $\mathbf{H^T}$, for the EG and PG LDPC codes, analysed under parallel contenation in future Chapters.

- EG(2,2^2), $n = 15$, $k = 7$, $\rho = 4$, $\gamma = 4$
- $\mathbf{E}$=$x^{14} + x^{10} + x^2 + x$
- $\mathbf{GP_H}$=$x^8 + x^4 + x^2 + x + 1$
- $\mathbf{GP_{H^T}}$=$x^8 + x^7 + x^6 + x^4 + 1$
- EG(2,2^3), $n = 63$, $k = 37$, $\rho = 8$, $\gamma = 8$
- $\mathbf{E}$=$x^{62} + x^{54} + x^{49} + x^{38} + x^{28} + x^6 + x^5 + x^3$
- $\mathbf{GP_H}$=$x^{26} + x^{22} + x^{16} + x^{12} + x^6 + x^5 + x^2 + x + 1$

x	y	number	dep/independent
0	1	1	dep
0	w1	6	dep
0	w2	11	dep
1	0	0	independent
1	1	4	independent
1	w1	13	independent
1	w2	12	independent
w1	0	5	dep
w1	1	2	dep
w1	w1	9	dep
w1	w2	3	dep
w2	0	10	dep
w2	1	8	dep
w2	w1	7	dep
w2	w2	14	dep

Table 2.7: Table with the 15 $(\alpha^i + \eta_2\alpha^j)$ combinations using $GF(2^2)$, indicating which 2-tuples are independent

- $\mathbf{GP_{H^T}}$=$x^{26} + x^{25} + x^{24} + x^{21} + x^{20} + x^{14} + x^{10} + x^4 + 1$

- EG(2,2^4), $n = 255$, $k = 175$, $\rho = 16$, $\gamma = 16$

- $\mathbf{E}$=$x^{254} + x^{251} + x^{225} + x^{221} + x^{215} + x^{163} + x^{155} + x^{144} + x^{143} + x^{88}$ $+x^{60} + x^{46} + x^{39} + x^{23} + x^{14} + x$

- $\mathbf{GP_H}$=$x^{80} + x^{76} + x^{73} + x^{72} + x^{70} + x^{69} + x^{68} + x^{66} + x^{65} + x^{64} + x^{63} +$ $x^{62} + x^{61} + x^{60} + x^{58} + x^{57} + x^{56} + x^{54} + x^{53} + x^{52} + x^{43} + x^{42} + x^{41} +$ $x^{40} + x^{39} + x^{37} + x^{34} + x^{32} + x^{30} + x^{28} + x^{25} + x^{24} + x^{21} + x^{20} + x^{18} +$ $x^{16} + x^{15} + x^{13} + x^{12} + x^{11} + x^9 + x^6 + x^4 + x^2 + 1$

- $\mathbf{GP_{H^T}}$=$x^{80} + x^{78} + x^{76} + x^{74} + x^{71} + x^{69} + x^{68} + x^{67} + x^{65} + x^{64} +$ $x^{62} + x^{60} + x^{59} + x^{56} + x^{55} + x^{52} + x^{50} + x^{48} + x^{46} + x^{43} + x^{41} + x^{40} +$ $x^{39} + x^{38} + x^{37} + x^{28} + x^{27} + x^{26} + x^{24} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} +$

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

Table 2.8: LDPC based on EG(2,4)

$$x^{17} + x^{16} + x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^4 + 1$$

- EG(2,2^5), $n = 1023$, $k = 781$, $\rho = 32$, $\gamma = 32$

- $\mathbf{E}$=$x^{1022} + x^{934} + x^{932} + x^{885} + x^{874} + x^{809} + x^{768} + x^{761} + x^{740} + x^{656} + x^{618} + x^{594} + x^{418} + x^{414} + x^{409} + x^{401} + x^{391} + x^{375} + x^{355} + x^{323} + x^{320} + x^{298} + x^{283} + x^{219} + x^{213} + x^{169} + x^{168} + x^{139} + x^{86} + x^{72} + x^{30} + x^{11}$

- PG(2,2^2), $n = 21$, $k = 11$, $\rho = 5$, $\gamma = 5$

- $\mathbf{E}$=$x^{20} + x^{19} + x^{14} + x^{12} + x^2$

- $\mathbf{GP_H}$=$x^{10} + x^8 + x^6 + x^4 + x^3 + 1$

- $\mathbf{GP_{H^T}}$=$x^{10} + x^7 + x^6 + x^4 + x^2 + 1$

- PG(2,2^3), $n = 73$, $k = 45$, $\rho = 9$, $\gamma = 9$ $\mathbf{E}$=$x^{72} + x^{71} + x^{69} + x^{65} + x^{57} + x^{41} + x^{36} + x^{18} + x^9$

- $\mathbf{GP_H}$=$x^{28} + x^{26} + x^{22} + x^{19} + x^{18} + x^{17} + x^{14} + x^{13} + x^{12} + x^{9} + x^{8} + x^{5} + x^{4} + x^{3} + x + 1$

- $\mathbf{GP_{H^T}}$=$x^{28} + x^{27} + x^{25} + x^{24} + x^{23} + x^{20} + x^{19} + x^{16} + x^{15} + x^{14} + x^{11} + x^{10} + x^{9} + x^{6} + x^{2} + 1$

- PG(2,2^4), $n = 273$, $k = 191$, $\rho = 17$, $\gamma = 17$

- $\mathbf{E}$=$x^{272} + x^{271} + x^{264} + x^{261} + x^{246} + x^{213} + x^{208} + x^{184} + x^{116} + x^{114} + x^{100} + x^{94} + x^{77} + x^{73} + x^{64} + x^{45} + x^{33}$

- $\mathbf{GP_H}$=$x^{82} + x^{76} + x^{68} + x^{58} + x^{53} + x^{52} + x^{47} + x^{44} + x^{39} + x^{34} + x^{28} + x^{24} + x^{18} + x^{14} + x^{6} + x^{4} + x^{2} + 1$

- $\mathbf{GP_{H^T}}$=$x^{82} + x^{80} + x^{78} + x^{76} + x^{68} + x^{64} + x^{58} + x^{54} + x^{48} + x^{43} + x^{38} + x^{35} + x^{30} + x^{29} + x^{24} + x^{14} + x^{6} + 1$

- PG(2,2^5), $n = 1057$, $k = 813$, $\rho = 33$, $\gamma = 33$

- $\mathbf{E}$=$x^{1056} + x^{1055} + x^{1012} + x^{1000} + x^{974} + x^{942} + x^{932} + x^{867} + x^{859} + x^{767} + x^{673} + x^{669} + x^{623} + x^{576} + x^{574} + x^{551} + x^{515} + x^{497} + x^{488} + x^{464} + x^{449} + x^{419} + x^{412} + x^{359} + x^{348} + x^{345} + x^{328} + x^{239} + x^{217} + x^{211} + x^{198} + x^{182} + x^{177}$

2.3.3 The construction of LDPC codes based on Cayley Graphs

In [44] an algebraic construction of LDPC codes is proposed based on k-regular graphs by means of Cayley graphs. For this, consider a finite group G. Let A be a subset of G (i.e. $A \in G$ satisfying $A = A^{-1}$. The Cayley

graph $X(G, A)$ is the graph having as vertices the elements $g \in A$ such that $h = ga$. $X(G, A)$ is an undirected k-regular graph with $k = |A|$.

Let q be an odd prime and let F_q be the finite field of q elements and consider the *SpecialLinearGroup* $SL_2(F_q)$ consisting of all 2 x 2 matrices with entries in F_q and having determinant $ad - bc$=1. $SL_2(F_q)$ is a group of order $q^3 - q$.

Let $G = SL_2(F_q)$ and let

$$A := \left\{ A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}, A^{-1} = \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}, B^{-1} = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} \right\} \tag{2.21}$$

Then the Cayley graph $X(G, A)$ is a 4-regular graph and (3,6) regular LDPC codes can be constructed with length $2(q^3 - q)$. Two copies of G (G_u, G_v) represent the left vertices in the bipartite graph, while one copy of G represents the right vertices. An element $g_u \in G_u$ on the left will be connected to the right vertices $g_u A^2, g_u ABA^{-1}, g_u B$. An element $g_v \in G_v$ on the left will be connected with the right vertices $g_v A^{-2}, g_v AB^{-1}A^{-1}, g_v B^{-1}$

2.3.4 Construction of LDPC codes based on Graphical Models

In [17], [45] and [18], the construction of LDPC codes based on graphical models is introduced. This construction is outstanding as the constraints are relatively loose, allowing a wide range of code lengths and rates. Such codes have $girth > 6$ and their column weight can be either 2 or 3.

In [17] and [43], the procedure to construct $girth = 16(n, 2, n/2)$ LDPC codes is introduced. The main idea is to represent the LDPC matrix by eight sections. Each section is either the identity matrix or a shifted version. The number of positions that the elements are shifted, is called the slope-pair, which has a graphical representation. The procedure looks for sixteen slope-pairs that fulfill some constraints; e.g. two slope-pairs per section. Fig. 2.3 shows the $unfolded$ eight sections of the cylinder-like graphical model, for the (368,2,184) LDPC code with $g = 16$. When the LDPC matrix is represented by ten sections, with two slope-pairs for even sections and one slope-pair for odd sections, $girth = 20(n, 2, n/3)$ LDPC codes can be developed. Also, by modifying the graphical representation and looking for slope-pairs, $girth = 12(n, 2, k)$ LDPC codes with $1/2 < k < 1$ can be constructed.

2.4 Factor Graphs

A factor graph is a diagram that represents the factorization of a function of several variables (e.g. the function $f(v, w, x, y, z)$ can be factorized as $f(v, w, x, y, z) = f1(v|w, x)f2(w)f3(x|y, z)$ as shown in Fig. 2.4). Such a graph consists of nodes (or squares) and edges (or circles). There is a unique node for every factor and there is a unique edge for every variable. The factors are also called local functions and their product is called the global function [5]. Therefore, $f(v, w, x, y, z)$ represents a global function and $f1, ..., f3$ represent local functions.

A main application of factor graphs are probabilistic models, where for a group of random variables (local functions), their joint probability density

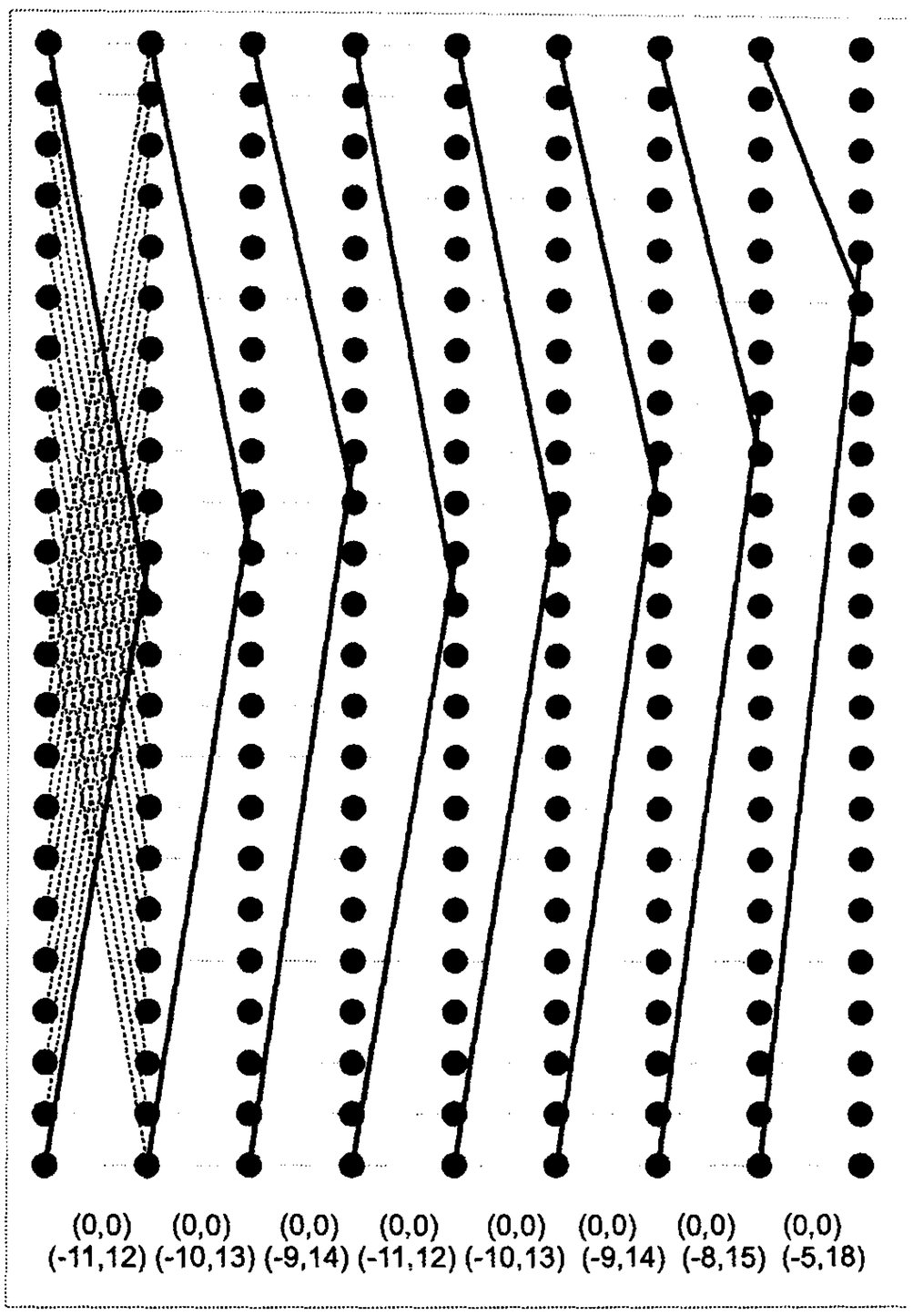

Figure 2.3: Cylinder-like structure for the GM(368,2,184) LDPC code with g=16

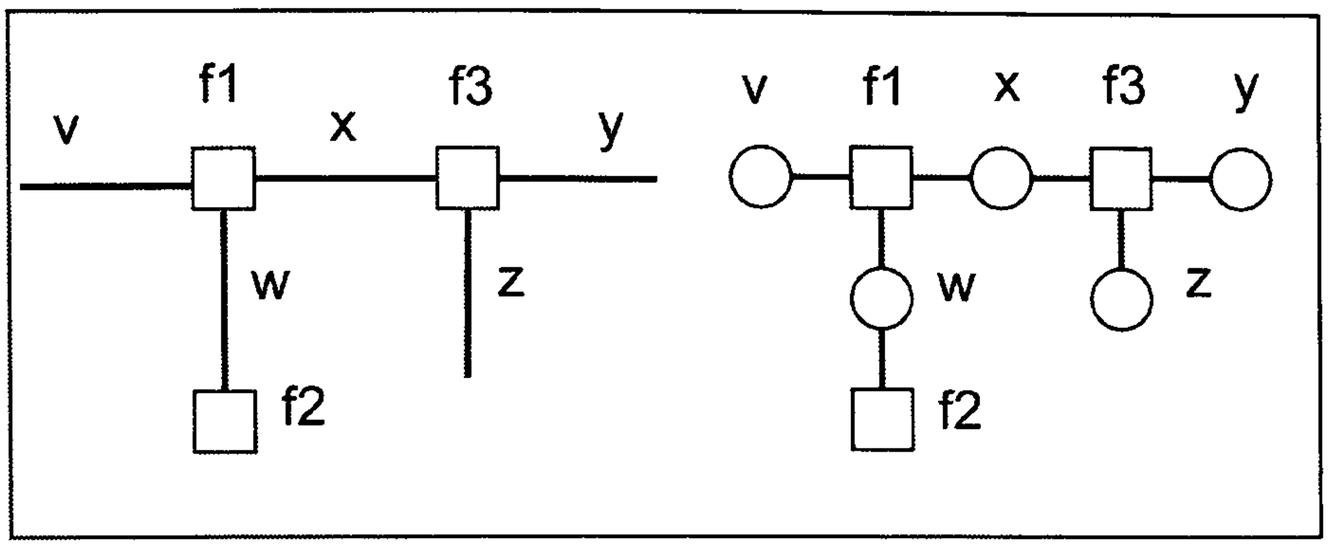

Figure 2.4: Factor graph of the factorization of the function $f(v, w, x, y, z) = f1(v|w, x)f2(w)f3(x|y, z)$

function would represent the global function. The example of probability propagation, which had been used only for cycle-free graphs, proved later to be useful also for graphs with cycles.

In coding theory, different decoding algorithms make use of graphical models to be understood, such as the Viterbi algorithm that operates on a trellis diagram to decode convolutional codes. The generalization of the concept behind traditional decoding algorithms results in two main summary propagation algorithms, the sum-product algorithm and the max-product (or min-sum) algorithm. The structure of these algorithms matches the graph directly as they operate by passing messages (or summaries) along the edges of the graph.

Making use of linear algebra, a binary linear block code can be represented as in eq. 2.22,

$$C = \left\{ x \in F^n : \mathbf{H}x^T = 0 \right\} \tag{2.22}$$

where $\mathbf{H}$ is the parity-check matrix over F and where n in the length of the code. Consider for example the EG(2,2) LDPC matrix depicted in eq. 2.23

that defines the $(15, 4, 7)$ LDPC code.

$$\mathbf{H_{EG(2,2)}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1
\end{bmatrix} \tag{2.23}$$

It follows from 2.22 that the membership indicator function 2.24 (i.e. the function that dictates which codewords of length n belong to the null space of $\mathbf{H_{EG(2,2)}}$) of this code can also be written as eq. 2.25, where $\oplus$ denotes addition modulo 2. Note that each factor in eq. 2.25 corresponds to one row of the parity-check matrix of eq. 2.8.

$$I_C : F^n = \{0,1\} : x \mapsto \begin{cases} 1, if x \in C \\ 0, else \end{cases} \tag{2.24}$$

$$I_C\left(x_1,..,x_{15}\right) = \begin{array}{c} (x_1 \oplus x_5 \oplus x_{13} \oplus x_{14}) \\ \cdot(x_2 \oplus x_6 \oplus x_{14} \oplus x_{15}) \\ \cdot(x_1 \oplus x_3 \oplus x_7 \oplus x_{15}) \\ \cdot(x_1 \oplus x_2 \oplus x_4 \oplus x_8) \\ \vdots \\ \cdot(x_4 \oplus x_{12} \oplus x_{13} \oplus x_{15}) \end{array} \tag{2.25}$$

The left column (i.e. the first variable and the corresponding parity checks) and the bottom row (i.e. the last parity check and the corresponding variables) of the LDPC matrix $\mathbf{H_{EG(2,2)}}$ in eq. 2.8 are depicted in Fig. 2.5. This is assuming the variable nodes in the LDPC matrix $\mathbf{H_{EG(2,2)}}$ are numbered from left to right, and the parity-check nodes are numbered from top to bottom.

Iterative Decoding of LDPC codes

For the decoding of LDPC codes, the two most popular belief propagation decoding algorithms are the min-sum and the sum-product algorithms. The first one is a generalization of the Viterbi algorithm and looks for the most likely transmitted codeword, while the second one looks for the most likely transmitted bit, for each bit in a codeword.

Let $f(x_1, x_2, x_3, ..., x_{15})$ be some discrete probability mass function, then the marginal probability for $p(x_2)$ can be written as in eq. 2.26.

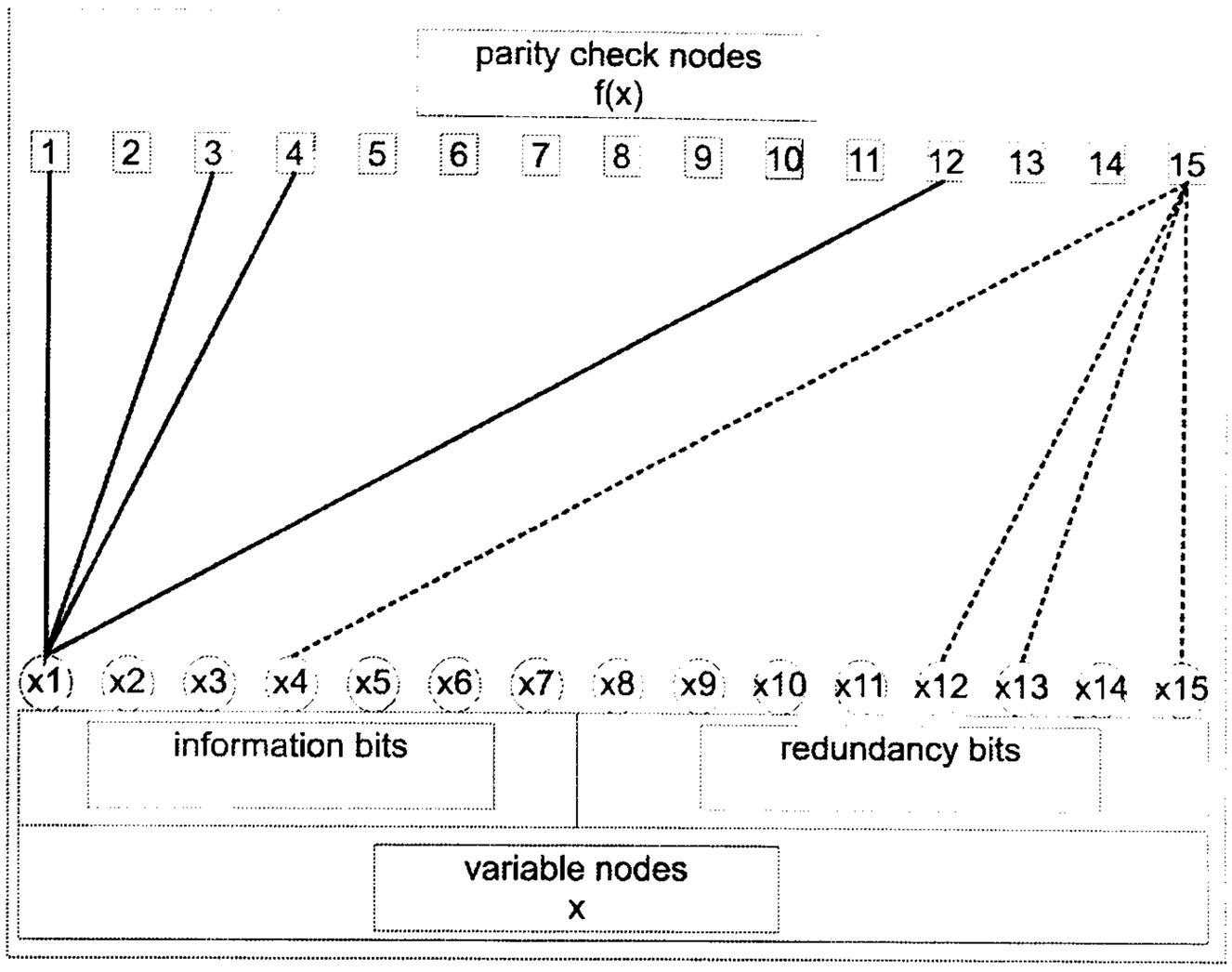

Figure 2.5: Factor graph of the $EG(2, 2^2)$ LDPC code

$$p(x_2) = \sum_{x_1, x_3, \ldots, x_{15}} f(x_1, x_2, x_3, \ldots, x_{15}) \tag{2.26}$$

If the factor graph in Fig. 2.5 includes all the variable nodes but without cycles, and considers x_2 as the root of the factor graph, it can be depicted as in Fig. 2.6. Let the function $f(x_1, x_2, x_3, \ldots, x_{15})$ be the discrete probability mass function of the factor graph in Fig. 2.6; then f can be written as eq. 2.27.

$$\begin{aligned} f(x_1, \ldots, x_{15}) = & \cdot f_2(x_2, x_{14}) \\ & \cdot f_5(x_2, x_3, x_5, x_9) \\ & \cdot (f_4(x_1, x_2, x_4, x_8) f_6(x_6) f_7(x_7) f_{15}(x_{12}, x_{15})) \end{aligned} \tag{2.27}$$

$$\cdot f_{13}(x_2, x_{10}, x_{11}, x_{13})$$

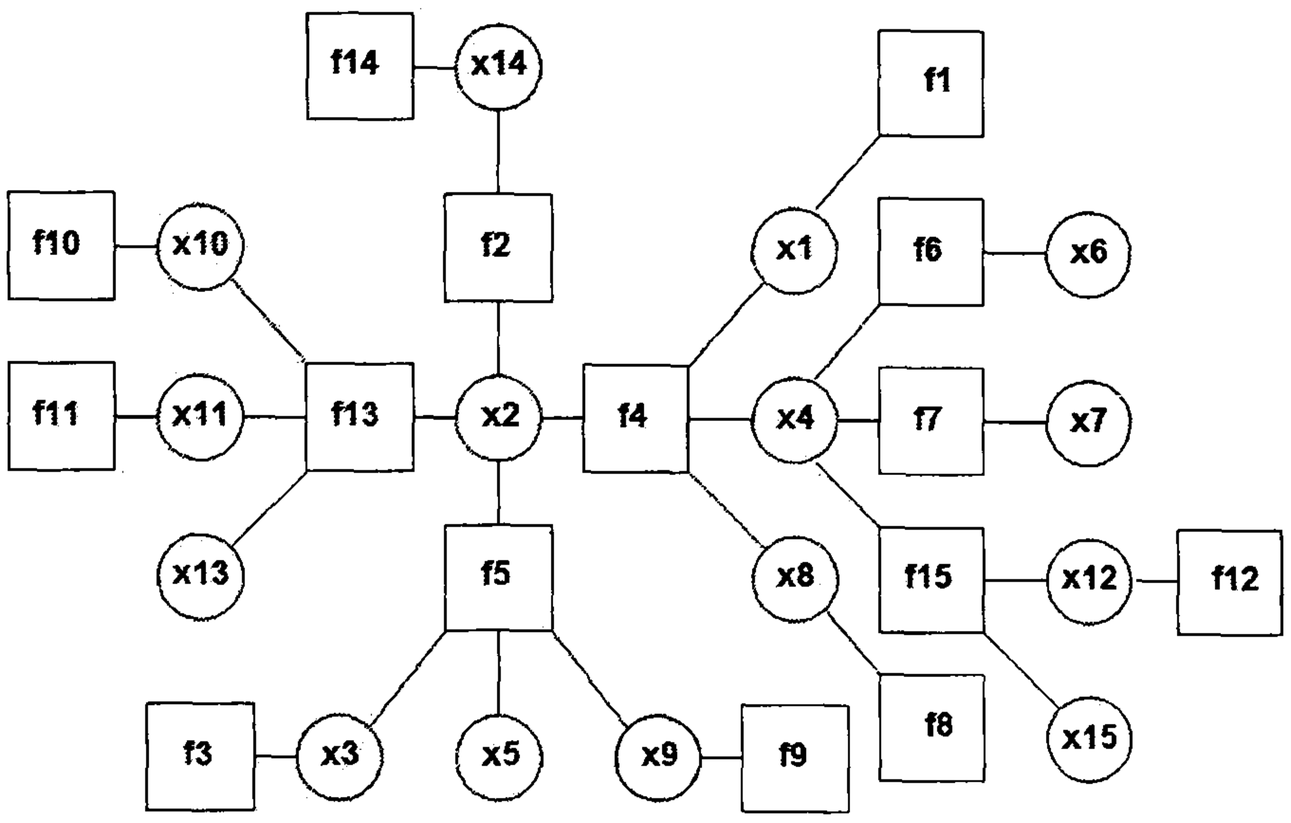

Figure 2.6: Factor graph of the $\mathbf{H}_{\mathbf{EG}(\mathbf{2},\mathbf{2^2})}$ LDPC matrix in eq. 2.8, including only once all the variable nodes (i.e. without cycles), and using x_2 as the root variable node

Inserting eq. 2.27 into eq. 2.26 and applying the distributive law, yields eq. 2.28.

$$
\begin{aligned}
p(x_2) = & \underbrace{\sum_{x_{10}} \sum_{x_{11}} \sum_{x_{13}} f_{13}(x_2, x_{10}, x_{11}, x_{13}) f_{11}(x_{11}) f_{10}(x_{10})}_{\mu f_{13} \to x_2} \\
& \underbrace{\cdot \sum_{x_{14}} f_2(x_2, x_{14}) f_{14}(x_{14})}_{\mu f_2 \to x_2} \\
& \underbrace{\cdot \sum_{x_9} \sum_{x_3} \sum_{x_5} f_5(x_2, x_3, x_5, x_9) f_3(x_3) f_9(x_9)}_{\mu f_5 \to x_2} \\
& \cdot \left(\sum_{x_1} \sum_{x_8} \sum_{x_4} f_4(x_1, x_2, x_4, x_8) f_1(x_1) f_8(x_8) \right. \\
& \left. \cdot \left(\sum_{x_6} f_6(x_4, x_6) \sum_{x_7} f_7(x_4, x_7) \sum_{x_{12}} \sum_{x_{15}} f_{15}(x_4, x_{12}, x_{15}) f_{12}(x_{12}) \right) \right)
\end{aligned}
\tag{2.28}
$$

The factor $\mu f_{13} \to x_2$ is the summary of the functions (i.e. the parity-check nodes) including all the variable nodes in the left part of Fig. 2.6.

Therefore, eq. 2.28 is transformed into eq. 2.29,

$$p(x_2) = \mu f_{13} \rightarrow x_2 \cdot \mu f_2 \rightarrow x_2 \cdot \mu f_5 \rightarrow x_2 \cdot \mu f_4 \rightarrow x_2 \tag{2.29}$$

which corresponds to the summary factors (messages) of each one of the four branches in Fig. 2.6.

The Sum-Product rule is the function (eq. 2.30) that represents the message out of some node $f(x, y_1, ..., y_n)$ along the branch x.

$$\mu_{g \rightarrow x}(x) = \sum_{y_1} \cdots \sum_{y_n} f(x, y_1, ..., y_n) \cdot \mu f_{y_1} \rightarrow f(y_1) \cdots \mu f_{y_n} \rightarrow f(y_n) \tag{2.30}$$

The Max-Product rule can be obtained from the previous equation if the maximization is used as the summary operator.

The algorithm can be applied to both graphs with and without cycles; however, the scheduling is different, as for a graph without cycles it is efficient to begin the message computation from the leaves (i.e. all the variable nodes that are only connected to one parity-check node) and to successively compute messages as their required input messages become available, so that each message is computed exactly once. In a factor graph with cycles, the algorithm becomes iterative. All messages are repeatedly updated, according to some schedule. The computation stops when the available time is over, or when another stopping condition is satisfied, such as finding a valid codeword or reaching a maximum predefined number of iterations; therefore the computation of eq. 2.30 on a graph with cycles is an approximation of the true summary.

The flooding update schedule, which is the most popular when decoding LDPC codes, alternates between updating the messages out of equality constraint nodes and updating the messages out of parity-check nodes. Even though the flooding schedule is very popular, there is no evidence that it is optimal and no evidence that it provides good complexity-performance trade off. Recently, a lazy scheduling approach was studied in which not all messages are updated every iteration, and the probability of updating a message is a function of its reliability and updating history [46]. In [47] an analysis of the convergence rate of the semi-serial and serial schedulers is presented, proving that the computation tree under serial scheduling grows asymptotically in the number of iterations twice as fast compared to the flooding schedule, and that the serial schedule decoder is expected to converge in half the number of iterations compared to the flooding decoder when working near the decoder's capacity.

For the decoding of LDPC codes, in [4] there is a detailed explanation of both algorithms. Let the $local functions$ be the likelihoods at the parity-check nodes, described as $\gamma_e(x)$ (where the subindex e stands for edge), and the likelihoods at the variable nodes, described as $\gamma_n(x)$ (where the subindex n stands for node).

Let the $intermediate functions$ $\mu_{e\to n}(x)$ be the messages (contributions) sent from the set of parity-check nodes to a particular variable node, and $\mu_{n\to e}(x)$ as the messages sent from the set of variable nodes to a particular parity-check node.

Finally, let the $final functions$ $\mu_n(x)$ and $\mu_e(x)$ represent the concentration of the information for a particular variable node or parity-check node

(i.e. $\gamma_n(x)$ and $\gamma_e(x)$ respectively), together with all the contributions coming into that particular variable node or parity-check node, respectively (i.e. $\mu_{e\to n}(x)$ and $\mu_{n\to e}(x)$ respectively).

The Sum-Product algorithm consists of the following three steps:

- Initialization: The local functions $\gamma_n(x)$ and $\gamma_e(x)$ are initialized. The intermediate functions $\mu_{e\to n}(x)$ and $\mu_{n\to e}(x)$ are set to zero.

- Iteration: The intermediate functions $\mu_{e\to n}(x)$ and $\mu_{n\to e}(x)$ are updated alternatively. The variable-to-check function $\mu_{n\to e}(x)$ is computed as the product of the variable's local value and all contributions coming into the variable n, except the one from the parity-check e.

$$\mu_{n\to e}(x) = \gamma_n(x) \prod_{n\in e', e'\neq e} \mu_{e'\to n}(x) \tag{2.31}$$

The check-to-variable function $\mu_{e\to n}(x)$ is computed as the summation over all local configurations on the parity-check e that match the x on the variable n, each term being the product of the parity-check's local cost and all the contributions coming into the parity-check e, except the one from the variable n. The sum

$$\mu_{e\to n}(x) = \sum \gamma_e(x_e) \prod_{n'\in e, n'\neq n} \mu_{n'\to e}(x_{n'}) \tag{2.32}$$

- Termination: The final function $\mu_n(x)$ is computed as the product of the variable's local value and all contributions coming into n. This final

function it is the estimated output value of the codeword.

$$\mu_n(x) = \gamma_n(x) \prod_{n \in e'} \mu_{e' \to n}(x) \tag{2.33}$$

and the final function $\mu_e(x)$ is computed as the product of the check's local value and all contributions coming into e.

$$\mu_e(x) = \gamma_e(x) \prod_{n' \in e} \mu_{n' \to e}(x_{s'}) \tag{2.34}$$

The iteration and termination functions must be modified when the log-likelihood ratio (LLR) 2.35 is considered as the input to the decoder, where z is the channel output. Therefore, the variable-to-check and check-to-variable intermediate functions $\mu_{n \to e}(x)$ and $\mu_{e \to n}(x)$ are defined as eq. 2.36 and eq. 2.37 respectively.

$$L(n) = ln\left(\frac{P(n=0)|z}{P(n=0)|z}\right) \tag{2.35}$$

$$\mu_{n \to e}(x) = \gamma_n(x) \sum_{n \in e', e' \neq e} \mu_{e' \to n}(x) \tag{2.36}$$

$$\mu_{e \to n}(x) = 2tanh^{-1}\left[\prod_{n' \in e, n' \neq n} tanh\left(\frac{1}{2}\mu_{n' \to e}(x_{n'})\right)\right] \tag{2.37}$$

Also, the final cost function changes to eq. 2.38.

$$\mu_n(x) = \gamma_n(x) \sum_{n \in e'} \mu_{e' \to n}(x) \tag{2.38}$$

As an example, consider the (9,2,6) LDPC matrix in eq. 2.39. Assume the bpsk modulated zero codeword has been transmitted, and assume the received log-likelihood vector includes positive values for correct log-likelihoods (i.e. decode to zero), while it includes negative values for incorrect log-likelihoods (i.e. decode to one). The received log-likelihoods are shown in eq. 2.40.

$$\mathbf{H_{(9,2,6)}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.39}$$

$$\{\gamma_n(1), \gamma_n(2), ..., \gamma_n(15)\} = \{+4.0, +4.0, +3.9, -1.5, +2.7, +4.0, +4.0, +4.0, +3.9\} \tag{2.40}$$

The intermediate cost functions $\mathbf{M_{s\rightarrow e(9,2,6)}}$ and $\mathbf{M_{e\rightarrow s(9,2,6)}}$ are initiated, with all their elements equal to zero. After doing the variable-to-check update, and the check-to-variable update, $\mathbf{M_{s\rightarrow e(9,2,6)}}$ and $\mathbf{M_{e\rightarrow s(9,2,6)}}$ take the values in eq. 2.41 and eq. 2.43 respectively. The update of $\mathbf{M_{e\rightarrow s(9,2,6)}}\,(1,1)$ is calculated in eq. 2.42.

$$\mathbf{M_{s\rightarrow e(9,2,6)}} = \begin{bmatrix} +4.0 & 0 & 0 & -1.5 & 0 & 0 & +4.0 & 0 & 0 \\ 0 & +4.0 & 0 & 0 & +2.7 & 0 & 0 & +4.0 & 0 \\ 0 & 0 & +3.9 & 0 & 0 & +4.0 & 0 & 0 & +3.9 \\ +4.0 & 0 & 0 & 0 & 0 & +4.0 & 0 & +4.0 & 0 \\ 0 & +4.0 & 0 & -1.5 & 0 & 0 & 0 & 0 & +3.9 \\ 0 & 0 & +3.9 & 0 & +2.7 & 0 & +4.0 & 0 & 0 \end{bmatrix} \tag{2.41}$$

$$\begin{aligned} \mathbf{M_{e\rightarrow s(9,2,6)}}(1,1) & = 2 \cdot \left(tanh\left(\mathbf{M_{s\rightarrow e(9,2,6)}}(1,4)/2 \right) \cdot tanh\left(\mathbf{M_{s\rightarrow e(9,2,6)}}(1,7)/2 \right) \right) \\ & \cdot \mathbf{M_{e\rightarrow s(9,2,6)}}(1,1) \\ & = 2 \cdot \left(tanh\left(-1.5/2\right) \cdot tanh\left(+4.0/2\right) \right) \mathbf{M_{e\rightarrow s(9,2,6)}}(1,1) \\ & = -1.4 \end{aligned} \tag{2.42}$$

$$\mathbf{M_{e\rightarrow s(9,2,6)}} = \begin{bmatrix} -1.4 & 0 & 0 & +3.3 & 0 & 0 & -1.4 & 0 & 0 \\ 0 & +2.5 & 0 & 0 & +3.3 & 0 & 0 & +2.5 & 0 \\ 0 & 0 & +3.3 & 0 & 0 & +3.2 & 0 & 0 & +3.3 \\ +3.3 & 0 & 0 & 0 & 0 & +3.3 & 0 & +3.3 & 0 \\ 0 & -1.4 & 0 & +3.2 & 0 & 0 & 0 & 0 & -1.4 \\ 0 & 0 & +2.5 & 0 & +3.3 & 0 & +2.5 & 0 & 0 \end{bmatrix} \tag{2.43}$$

Just after one iteration, the final variable cost $\mu_n(x)$ for all the variables, shows a correct decoding, as shown in eq. 2.44.

$$\{\mu_n(1), \mu_n(2), ..., \mu_n(15)\} = \{+5.9, +5.1, +9.7, -5.1, +9.3, +1.1, +5.0, +9.8, +5.7\} \tag{2.44}$$

Density evolution (DE) [40], [48] is a technique to analyse the performance of an LDPC code with particular ensemble parameters (d_v, d_c), or certain ensemble of LDPC code, when decoded with a particular algorithm under a particular channel. This technique determines the corresponding capacity to any desired degree of accuracy; in other words, DE assists to determine the minimum E_b/N_0 required to transmit information at a target bit-error probability, given a maximum number of iterations and a particular decoding algorithm to decode an LDPC code.

There are three principles that support the use of this technique, called *concentration*, *convergence to cycle − free case*, and *density evolution and threshold determination* [8]. The first principle asserts that the behaviour of almost all codes is alike and therefore, the determination of the average behaviour of the ensemble suffices to characterize the individual behavior of the vast mayority of the codes. The second principle suggests that, since the average behaviour of long codes is equal to the behaviour observed on cycle-free graphs, the average behaviour is computable by a deterministic algorithm. Finally, the third principle claims that long codes will exhibit a threshold phenomenon which separates the region where reliable transmission is possible, from that where it is not.

Given the PDF of the variable-to-check update equation $\mathbf{M^0_{n\to e}}$ for the initial values, denoted as $\mathbf{P^0}$, then, the PDF of the check-to-variable update

equation $\mathbf{M^i_{e\rightarrow n}}$ and the PDF of the variable-to-check update equation $\mathbf{M^i_{n\rightarrow e}}$ for any $i \geq 1$, denoted as $\mathbf{Q^i}$ and $\mathbf{P^i}$ respectively, can be computed.

Assuming that the channel is AWGN, the PDF of the received codeword is denoted by eq. 2.45. The all-zero or the all-one codewords can be assumed, if the channel is symmetric. The cumulative distribution function (CDF) (described by eq. 2.46) of the PDF, provides the converge threshold, where for certain E_b/N_0, the CDF provides the same probability of error for any number of iterations.

$$PDF_{Gaussian} = \frac{1}{\sigma\sqrt{2pi}} e\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.45}$$

$$CDF_{Gaussian} = \frac{1}{2}\left(1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right) erf = \frac{2}{\sqrt{pi}}\int_0^x e^{-t^2} dt \tag{2.46}$$

Through observing the evolution of $\mathbf{P^i}$ and $\mathbf{Q^i}$ it is possible to ascertain whether the fraction of an incorrect message approaches zero or not, as the number of iterations increases.

Convergence thresholds at a rate of one half for regular LDPC codes under Log-MAP and Max-Log-MAP decoding are given in Table 2.9. From this graph it becomes clear that as the value of γ and ρ increases, the thresholds also is increased for each particular ensemble, with the best performance obtained by the most sparse ensembles of LDPC matrices.

In [40], DE is introduced for the analysis of GF(q) LDPC codes. In [49], the DE equations for non-binary LDPC ensembles are derived, trying to make use of an additional degree of freedom when compared to binary LDPC

j	k	DE Max-Log-MAP	DE Log-MAP
3	6	1.7	1.11
4	8	2.5	1.62
5	10	3.1	2.04

Table 2.9: Convergente thresholds at $r = 0.5$ for binary-input AWGN channel under Log-MAP and Max-Log-MAP decoding

codes. The results of the study suggests that for a fixed degree distribution, the threshold seems to be a unimodal function of the alphabet size, but as the alphabet size is increased, the performance of the iterative decoder seems not to approach the Shannon limit.

However, DE ignores dependency between bits, which is particularly problematic for specific decoding instances. Also, DE does not show failure to converge for specific decoding instances, as it only shows the average behaviour.

DE is computationally intensive and hence using a Gaussian Approximation (GA), low-complexity approximate analysis methods that track a single parameter of the messages that have been developed through iterations, instead of the density functions. In Gaussian approximation, the mean and the mutual information of the messages exchanged in the decoder, are respectively tracked based on mean [9], and the extrinsic information transfer (EXIT) charts techniques [50], [51], [52].

While Density Evolution and Gaussian Approximation, based on mean techniques maximize the noise threshold for convergence of the iterative decoder as optimization criterion, the EXIT charts use curve fitting as an indirect and sub-optimal way of optimizing for the best convergence threshold, for a given check and variable node degree profiles [10]

The mutual information function of a message whose pdf is symmetric Gaussian density of mean $\sigma^2/2$ and variance σ^2 is defined by eq. 2.47 [53]

$$\begin{aligned} J(\sigma) &= T\left(N(\sigma^2/2, \sigma^2)\right) \\ &= 1 - \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{\left(x-\sigma^2/2\right)^2}{2\sigma^2}} \log_2\left(1+e^{-x}\right) dx \end{aligned} \tag{2.47}$$

The performance of TC and LDPC codes, through the use of EXIT charts, is presented in [53] and [10] respectively. In the last one, eq. 2.48 and eq. 2.49 are presented as an approximated EXIT function for the variable node and the check node respectively, when the LDPC code is regular. I_{av} and I_{ac} are the mutual information of extrinsic output messages of the check nodes, and the mutual information of extrinsic output messages of the variable nodes, respectively; d_l and d_r are the variable node degree and the parity-check node degree, respectively, and σ_{ch} is the variance of the channel messages. When the LDPC code is irregular, eq. 2.48 and eq. 2.49 are modified to include the variable node and parity-check node distribution.

$$T_v = J\left(\sqrt{(d_l - 1)\left(J^{-1}(I_{av})\right)^2 + \sigma_{ch}^2}\right) \tag{2.48}$$

$$T_c = 1 - J\left(\sqrt{d_r - 1} J^{-1}(1 - I_{ac})\right) \tag{2.49}$$

2.5 Design of Iterative Subsystems

As soon as probability propagation algorithms were defined in terms of factor graphs, the concept was used to model advanced receiver designs for particu-

lar types of channels and modulation schemes [7]. Up to that point, communication receivers consisted of serially concatenated processing subsystems. each one optimized to perform a specific task. Such receiver designs had two major drawbacks. Whenever the interface between subsystems involves the passing of hard-decisions, the information lost becomes unavailable to subsequent stages. On top of that, stages at the beginning of the processing chain do not benefit from new information derived by stages further down the chain.

The interpretation of subsystems with factor graphs, such as the modulation demapper, the equalizer, the multiuser detector, the source decoder, etc, with Soft-Input Soft-Output (SISO) algorithms typically in the form of log-likelihood ratios, has proved to improve the BER and PER performance compared to previous receiver designs.

2.5.1 Joint Demapping and Channel Decoding

This scheme considers the concatenation of a channel decoder and a modified demapper where both subsystems exchange soft information. This concept, proposed in [12] and [11] reduces the BER when compared to a conventional receiver where both demapping and channel decoding are carried out independently and the channel decoder is a convolutional code.

By considering this approach, the soft demapping device replaces the second encoder/decoder, but still improves the overall BER performance without further reduction of the coding rate or increased complexity. Different types of mapping are analysed and a design rule based on mutual information

to find good mappings is proposed. It shows through simulations that gray mapping, with the highest unconditioned bit-wise mutual information value $I_0 = 0.55$, has a low performance when used for this scheme, while antigray mapping, with $I_0 \approx 0.32$, has a better performance, although, a mapping with $I_0 \approx 0.41$ would have the best performance, when considering 16QAM.

2.5.2 Joint Source and Channel Decoding

For reasons of delay and complexity and because of highly non stationary sources, many source coding schemes still contain redundancy and output bits of different significance and error sensitivity. Also, most of the source decoders are highly sensitive to channel errors when these are not restricted to a small time or space. This means that when the channel bit error rate is very high, the decoded sequence still have many corrupted bits which could potentially degrade the quality of the application in turn.

The joint source-channel scheme is based on the idea of making use of the source redundancy, which can be modeled as a Markov source, and therefore, can be decoded making use of a modified BCJR algorithm. Different ideas have been proposed for this scheme, some of which use as source a Markov source with different parameters, while others make use of modified versions of source coding standards.

In [54], the case of two correlated binary information sequences that are not compressed by source coding, but are independently channel encoded, is considered where by the correlation between both sequences is exploited at the receiver. One of the conclusions is that the gap between the theoreti-

cally attainable performance and that obtained through simulations increases (the performance degrades) as the correlation between the input sequences increases.

In [55], the information generated by a binary Markov source is sent through a Rayleigh fading channel using complete channel state information. The performance of the system is analised when the source information is not compressed at the transmitter, and the receiver performs a joint source-channel decoding. The main conclusion is that when the source is very redundant, it is better to compress, but as the redundancy of the source decreases the gap is reduced until it becomes better not to compress and instead make use of the little redundancy left on the source which is still high enough to improve the channel decoding performance. The decoder is a modified version of the Viterbi decoder.

[56] presents the general block diagram behind joint source channel-decoding. [57], [58], [59] and [60] introduce joint source channel decoding when residual redundancy is present at the output of speech, video and image encoders respectively. All the proposed joint schemes offer a better performance than the conventional unjoint approach but at the cost of higher complexity, latency and modifications to the interlayer protocols to accept the exchange of soft information in both directions.

[61] shows a different approach by considering a two-state Markov source together with a Low-Density Parity-Check code. It does not only conclude that the joint scheme shows better performance compared to the separate scheme, but also shows that the number of iterations required by the LDPC decoder is greatly reduced. If parallelised with different processors, this im-

plies an overall processing time reduction.

Source Coding

Source coding deals with the task of creating efficient descriptions of information sources, which permit a reduction in the memory or bandwidth resources required to store or transport sample realizations of the source data [62]. A brief explanation of source coding for discrete sources is given here. Regarding discrete sources, the ability to create reduced data-rate descriptions is related to the information content and the statistical correlation among the source symbols. The source coding subsystem goal is to form the best possible fidelity description of the source for a given available bit rate, or to achieve the lowest possible bit rate in order to obtain a specified fidelity description of the source.

A discrete source generates a sequence of symbols $X(k)$ selected from a source alphabet at discrete time intervals kT, where k=1,2,... is a counting index. If the alphabet contains a finite number of symbols, say N symbols, the source is said to be a finite discrete source.

A finite discrete source is defined by an alphabet and the probability assigned to each symbol. Once determined the probability $P(X_j)$ of each symbol X_j, we can determine the self-information $I(X_j)$ for each symbol in the alphabet as eq. 2.50.

$$I(X_j) = -log_2(p_j) \tag{2.50}$$

The average self-information for the symbols in the alphabet, also called

the source entropy, is defined as eq. 2.51,

$$H(X) = \mathbf{E}\{I(X_j)\} = -\sum_{j=1}^{N} p_j log_2(p_j) \tag{2.51}$$

where $\mathbf{E}\{X\}$ is the expected value of X. The source entropy is defined as the average amount of information per source output, and represents the average amount of uncertainty that is resolved by the use of the alphabet. The amount of information in bits per symbol is bounded below by zero if there is no uncertainty, and above by $log_2(N)$ if there is maximum uncertainty.

$$0 \leq H(X) \leq log_2(N) \tag{2.52}$$

Source coding is used when the information content of an N-symbol alphabet is less than the upper bound $log_2(N)$.

A discrete source is said to be memoryless if the symbols emitted by the source are statistically independent. In particular, this means that for the symbols taken two at a time, the joint probability of the two elements is simply the product of their respective probabilities as shown in eq. 2.53. The average entropy per symbol of a statistically independent M-tuple is written as eq. 2.54.

$$P(X_j, X_k) = P(X_j X_k) | P(X_k) = P(X_j) P(X_k) \tag{2.53}$$

$$\begin{aligned} &= \tfrac{1}{M}\mathbf{E}\{-log_2 P(X_1)\} \\ H_M(X) &= \tfrac{1}{M}\textstyle\sum_{X_m}[-P(X_m) log_2 P(X_m)] \\ &= H(X) \end{aligned} \tag{2.54}$$

A discrete source is said to have memory if the symbols emitted by the source are not statistically independent. The dependency between symbols means that in a sequence of M symbols, there is reduced uncertainty about the $M-th$ symbol, if the previous $M-1$ symbols are known. The entropy for a source with memory is the limit in eq. 2.55.

$$H(X) = lim_{M \to \infty} H_M(X) \tag{2.55}$$

The entropy of an M-tuple from a source with memory is always less than the entropy of a source with the same alphabet and symbol probability but without memory. In other words, an M-tuple with dependent symbols contains less information, or resolves less uncertainty, than does one with independent symbols.

$$H_M(X)_{memory} < H_M(X)_{nomemory} \tag{2.56}$$

The average entropy per symbol of an M-tuple from a source with memory, decreases as the length M increases. A consequence is that it is more efficient to encode symbols from a source with memory in groups of several symbols rather than to encode them one symbol at a time. For purposes of source encoding, encoder complexity, memory constraints, and delay considerations, limit the size of a symbol sequences treated as a group.

Source Coding for Digital Data

The coding that takes place to reduce the redundancy of a data source requires the selection of an efficient binary representation of that source. Often

this implies the substitution of one binary representation of the source symbols with an alternative representation. The binary code assigned to each source symbol must satisfy certain constraints to permit reversal of the substitution. For the specific task of data compression, the primary goal is to reduce the number of bits, which in turn, will reduce the redundancy on the data source.

To achieve this task, a complete characterization of the source is required, by obtaining the probability of each symbol and the joint probabilities of the symbols taken two at a time, three at a time, and so on. These non binary symbols are mapped via a dictionary called a character code, to a binary alphabet description.

Data compression codes are often variable-length codes. The length of a binary sequence assigned to each alphabet symbol should be inversely related to the probability of that symbol. A better data compression can be achieved when there is a wide difference between the probabilities of the symbols, compared to the case when the probabilities of the symbols are equal or relatively similar. Also, having a sufficiently large set of symbols increases the data compression. An extension code can be derived from the original set to form a new set of symbols, so that the previous condition can be met. The data compression code should have as desired properties, the capacity to be uniquely decodable and be prefix-free. Compression performance is measured by the compression ratio, which is the ratio of the average number of bits per sample before compression, to the average number of bits per sample after compression.

Recent work in the field include [63] where the BCJR algorithm is anal-

ysed and a new algorithm is derived for rate distortion source coding. Also, in [64], entropy coding and channel coding are merged into a single non-catastrophic encoding operation, where it is claimed that during the event of an incorrectly decoded codeword, there would not be a spread of mistakes.

Huffman Code

The Huffman code [65] is a prefix-free, variable length compression code that can achieve the shortest average code length $\bar{n}$ for a given input alphabet. The possibility of having the shortest average code length for a particular alphabet, to be significantly greater than the entropy of the source alphabet, shows an inability to exploit the promised data compression due to the alphabet, not the coding technique. The Huffman coding procedure provides a transform between any two alphabets and is generated as part of a tree-forming process. The process starts by listing the input alphabet symbols, along with their probabilities, in descending order of occurrence, where each tabular entry correspond to the branch ends of a tree. Each branch is assigned a weight equal to its probability. The two entries with the lowest probabilities are merged to form a new branch with their composite probability. After every merging, the new branch and remaining branches are reordered, always in descending order of occurrence. If at any point, the new branch has equal probability to some other branch, the new branch is given a higher position; this condition reduces the code length variance, which in turn, lowers the chance of buffer overflow.

Markov Chains

In general, the random variables within the family defining a stochastic process are not independent, and in fact can be statistically dependent in very complex ways. However, the class of Markov random processes which are useful in modeling problems, have a simple form of dependence [66].l in modeling problems, have a simple form of dependence [66].

A random process $X(t)$ is a Markov process if the future of the process given the present is independent of the past, that is, if for arbitrary times $t_1 < t_2 < \cdots < t_k < t_{k+1}$,

$$\begin{aligned} &P\left[X\left(t_{k+1}\right) = x_{k+1} | X\left(t_k\right) = x_k, ..., X\left(t_1\right) = x_1\right] \\ &P\left[X\left(t_{k+1}\right) = x_{k+1} | X\left(t_k\right) = x_k\right] \end{aligned} \tag{2.57}$$

if $X(t)$ is discrete-valued, and

$$\begin{aligned} &P\left[a < X\left(t_{k+1} \leq b\right) = x_{k+1} | X\left(t_k\right) = x_k, ..., X\left(t_1\right) = x_1\right] \\ &P\left[a < X\left(t_{k+1} \leq b\right) = x_{k+1} | X\left(t_k\right) = x_k\right] \end{aligned} \tag{2.58}$$

Equations 2.57 and 2.58 define the Markov property where the value of $X(t)$ at time t is referred to as the state of the process at time t while the change between states is called a transition.

A Markov chain is an integer-valued Markov process. If $X(t)$ is a Markov chain, then the joint probability mass function (PMF) for $k+1$ arbitrary time instants is given by eq. 2.59.

$$P\left[X\left(t_{k+1}\right)=x_{k+1}, X\left(t_{k}\right)=x_{k}, ..., X\left(t_{1}\right)=x_{1}\right]$$
$$=P\left[X\left(t_{k+1}\right)=x_{k+1}|X\left(t_{k}\right)=x_{k}\right]P\left[X\left(t_{k}\right)=x_{k}|X\left(t_{k-1}\right)=x_{k-1}\right]\cdots$$
$$\times P\left[X\left(t_{2}\right)=x_{2}|X\left(t_{1}\right)=x_{1}\right]P\left[X\left(t_{1}\right)=x_{1}\right] \tag{2.59}$$

Thus the joint PMF of $X(t)$ at arbitrary time instants is given by the product of the PMF of the initial time instant and the probabilities for the subsequent state transitions. Therefore, it is clear that the state transition probabilities determine the statistical behaviour of a Markov chain.

Let X_n be a discrete-time integer-valued Markov chain that starts at $n=0$ with the PMF given by 2.60.

$$p_j\left(0\right)=P\left[X_0=j\right] \quad \forall j=0,1,2,... \tag{2.60}$$

Assuming that the one-step state transition probabilities are fixed and do not change with time, it takes the general form of eq. 2.61,

$$P\left[X_{n+1}=j|X_n=i\right]=p_{ij} \quad \forall n \tag{2.61}$$

and X_n is said to have homogeneous transition probabilities. The joint PMF for $X_n, ...X_0$ is the given by eq. 2.62

$$P\left[X_n=i_n, ..., X_0=i_0\right]=p_{i_{n-1},i_n}\cdots p_{i_0,i_1}p_{i_0}\left(0\right) \tag{2.62}$$

Thus X_n is completely specified by the initial PMF $p_i\left(0\right)$ and the matrix of one-step transition probabilities depicted by eq. 2.63, where $\mathbf{P}$ is the

transition probability matrix.

$$\mathbf{P} = \begin{bmatrix} p_{00}\ p_{01}\ p_{02}\cdots \\ p_{10}\ p_{11}\ p_{12}\cdots \\ \cdots \\ p_{i0}\ p_{i1}\ p_{i2}\cdots \\ \cdots \end{bmatrix} \tag{2.63}$$

An n-order Markov chain has an n-step transition probability matrix. Considering the state probabilities at time n, let $\mathbf{p}(n) = \{p_j(n)\}$ denote the row vector of state probabilities at time n. The probability $p_j(n)$ is related to $\mathbf{p}(n-1)$ through eq. 2.64.

$$p_j\,(n) = \sum_i P\,[X_n = j | X_{n-1} = i]\,P\,[X_{n-1} = i]\,p_j\,(n) = \sum_i p_{ij} p_i\,(n-1) \tag{2.64}$$

Equation 2.64 states that $\mathbf{p}(n)$ is obtained by multiplying the row vector $\mathbf{p}(n-1)$ by the matrix P. Making a generalization of the previous concept, we have the expression in 2.65.

$$\mathbf{p}\,(n) = \mathbf{p}\,(0)\,P\,(n) = \mathbf{p}\,(0)\,P^n \quad n = 1, 2, ... \tag{2.65}$$

Thus the state PMF at time n is obtained by multiplying the initial state PMF by P^n; in other words, the n-step transition probability matrix is the n-th power of the one-step transition probability matrix.

2.5.3 Serial vs Concatenated codes

A serial concatenation is formed from two codes; an n_1, k_1 code C_1 as inner code and an n_2, k_2 code C_2 as outer code, where in general the inner code is a binary code and the outer code is a nonbinary code with symbols from $GF(2^{k_1})$. The symbols of C_2 are represented by their corresponding bits of k_1 binary symbols (or k_1-tuples). Encoding consists of two steps. First, the $k_1 k_2$ binary information digits are divided into k_2 bytes of k_1 information digits each. These k_2 bytes are encoded according to the rules for C_2 to form an n_2-byte codeword. Second, each k_1-digit byte is encoded into a codeword in C_1, resulting in a string of n_2 codewords of C_1, a total of $n_2 n_1$ digits. These digits are then transmitted, one C_1 codeword at a time, in succession. Thus, the resulting code is an $(n_1 n_2, k_1 k_2)$ binary linear code. The concatenated code of C_1 and C_2, regardless of whether iterative or non-iterative, is decoded in two steps; First each C_1 codeword is decoded and the check digits are removed, leaving a sequence of $n_2 k_1$-digit bytes. These bytes are then decoded according to the decoding method for C_2, to leave the final corrected message.

A parallel concatenation is also formed by two codes; code C_1 and code C_2 have the same n, k. Encoding is implemented in parallel and the codewords must be in systematic. The binary information k are encoded according to the rules for C_1 to obtain n_1 digits. The binary information k is in parallel interleaved and then encoded according to the rules for C_2 to obtain n_2 digits. Since both codes have the same n, k, it is clear that n_1 and n_2 are the same. The transmitted codeword is formed by the information bits,

the parity bits from C_1 with length $n_1 - k$ and the parity bits from $C2$ with length $n_2 - k$; therefore, the length of the transmitted codeword is $k + (n - k) + (n - k) = 2n - k$.

In [67], a study to compare the BER performance of serial versus parallel concatenation is introduced over AWGN and Fading channels, using non-iterative decoding. Although the parallel concatenation has a lower rate, the performance curve of the parallel concatenated approach shows little E_b/N_0 gain. Other studies [68], this time with iterative decoding and using a turbo code as inner code show the same performance for both schemes at lower complexity of the serial concatenation, although it must be considered that the turbo code is a parallel concatenation and that may be the reason of the similar performance.

2.5.4 Parallel concatenated LDPC codes

This scheme results from the direct parallel concatenation of LDPC codes and is presented in detail in [13], [15], [14] and [69]. There are several benefits with such an approach, including an increased redundancy and therefore performance, but with a reduced requirement of memory to allocate a non-concatenated LCPC code of the same length. Also, such an approach can produce better performance in the low E_b/N_0 region, compared to a single LDPC code whose performance in such region depends on the mean column weight $\bar{\gamma}$.

A theoretical model to calculate the BER at low E_b/N_0 values is introduced and a study regarding the performance of this approach with different

LDPC codes is included in [13], where one of them has fixed $\bar{\gamma} = 2$ while the second one varies between $2 < \bar{\gamma} \leq 3$. The term used to describe $\bar{\gamma}$ is mean column weight (MCW).

Finally, this approach does not require the use of an interleaver, compared to the turbo code counterpart, and can stop without going through unnecessary iterations.

2.6 Summary and proposed framework

Turbo codes decoded with the belief-propagation algorithm, have been already in use for some time in practical systems. Irregular LDPC codes can achieve better performance compared to regular LDPC codes and turbo codes, although they may present an error floor at low BER values and have high implementation complexity, due to the irregular composition of the LDPC matrix.

Construction of regular LDPC codes through structured procedures has attracted substantial attention since their performance is similar to, or better than the performance obtained with randomly created regular LDPC codes; other benefit of this approach is the reduced requirement of memory to save the LDPC matrix for some designs, as well as a reduced number of operations for the encoding and decoding processes. However, regular LDPC codes through structured procedures present a limitation on the number of codes with different length, different rate, different column weight, different weight and different girth. Overall, LDPC codes give mayor advantages over turbo codes such as not needing an interleaver and the ability to avoid unnecessary

iterations when a valid codeword is decoded.

The representation of LDPC codes through factor graphs, has assisted to the better understanding of the decoding performance of belief propagation algorithms applied to turbo codes and LDPC codes. Also, factor graph theory has assisted the development of new receivers that are optimum for certain channels; such receivers may take the form of joint iterative subsystems that accept and deliver soft information, making it possible to expand the propagation of messages, previously limited to the channel decoder, to provide the best performance ever achieved. The fact that joint iterative subsystems are highly parallelisable also makes them an attractive option.

For some proposed schemes, it is not improving the BER or FER performance what matters, which are already close to the channel capacity, but to achieve the same performance with reduced complexity (e.g. a reduced number of iterations). Therefore, it has been important to provide new algorithms that are suboptimal (e.g. receivers with a reduced level of complexity).

The main concepts needed to understand the work presented in this thesis related to channel coding and joint iterative subsystems have been introduced. Addition material has been included to assist with the comparison graphs (e.g. Huffman coding).

The two capacity approaching channel encoding/decoding techniques are introduced as well as some regular construction techniques. Of special interest is subsection 2.3.4 which is fundamental for a new type of regular LDPC construction presented in this thesis. The joint iterative subsystem schemes presented in subsections 2.5.4 and 2.5.5 serve as background for a proposed joint scheme without channel encoder/decoder. Subsection 2.5.6 considered

a parallel concatenated Gallager codes scheme where irregular random LDPC codes with different mean column weight $\bar{\gamma}$ are concatenated. The performance of a similar scheme for regular structured LDPC codes is investigated in the thesis.

The novelty of this work relies on the development of an efficient algorithm to construct Quasi-cyclic LDPC codes that are regular, half-rate, with a maximum girth of 16, variable column weight, for different code lengths. EXIT charts to understand their performance are included for different column weights, as well as DE graphs to analise their performance. The performance of several LDPC codes constructed through structured and random algorithms, is analised when concatenated in parallel. The performance of joint demapping-source decoding is analysed independently first, to later incorporate a parallel concatenated channel decoder. The performance of the previous scenarios is analysed under the AWGN channel, with multilevel modulation (i.e. QPSK, 16QAM and 64QAM).

Chapter 3

Quasi-Cyclic Girth Partition and Shift LDPC Codes

3.1 Purpose

Since it was discovered that the error performance achieved by LDPC codes when decoded with the sum-product algorithm [70], [71], approached the channel capacity [72], research was then focussed on the construction of new LDPC codes, the analysis of their performance, and new procedures to make their encoding and decoding more efficient (e.g. reduce the number of flip-flops and reduce the number of gates required, while achieving encoding or decoding in the least possible number of clock cycles).

The importance of developing structured LDPC codes, is to provide a wide family of codes that share certain performance properties, but also to facilitate the development of efficient encoding and decoding algorithms. Cyclic and Quasi-Cyclic (QC) codes [73] [74], [75], [76], [77], [78], [29], [79],

[80] based on sparse matrices are considered a particular type of LDPC codes.

Euclidean Geometry and Projective Geometry codes [81] are an example of Cyclic LDPC codes, while the codes based on arrays of EG or PG LDPC matrices, are an example of QC-LDPC codes. EG and PG LDPC codes have girth g six, while [82] is an example of girth twelve QC-LDPC codes. The structure of Cyclic and QC LDPC codes, allow low complexity encoding [83], [84], with simple shift registers based on their generators or generator matrices, taking advantage of the sectionised cyclic structure, with complexity linearly proportional to the number of parity-check bits of the code for serial encoding, and to the length of the code for high-speed parallel encoding. Also, their structure has an advantage in integrated circuit decoder implementations. References [85], [86], and [87] are similar studies for the decoding part, where low complexity decoding algorithms are proposed.

In this section a novel algorithm to create new regular LDPC codes is proposed. This procedure facilitates the construction of LDPC codes with for a particular column weight and a particular girth. The girth and the length are directly related to each other in this algorithm; therefore, by increasing the length as a parameter of design, the girth can be increased as well. The Quasi-Cyclic LDPC codes created from the new proposed algorithm are related to the Partition-and-shift LDPC codes presented in [18], [17], and [45].

Cyclic and QC LDPC codes will also be defined, in order to establish the nomenclature. The main benefit of this algorithm, is a reduction in the number of operations to generate new QC-LDPC codes for a particular girth, regardless of the length.

3.2 Introduction to QC LDPC codes

A circulant $\mathbf{H_c}$ is a square matrix in which each row is the cyclic shift of the row above it. A cyclic shift is defined as a shift of one position to the right applied to all the values in a vector, where the value in the right most position is then placed in the left most position. This implies that the row at the top of the circulant is the cyclic shift of the row at the bottom of the same circulant. It can be noticed that in a circulant, each column is the downward cyclic shift of the column on its left, with the left most column being the downward cyclic shift of the right most column. The matrix 3.1 shows a 5×5 circulant.

$$\mathbf{H_c} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The row and column weights of a circulant, ρ_c and γ_c respectively, are the same, therefore, it will be refered to, as ω_c. When $\omega_c = 1$, the circulant takes the form of an identity matrix, and is called a circulant permutation matrix. Due to the structure of a circulant, it is completely characterized by any row or any column. The top row is therefore called the generator of the circulant.

If a matrix with dimensions $r_c \times c_c$ fulfils all the conditions described previously, then $r_c = c_c$. For the $r_c \times r_c$ circulant $\mathbf{H_c}$ over GF(2), if its rank

is r_c, then all its rows are linearly independent.

A QC-LDPC matrix is an array of sparse circulants of the same size as shown in 3.2, whose null space generates a QC-LDPC code. For such QC-LDPC matrix $\mathbf{H_{qc}}$ with size $r_{H_{qc}}$ by $c_{H_{qc}}$ over GF(2), positive and integer, with $r_{H_{qc}} \leq c_{H_{qc}}$, the following properties are met.

$$\mathbf{H}_{qc} = \begin{bmatrix} E_{1,1} & E_{1,2} & \cdots & E_{1,c} \\ E_{2,1} & E_{2,2} & \cdots & E_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ E_{r,1} & E_{r,2} & \cdots & E_{r,c} \end{bmatrix} \tag{3.2}$$

The weight of each circulant $\mathbf{H_c}(i,j)$, for all $1 \leq i \leq r_{H_{qc}}$ and for all $1 \leq j \leq c_{H_{qc}}$, is small compared to its size r_c. No two rows (or two columns) of $\mathbf{H_{qc}}$ have more than one 1-component in common, called the *row−column constraint*. The first property implies that each circulant $\mathbf{H_c}(i,j)$ in $\mathbf{H_{qc}}$ is sparse; the second property implies that the girth is at least six, since there are no cycles of length four, and the girth g of an LDPC code is always even. Then, the null space of $\mathbf{H_{qc}}$ is a QC-LDPC code with length $n = c_{H_{qc}} \cdot c_c$.

If $\mathbf{H_{qc}}$ contains $\mathbf{H_c}(i,j)$ with the same ρ_c and γ_c for all $1 \leq i \leq r_{H_{qc}}$ and for all $1 \leq j \leq c_{H_{qc}}$, then $\mathbf{H_{qc}}$ is regular; otherwise, it is irregular.

3.3 Half-rate GPS LDPC Codes with $\rho = 2$

The GPS-LDPC codes belong to the set of QC-LDPC codes. The novel GPS-LDPC codes have column weight γ and row weight $\rho = 2 \cdot \gamma$ and are half rate. The main goal of this algorithm is to obtain the matrix $\mathbf{H_{qc}}$ with

girth $6 \leq g \leq 16$, depending on the requirements of the design. Two new matrices are defined.

The first one is the cycle matrix $\mathbf{Y_{Hqc}}$. Each of the elements in the $\mathbf{Y_{Hqc}}$ matrix is called the circulant identifier (I), as shown in 3.3. $I = \{I_N, I_{D1}, I_{D2}, I_U\}$ is the set of circulant identifiers in the cycle matrix. $I \mapsto H_{qc}$ (i.e. each of the circulant identifiers in the cycle matrix maps to one of the circulants in $\mathbf{H_{qc}}$). The set of circulant identifiers do not take any numerical value.

$$\mathbf{Y_{Hqc}} = \begin{bmatrix} I_{1,1} & I_{1,2} & \cdots & I_{1,g} \\ I_{2,1} & I_{2,2} & \cdots & I_{2,g} \\ \vdots & \vdots & \ddots & \vdots \\ I_{\frac{g}{2},1} & I_{\frac{g}{2},2} & \cdots & I_{\frac{g}{2},g} \end{bmatrix} \tag{3.3}$$

The second one is the slope matrix $\mathbf{S}_H$. The $\mathbf{S}_H$ matrix contains two types of elements called slopes (S) and nulls (N), as shown in 3.4. $S \mapsto H_{qc}$ (i.e. each of the slopes in the slope matrix maps to one of the circulants in $\mathbf{H_{qc}}$). The slope denotes a circulant whose generator has weight one, since the elements of $\mathbf{H_{qc}}$ belong to GF(2). $S = \{0, ..., r_c - 1\} \forall N$ (i.e. the value of the slope takes positive integer values from zero up to $r_c - 1$). When $S = 0$, the slope denotes a circulant's generator whose non-zero element is located in the left most position, and when $S = r_c - 1$, the slope denotes a circulant's generator whose non-zero element is located in the right most position. The value of the slope denotes the position of the non-zero element in a circulant's generator. $N \mapsto \mathbf{H_{qc}}$ (i.e. each of the nulls in the slope matrix maps to one of the circulants in $\mathbf{H_{qc}}$). The null denotes a circulant

whose generator has weight zero, and therefore does not take any numerical value. Also, $I \mapsto \{S, N\}$.

$$\mathbf{S}_H = \begin{bmatrix} S_{1,1} & S_{1,2} & \cdots & S_{1,g} \\ N_{2,1} & S_{2,2} & \cdots & N_{2,g} \\ \vdots & \vdots & \ddots & \vdots \\ N_{\frac{g}{2},1} & S_{\frac{g}{2},2} & \cdots & S_{\frac{g}{2},g} \end{bmatrix} \tag{3.4}$$

The algorithm is divided into three parts. The first one makes use of the cycle matrix, the second one makes use of the shift matrix, and finally the third one is related to the construction of the GPS-LDPC matrix.

During the first part of the algorithm, only g needs to be defined, in order to build the cycle matrix of size $(g/2) \times g$. The cycle matrix contains $g + (g/2)$ circulant identifiers of the type I_{D1} located in the positions $(i, j) \forall 1 \leq i \leq (g/2), 1 \leq j \leq g$, when $j = i$, $j = i + (g/2)$ or $j = (i - 1 \bmod (g/2)) + (g/2)$. The cycle matrix contains at the beginning of the algorithm one circulant identifier of the type I_{D2} located in the position $(2, 1)$. Also, the cycle matrix contains $(g/2) - 1$ circulant identifiers of the type I_U located in the positions $(i, j) \forall 1 \leq i \leq (g/2), i \neq 2, 2 \leq j \leq (g/2)$, when $i = (j + 1) \bmod j$. Finally, the cycle matrix contains $g \times (g/2) - (2 \cdot g)$ circulant identifiers of the type I_N in the positions left. The number and position of the I_N and I_{D1} circulant identifiers does not vary at any point during the algorithm, while the circulant identifiers I_U will be changed to I_{D2} (i.e. one by one), in accordance with the algorithm described next. The cycle matrix shown in 3.5 is an example when the girth required for the GPS-LDPC matrix is six and $\gamma = 2$.

$$\mathbf{Y}_{\mathbf{H}qc} = \begin{bmatrix} I_{D1} & I_N & I_U & I_{D1} & I_N & I_{D1} \\ I_{D2} & I_{D1} & I_N & I_{D1} & I_{D1} & I_N \\ I_N & I_U & I_{D1} & I_N & I_{D1} & I_{D1} \end{bmatrix} \tag{3.5}$$

The first part of the algorithm chooses any I_U as the starting cycle identifier and searches on the cycle matrix for all the possible cycles whose length is shorter than the girth proposed. The total number of cycle identifiers included in a cycle is called the length. A section is a vertical or horizontal path that joins any two cycle identifiers on the cycle matrix. A cycle is a path with the same starting and ending I_U cycle identifier, that fulfils the following conditions.

- The starting and ending cycle identifier of a cycle can be any I_U cycle identifier on the cycle matrix.
- The first and last sections in the cycle that meet at the same I_U cycle identifier, should be perperdicular to each other.
- A cycle should include only sections.
- A cycle should not include any I_N cycle identifiers.
- A cycle should not include any I_U cycle identifiers, other than the I_U cycle identifier chosen as the starting cycle identifier.
- A cycle has no limit in the number of I_{D1} cycle identifiers included in the path.
- A cycle has no limit in the number of I_{D2} cycle identifiers included in the path.

- A cycle includes at least four cycle identifiers.

- A cycle should not include more than four consecutive I_{D1} cycle identifiers in the path, if the first and the last one are the same I_{D1} cycle identifier (i.e. if the the first and the last I_{D1} are the same cycle identifier in the position $I_{i,j}$, and if and only if the cycle identifiers between them are three or more I_{D1} cycle identifiers, then such a path should not be considered a cycle).

- For any cycle identifier to be included in a cycle, the section with the proposed cycle identifier should be parallel to the previous section.

- Since there is no defined section at the beginning of the cycle search, any cycle identifier that fulfils the conditions previously stated, can be included as part of the cycle (i.e. the second cycle identifier in the path that forms a cycle, starting from a I_U should be of the type I_{D1}, or I_{D2} and should be located in the same row or in the same column within the cycle matrix.

- The length of a cycle should be shorter than the value of the girth proposed (i.e. the number of cycle identifiers should be smaller than the value of the girth).

For the better understanding of the previous conditions, assume the cycle matrix has as elements, circulants represented by the cycle identifiers, where each type of cycle identifier represents a different type of circulant. For this reason, it is necessary to redefine the concept of cycle, which was originally defined in section 2.3.

After all the possible cycles that include the same I_U as the beginning cycle identifier have been searched for, such cycle identifier I_U changes to I_{D2}. This process is repeated for each of the I_U cycle identifiers in the cycle matrix, until all of them have been changed to I_{D2}.

It must be noted that the definition of cycle and length provided here is different to the definition that is used for an LDPC matrix. All the cycles that include a particular I_U cycle identifier with position $I_{i,j}$ in the cycle matrix, as the starting point, should be saved on a database, called the cycle database ($CDB_{i,j}$).

$$\mathbf{Y_{Hqc}} = \begin{bmatrix} I_{D1} & I_N & I_U & \leftarrow & I_{D1} & \leftarrow & I_N & \leftarrow & I_{D1} \\ & & \downarrow & & & & & & \uparrow \\ I_{D2} & I_{D1} & I_N & & I_{D1} & & I_{D1} & & I_N \\ & & \downarrow & & & & & & \uparrow \\ I_N & I_{D2} & I_{D1} & \rightarrow & I_N & \rightarrow & I_{D1} & \rightarrow & I_{D1} \end{bmatrix} \tag{3.6}$$

The cycle matrix 3.6 shows the cycle $\{I_{1,3}, I_{3,3}, I_{3,6}, I_{1,6}, I_{1,3}\}$ in the cycle matrix. Such a cycle should be saved on $CDB_{1,3}$. From 3.6, it is inferred that $CDB_{3,2}$ has been completed and therefore $I_{3,2}$ has changed from I_U to I_{D2}. Since $I_{3,2}$ is now an I_{D2} cycle identifier, it can be considered for any cycle starting at $I_{1,3}$. The only cycles for 3.6 are:

- $\{I_{3,2}, I_{3,5}, I_{2,5}, I_{2,2}, I_{3,2}\}$
- $\{I_{1,3}, I_{3,3}, I_{3,6}, I_{1,6}, I_{1,3}\}$

Table 3.1 contains the number of cycles per girth and per cycle identifier,

in the same order considered as initial item on a cycle, to be then changed from I_U to I_{D2}. The number of cycles grows exponentially for every pair of different girth values. This is because most of the cycles are formed as multiples of four, when $\gamma = 2$ (e.g. when the girth is eight, the algorithm looks for cycles of length six, but since most of the cycles have length four, not many more cycles with length six are found in the cycle matrix, to be added to the cycle dababase).

Girth	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	Total
$g = 6$	1	1						2
$g = 8$	1	1	1					3
$g = 10$	4	4	4	6				18
$g = 12$	4	4	4	4	6			22
$g = 14$	11	19	19	19	21	43		132
$g = 16$	11	19	19	19	19	21	43	151

Table 3.1: Number of cycles per girth and per cycle identifier, when $\gamma = 2$

For the second part of the algorithm the length of the code n must be chosen, in order to build the slope matrix of size $(g/2) \times g$. The only condition that the length should meet is v mod $(g/2)$=0, where v is the number of rows in the GPS-LDPC matrix, and $v = n/2$. Parameters n and v must be positive integers. The slope matrix contains $2g$ slopes and $(g-4)g/2$ nulls. The nulls in the slope matrix have the same position as the I_N in the cycle matrix, the slopes with the value zero assigned $(S = 0)$, should have the same position as the I_{D1} in the cycle matrix, whereas the rest of the slopes in the slope matrix should remain without any assigned value and they have the same position as the I_{D2} and the I_U in the cycle matrix (e.g. based on the cycle matrix 3.5, when the girth is six, the slope matrix is displayed in 3.7).

$$\mathbf{S}_H = \begin{bmatrix} S_{1,1}=0 & N & S_{1,3} & S_{1,4}=0 & N & S_{1,6}=0 \\ S_{2,1} & S_{2,2}=0 & N & S_{2,4}=0 & S_{2,5}=0 & N \\ N & S_{3,2} & S_{3,3}=0 & N & S_{3,5}=0 & S_{3,6}=0 \end{bmatrix} \quad (3.7)$$

A second database contains all the possible shifts except zero, that can be assigned to each slope that has not been assigned a value yet (i.e. $1 \leq S \leq r_c - 1, \forall N$, with $r_c = n/g$). Such a database is called the slope database $SDB_{i,j}$. (e.g. according to the previous statements, on the slope matrix 3.7, the slope databases are $SDB_{2,1}$, $SDB_{3,2}$, and $SDB_{1,3}$; assuming $n = 30$, and $g = 6$ we have $r_c = 5$, and $SDB = 1, 2, 3, 4$).

The number and position of the nulls and the slopes does not vary at any point during the algorithm. The value the slopes will be assigned is in accordance with the following steps:

1. The first slope chosen is $S_{2,1}$, and is assigned temporarily the first value included in the $SBD_{2,1}$.

2. $S_{2,1}$ cannot produce on its own cycles shorter than the proposed girth, and therefore all the shifts included in the $SDB_{2,1}$ remain.

3. The slopes will be analysed following the same order the I_{DU} cycle identifiers on the cycle matrix were chosen to search for cycles. The idea of the analysis of the value of a slope, is to delete from their corresponding slope database those shifts that would generate a cycle on the shift matrix (e.g. on 3.7, the first slope to be analysed is $S_{3,2} = 1$, since $SDB_{3,2} = 1, 2, 3, 4$).

4. After choosing the slope to be analysed, based on the corresponding cycle database, any shift included in the corresponding slope dababase is added $modulo((n/g)-1)$ or substracted $modulo((n/g)-1)$ according to all the cycles in the cycle database, that start with the cycle identifier located in the same position (e.g. on 3.7, after choosing $I_{3,2}$, the corresponding slope database is $SDB_{3,2}$, and the cycles on the cycle database to be considered, are those ones that start with $I_{3,2}$).

5. The slopes included in the cycle with odd positions, have the value of their shift assigned, added $modulo((n/g)-1)$. (e.g. on 3.7, for $S_{3,2}$, $\{I_{3,5}, I_{2,2}\}$ are added $modulo((n/g)-1)$).

6. The slopes incuded in the cycle with even positions, have the value of their shift assigned, substracted $modulo((n/g)-1)$. (e.g. on 3.7, for $S_{3,2}$, $\{I_{2,5}, I_{3,2}\}$ are substracted $modulo((n/g)-1)$).

7. After all the values, corresponding to each one of the cycle identifiers, have been either added or substracted according to steps 5 and 6, if the resulting value equals zero, the value of the shift chosen from the slope database to be temporarilly assigned to the slope being analysed, is deleted. If the slope analysed is deleted, the algorithm returns to step 4 until one of the shifts doesn't need to be deleted, or until all the shifts on the slope database for the slope being analysed, have been considered. (e.g. on 3.7, $S_{3,2} = 1$, is the first value temporarilly assigned from the $SDB_{3,2} = 1, 2, 3, 4$. For the cycle $\{I_{3,2}, I_{3,5}, I_{2,5}, I_{2,2}, I_{3,2}\}$, the result after steps 5 and 6 equals $+0-0+0-1 = -1$; therefore, the shift with value one is not deleted from the slope database $SDB_{3,2}$, and therefore

there is no need to repeat the procedure for the shift with value two).

8. For the slope being analysed, if all the shifts in the slope database have been deleted, the algorithm returns to 1 and the next shift value included in the $SDB_{2,1}$ is assigned to $S_{2,1}$.

9. After a shift in the slope database has been analysed and has not been deleted, it is assigned permanently to the corresponding slope. The algorithm then returns to step 4 to analyse which shifts on the slope database corresponding to the next slope can generate a cycle.

10. The algorithm stops when all the slopes have been assigned a shift value from their corresponding slope database; if all the shift values in any of the slope databases have been deleted, and all the shift values in $SDB_{2,1}$ have been considered, there is no GPS-LDPC matrix that fulfils the requirements. This means that the value of the length should be increased or the value of the girth should be decreased to successfully obtain all the required slope values.

For the third part of the algorithm, if all the slopes were successfully assigned a shift value, then the corresponding GPS-LDPC matrix can be constructed. Since $N \mapsto H_{qc}$ and $S \mapsto H_{qc}$ as previously established, the GPS-LDPC matrix includes a circulant whose generator is the $\mathbf{0}$=(0,0,...,0) v/g-tuple (i.e. an all-zero squared matrix) for each of the N in $\mathbf{S}_H$; the GPS-LDPC matrix includes a circulant whose generator is the $\mathbf{1}$=(1,0,...,0) v/g-tuple (i.e. an identity matrix) for each of the $S = 1$ in $\mathbf{S}_H$; finally, the GPS-LDPC matrix includes a circulant whose generator is based on the shift

value assigned to the corresponding slope in $\mathbf{S}_H$. (e.g. for $S_{2,1} = 1$, $S_{3,2} = 2$, and $S_{1,3} = 3$ on the slope matrix in 3.7, after doing $N \mapsto H_{qc}$ and $S \mapsto H_{qc}$, the corresponding $\mathbf{H_{qc}}$ is shown in 3.8).

$$\mathbf{H}_{qc} = \left[\begin{array}{cccccc}
10000 & 00000 & 00010 & 10000 & 00000 & 10000 \\
01000 & 00000 & 00001 & 01000 & 00000 & 01000 \\
00100 & 00000 & 10000 & 00100 & 00000 & 00100 \\
00010 & 00000 & 01000 & 00010 & 00000 & 00010 \\
00001 & 00000 & 00100 & 00001 & 00000 & 00001 \\
 & & & & & \\
01000 & 10000 & 00000 & 10000 & 10000 & 00000 \\
00100 & 01000 & 00000 & 01000 & 01000 & 00000 \\
00010 & 00100 & 00000 & 00100 & 00100 & 00000 \\
00001 & 00010 & 00000 & 00010 & 00010 & 00000 \\
10000 & 00001 & 00000 & 00001 & 00001 & 00000 \\
 & & & & & \\
00000 & 00100 & 10000 & 00000 & 10000 & 10000 \\
00000 & 00010 & 01000 & 00000 & 01000 & 01000 \\
00000 & 00001 & 00100 & 00000 & 00100 & 00100 \\
00000 & 10000 & 00010 & 00000 & 00010 & 00010 \\
00000 & 01000 & 00001 & 00000 & 00001 & 00001
\end{array}\right] \tag{3.8}$$

The maximum achievable girth for this algorithm to construct a QC-LDPC is $g = 16$. Fig. 3.1 shows that for a cycle matrix, assuming girth 18, no shift value remains in the slope database after applying step 5 onwards; the cycle identifiers are: $I_{3,2}$, $I_{3,11}$, $I_{2,11}$, $I_{2,1}$, $I_{1,1}$, $I_{1,10}$, $I_{2,10}$, $I_{2,11}$

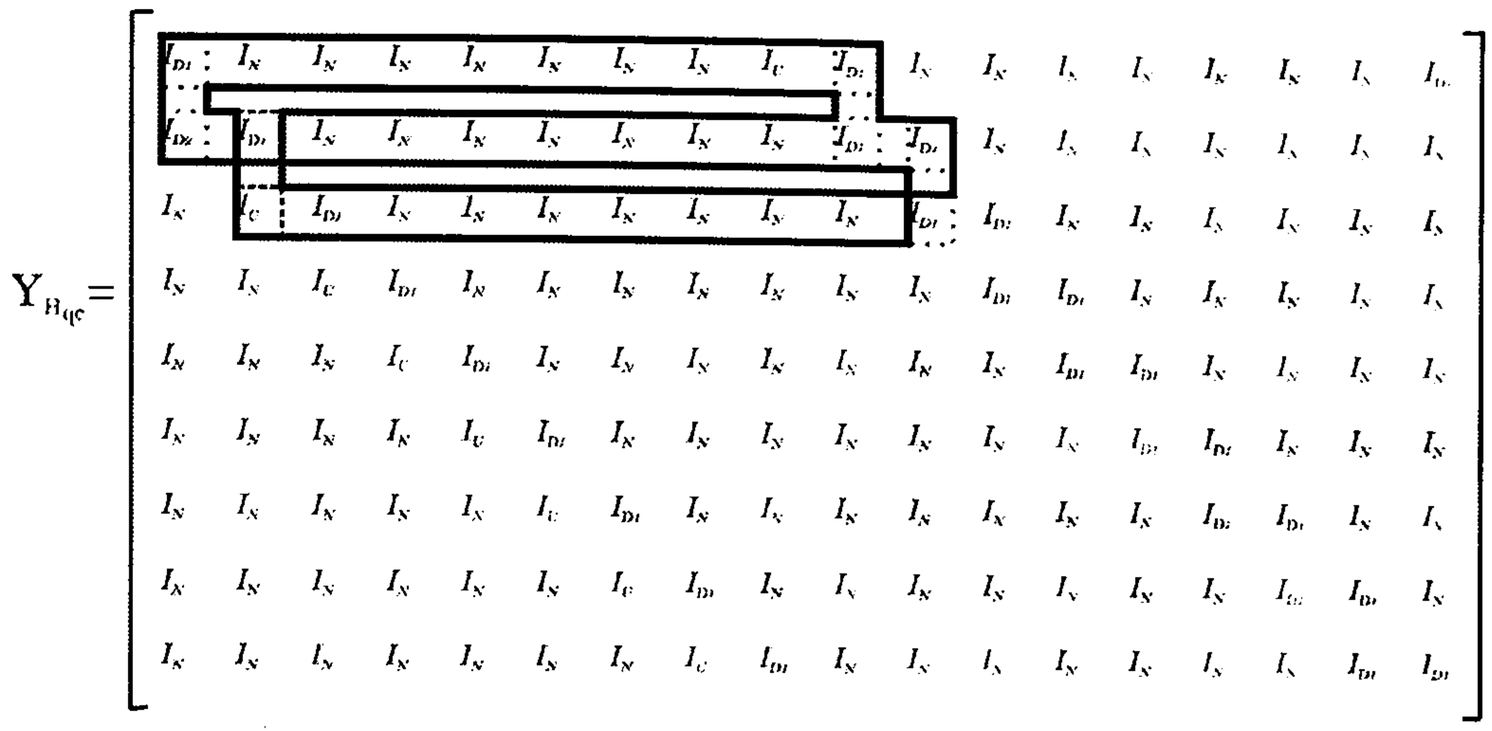

Figure 3.1: Cycle matrix when $g = 18$, including limiting length 16 cycle

$,I_{3,11},\ I_{3,2},\ I_{2,2},\ I_{2,10},\ I_{1,10},\ I_{1,1},\ I_{2,1},\ I_{2,2},\ I_{3,2}$. All the cycle identifiers included in this cycle are $I_{D1} \mapsto S = 0$ except $I_{2,1}$ and $I_{3,2}$, which are I_{D2}. Assuming $I_{2,1} \mapsto S = X_1$ and $I_{3,2} \mapsto S = X_2$, and after applying steps 5, 6 and 7, the addition of shifts equals zero for any value of X_1 and X_2 (i.e. $\{+0 - 0 + X_1 - 0 + 0 - 0 + 0 - 0 + X_2 - 0 + 0 - 0 + 0 - X_1 + 0 - X_2 = 0\}$ $\forall\{X_1, X_2\} \in N$).

Table 3.2 includes the slopes for each girth from 6 up to 16. In particular, the new algorithm looks for the minimum code length that had a solution on each case. (i.e. there is no solution, if the proposed code length is shorter than the one presented here and there are no changes to the initial slopes selected).

To compare the performance of the GPS-LDPC codes, the GPS(672,2,336), and the GPS(4368,2,2184) LDPC codes with g =16, and 18 respectively, are generated. They are compared in Fig. 3.2, and Fig. 3.3 against the graphical-model (GM) based LDPC codes with the identical parameters respectively; both LDPC code constructions are $(n, 2, k)$ codes with half rate. For the sim-

Girth	Length	Slopes
6	12	$S_{2,1} = 1$, $S_{3,2} = 1$, $S_{1,3} = 1$
8	16	$S_{2,1} = 1$, $S_{3,2} = 1$, $S_{4,3} = 1$, $S_{1,4} = 1$
10	70	$S_{2,1} = 1$, $S_{3,2} = 2$, $S_{4,3} = 1$, $S_{5.4} = 2$, $S_{1,5} = 3$
12	60	$s_{2,1} = 1$, $S_{3,2} = 2$, $S_{4,3} = 1$, $S_{5,4} = 2$, $S_{6,5} = 1$, $S_{1,6} = 2$
14	224	$S_{2,1} = 1$, $S_{3,2} = 3$, $S_{4,3} = 7$, $S_{5,4} = 2$, $S_{6.5} = 3$, $S_{7.6} = 4$, $S_{1,7} = 6$
16	272	$S_{2,1} = 1$, $S_{3,2} = 3$, $S_{4,3} = 5$, $S_{5,4} = 1$, $S_{6,5} = 3$, $S_{7,6} = 5$, $S_{8,7} = 4$, $S_{1,8} = 6$

Table 3.2: Slopes within the slope matrix for different length codes and different girth

ulations, the sum-product decoding algorithm was employed over an AWGN channel. The maximum number of iterations was 60.

The GPS-LDPC code shows better performance compared to the RG-LDPC code for $E_b/N_0 > 4.4$ dB; when compared against the GM-LDPC code, the GPS-LDPC code shows a worse performance for all the E_b/N_0 values investigated. A constant gap of approximately 0.25 dB between both BER curves occurs for all the code lengths analised. The gap between the GPS LDPC codes with length n=672, and 4368, and the channel capacity measured for a BER of 10^{-3}, is 3.4 and 3.1 dB respectively.

The performance of the new GPS LDPC codes was also compared over a flat Rayleigh fading channel, after 60 iterations for QPSK modulation. Once again, the GM-LDPC codes show better BER and FER that the GPS-LDPC codes with a gap of 2 dB between the BER and FER performance curves, for n=672. A comparison of the E_b/N_0 required by the GPS-LDPC codes, with length n=672, to achieve a target BER of 10^{-3} when analised over the flat Rayleigh fading channel and over the AWGN channel, show a gap of 3 dB and 2.5 dB respectively.

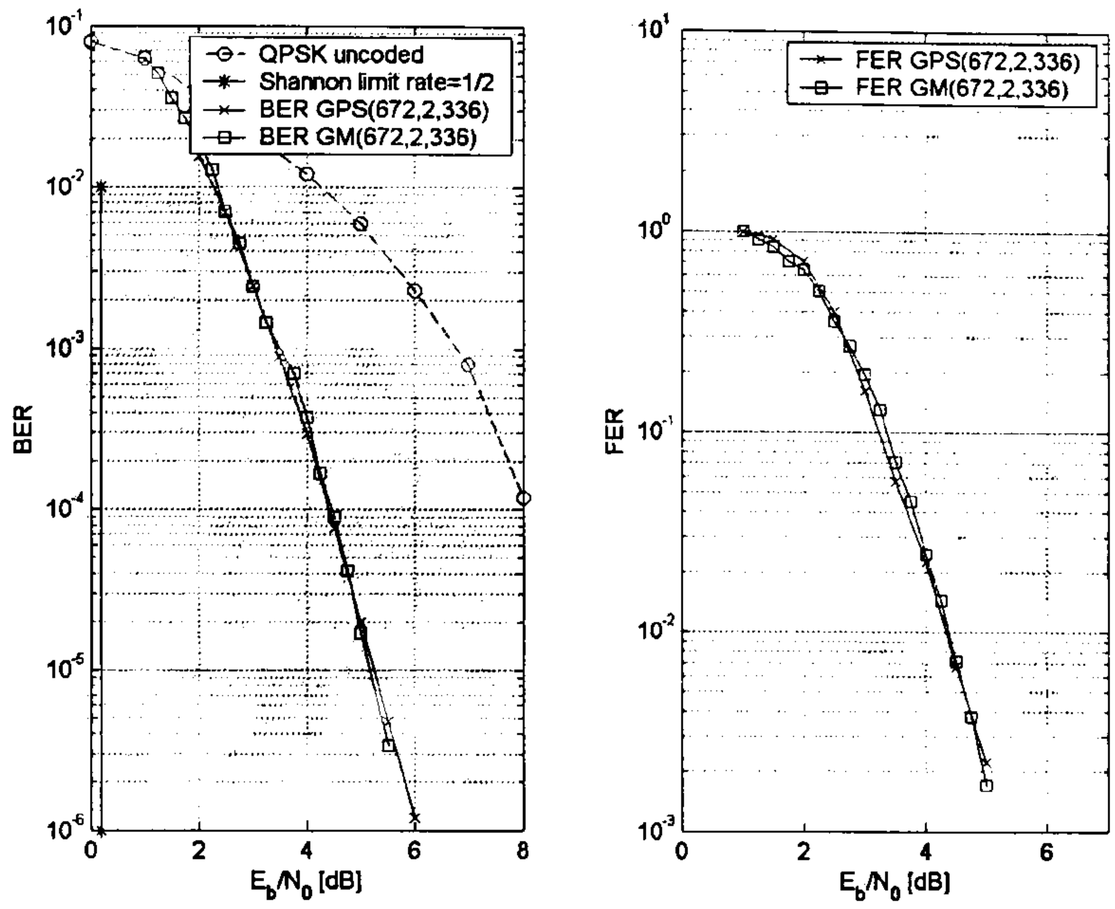

Figure 3.2: BER (left) and FER (right) performance of the GPS(672,2,336) and GM(672,2,336) LDPC codes. Maximum 60 iterations over AWGN channels, and QPSK modulation

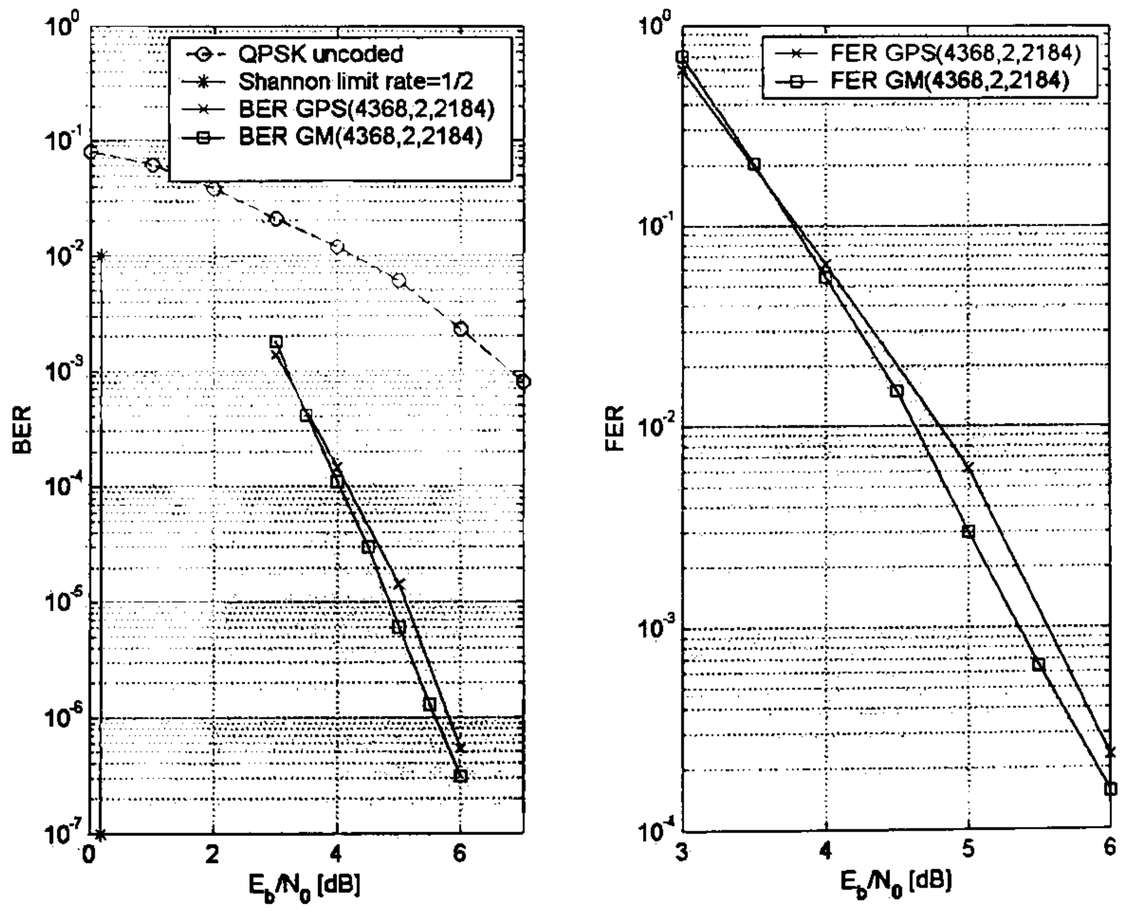

Figure 3.3: BER (left) and FER (right) performance of the GPS(4368,2,2184) and GM(4368,2,2184) LDPC codes. Maximum 60 iterations over AWGN channels, and QPSK modulation

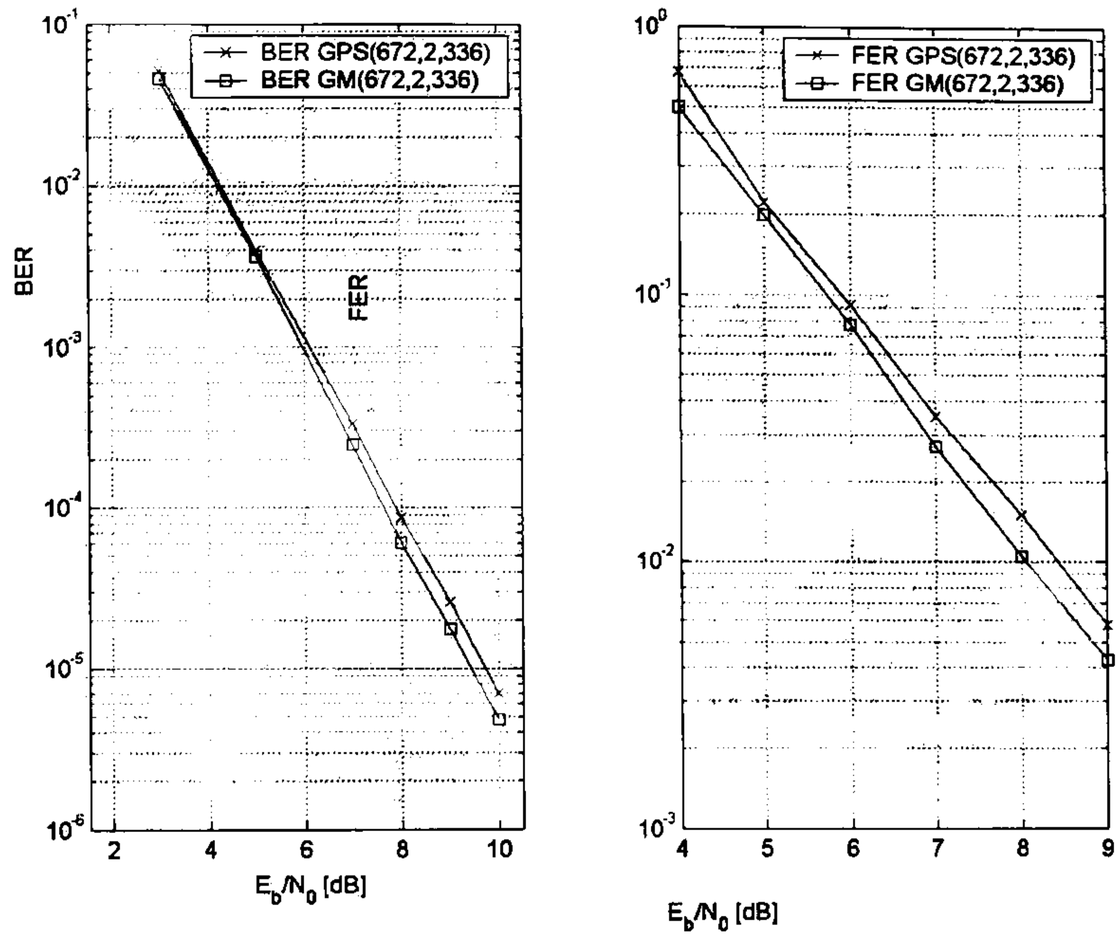

Figure 3.4: BER (left) and FER (right) performance of the GPS(672,2,336) and GM(672,2,336) LDPC codes. Maximum 60 iterations over the flat Rayleigh fading channel, and QPSK modulation

3.4 Half-rate GPS LDPC Codes with $\rho = 3$

When $\gamma = 3$, the number of circulant identifiers, slopes and nulls, as well as their location, change. However, once the cycle matrix is initialised adequately, the rest of the algorithm is applied without modifications. The distribution of the circulants on the LDPC matrix proposed, which varies as a function of the girth and the column weight as a design parameters, is part of the novelty of this work.

In this case, the cycle matrix contains $g + (g/2)$ circulant identifiers of the type I_{D1} located in the positions (i, j), $\forall\ (1 \leq i \leq (g/2)), (1 \leq j \leq g)$, when $j = i$, $j = (i - 1) \bmod (g/2)$, or $j = i + (g/2)$. At the beginning of the algorithm, the cycle matrix contains one circulant identifier of the type I_{D2} located in the position $(3, 1)$. The cycle matrix contains $g+(g/2)-1$ circulant

identifiers of the type I_U located in the positions (i, j), $\forall\ (1 \leq i \leq (g/2))\ , i \neq 2$, $(2 \leq j \leq (g/2))$, when $i = (j+1) \bmod j$, or $j = (i-1 \bmod (g/2)) + (g/2)$. Finally, the cycle matrix contains $g \times (g/2) - (3 \cdot g)$ circulant identifiers of the type I_N in the positions left. The number and position of the I_N and I_{D1} circulant identifiers does not vary at any point during the algorithm, while the circulant identifiers I_U will be changed to I_{D2}, following the same steps presented for $\gamma = 2$. Eq. 3.9 shows the cycle matrix when the girth required for the GPS-LDPC matrix is ten and $\gamma = 3$.

$$\mathbf{Y}_{\mathbf{H_{qc}}_{qc}} = \begin{bmatrix} I_{D1} & I_N & I_N & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} \\ I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D1} & I_N & I_N & I_{D2} \\ I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D1} & I_N & I_N \\ I_N & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D1} & I_N \\ I_N & I_N & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D1} \end{bmatrix} \tag{3.9}$$

Table 3.3 contains the number of cycles per girth and per cycle identifier, in the same order considered as an initial item on a cycle, to be then changed from I_U to I_{D2}. The number of cycles grows exponentially as the value of the girth increases.

Girth	Total
$g = 8$	210
$g = 10$	1567
$g = 12$	11785
$g = 14$	104653

Table 3.3: Number of cycles per girth, when $\gamma = 3$

After the cycles are saved in the cycle database, the length of the code n

must be chosen, in order to build the slope matrix of size $(g/2) \times g$, with the same condition (i.e. v mod $(g/2)$=0). The slope matrix contains $3g$ slopes and $(g-6)g/2$ nulls. The number and position of the nulls and the slopes does not vary at any point during the algorithm. Once all the slopes have been assigned a value, the GPS-LDPC matrix is formed. If not all the slopes were assigned a value, then, the length of the code should be increased, or the value of girth should be reduced, until a solution is found.

The maximum achievable girth to construct a QC-LDPC when $\gamma = 3$ remains $g = 16$. This is because the limit on the maximum achievable girth depends entirely on the number of I_{D1} cycle identifiers.

Table 3.4 includes the slopes for each girth from 8 up to 14. In particular, codes whose length is close to the minimum achievable were searched for, while the minimum length is included in brackets. (i.e. while the lengths are very similar, the slopes that are valid for one length, may not be valid for a different one, resulting in an LDPC matrix with a girth smaller than that desired, if not analysed on a length basis).

The reason for using this distribution, is to reduce the number of operations required by the algorithm to calculate all the cycles with a shorter girth; this translates into a reduced number of operations to obtain the QC-LDPC for the girth required. Also, since the cycles for a particular girth are kept in the database cycle, it is not necessary to calculate them every time a new code is to be built.

To compare the performance of the GPS-LDPC codes, the GPS(672,3,336), and the GPS(4368,3,2184) LDPC codes whose parity-check matrices have g =8 and 12, respectively, were generated. They are compared in Fig. 3.5,

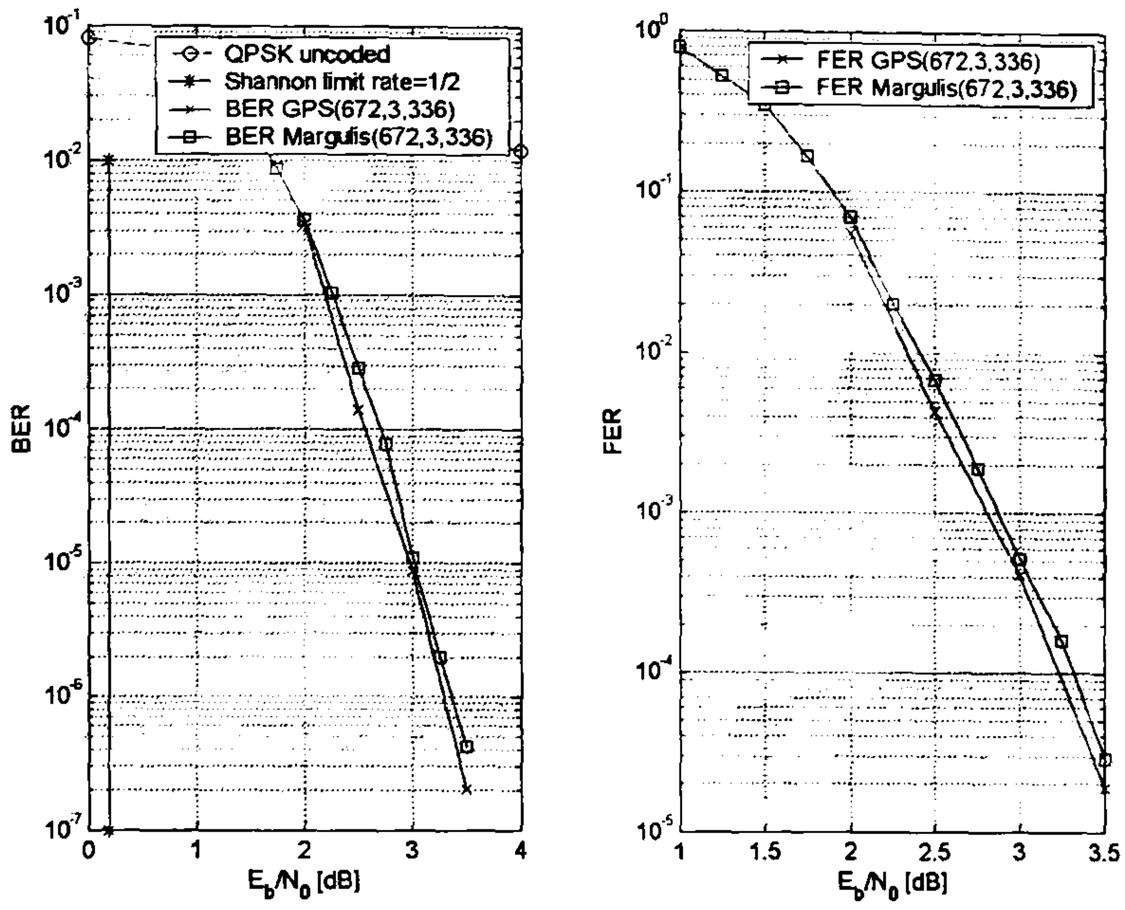

Figure 3.5: BER (left) and FER (right) performance of the GPS(672,3,336) and Margulis(672,3,336) LDPC codes. Maximum 60 iterations over AWGN channels, and QPSK modulation

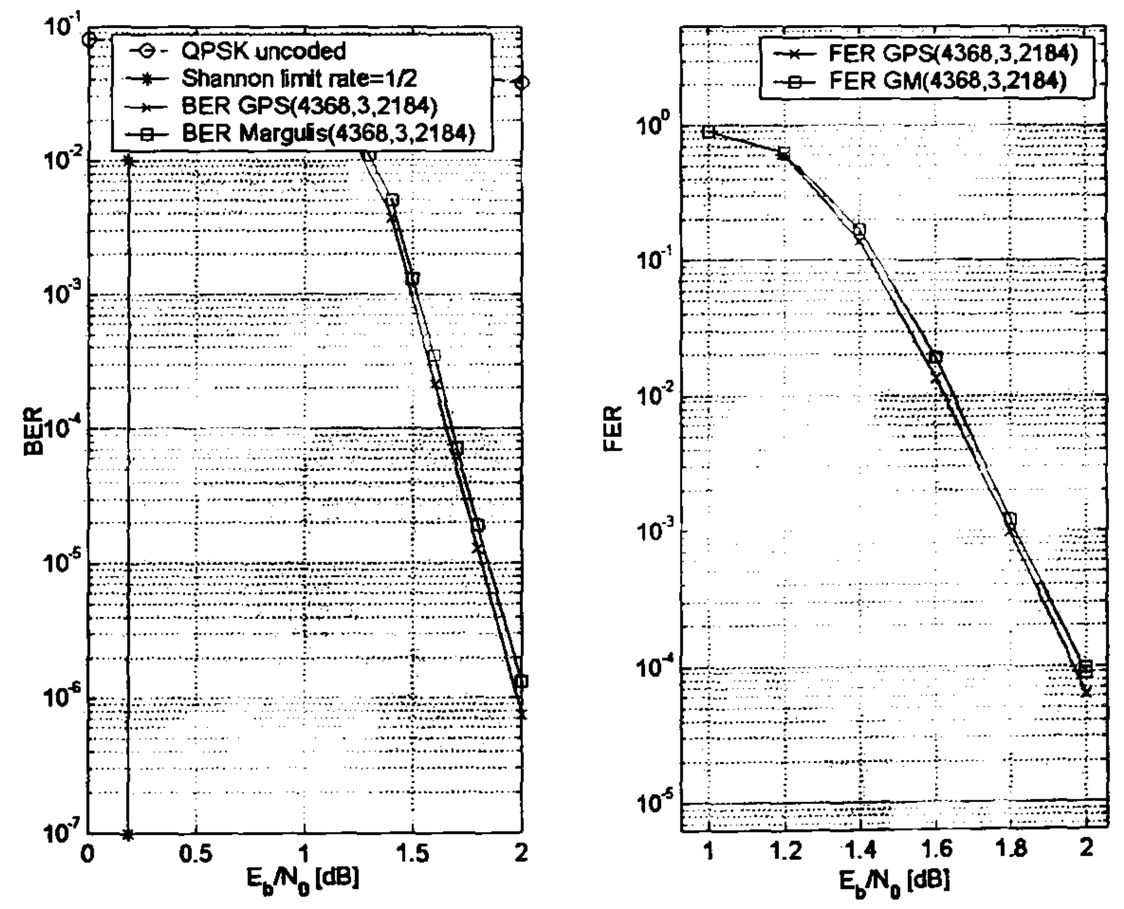

Figure 3.6: BER (left) and FER (right) performance of the GPS(4368,3,2184) and Margulis(4368,3,2184) LDPC codes. Maximum 60 iterations over AWGN channels, and QPSK modulation

Girth	Length	Slopes
8	160 (152)	$S_{3,1} = 1$, $S_{4,2} = 2$, $S_{1,3} = 1$, $S_{2,4} = 2$ $S_{2,5} = 3$, $S_{3,6} = 3$, $S_{4,7} = 3$, $S_{1,8} = 3$ $S_{3,5} = 7$, $S_{4,6} = 8$, $S_{1,7} = 7$, $S_{2,8} = 8$
10	1000 (830)	$S_{3,1} = 1$, $S_{4,2} = 2$, $S_{5,3} = 3$, $S_{1,4} = 2$, $S_{2,5} = 4$ $S_{2,6} = 7$, $S_{3,7} = 6$, $S_{4,8} = 10$, $S_{5,9} = 4$, $S_{1,10} = 10$ $S_{3,6} = 22$, $S_{4,7} = 33$, $S_{5,8} = 25$, $S_{1,9} = 20$, $S_{2,10} = 34$
12	4200 (3480)	$S_{3,1} = 1$, $S_{4,2} = 3$, $S_{5,3} = 5$, $S_{6,4} = 1$, $S_{1,5} = 3$, $S_{2,6} = 5$ $S_{2,7} = 13$, $S_{3,8} = 15$, $S_{4,9} = 21$, $S_{5,10} = 13$, $S_{6,11} = 15$, $S_{1,12} = 21$ $S_{3,7} = 45$, $S_{4,8} = 73$, $S_{5,9} = 69$, $S_{6,10} = 105$, $S_{1,11} = 73$, $S_{2,12} = 122$
14	140000	$I_{3,1} = 1$, $I_{4,2} = 3$, $I_{5,3} = 7$, $I_{6,4} = 12$, $I_{7,5} = 1$, $I_{1,6} = 14$, $I_{2,7} = 24$, $I_{2,8} = 16$, $I_{3,9} = 42$, $I_{4,10} = 55$, $I_{5,11} = 68$, $I_{6,12} = 43$, $I_{7,13} = 59$, $I_{1,14} = 100$, $I_{3,8} = 180$, $I_{4,9} = 340$, $I_{5,10} = 549$, $I_{6,11} = 704$, $I_{7,12} = 412$, $I_{1,13} = 771$, $I_{2,14} = 1061$

Table 3.4: Slopes within the slope matrix for different length codes and different girths

and Fig. 3.6 against a Margulis based LDPC codes [88], [44]; both ensembles are $(n, 3, k)$ codes with half rate. For the simulations, the sum-product decoding algorithm was employed over an AWGN channel. The maximum number of iterations was 60.

The GPS-LDPC code shows similar BER performance and an small improvement on the FER performance compared to the Margulis LDPC code for all E_b/N_0 values investigated; however, the gap between both codes is less than 0.1 dB for the different code lengths depicted. The gap between the GPS LDPC codes with length n=672, and 4368, and the channel capacity measured for a BER of 10^{-3}, is 2.0 and 1.3 dB respectively.

Similarly, to compare the performance of the GPS-LDPC codes over a flat Rayleigh fading channel, the GPS(672,3,336) code whose were compared in Fig. 3.7, against two Margulis based LDPC codes with the same (n, γ, k)

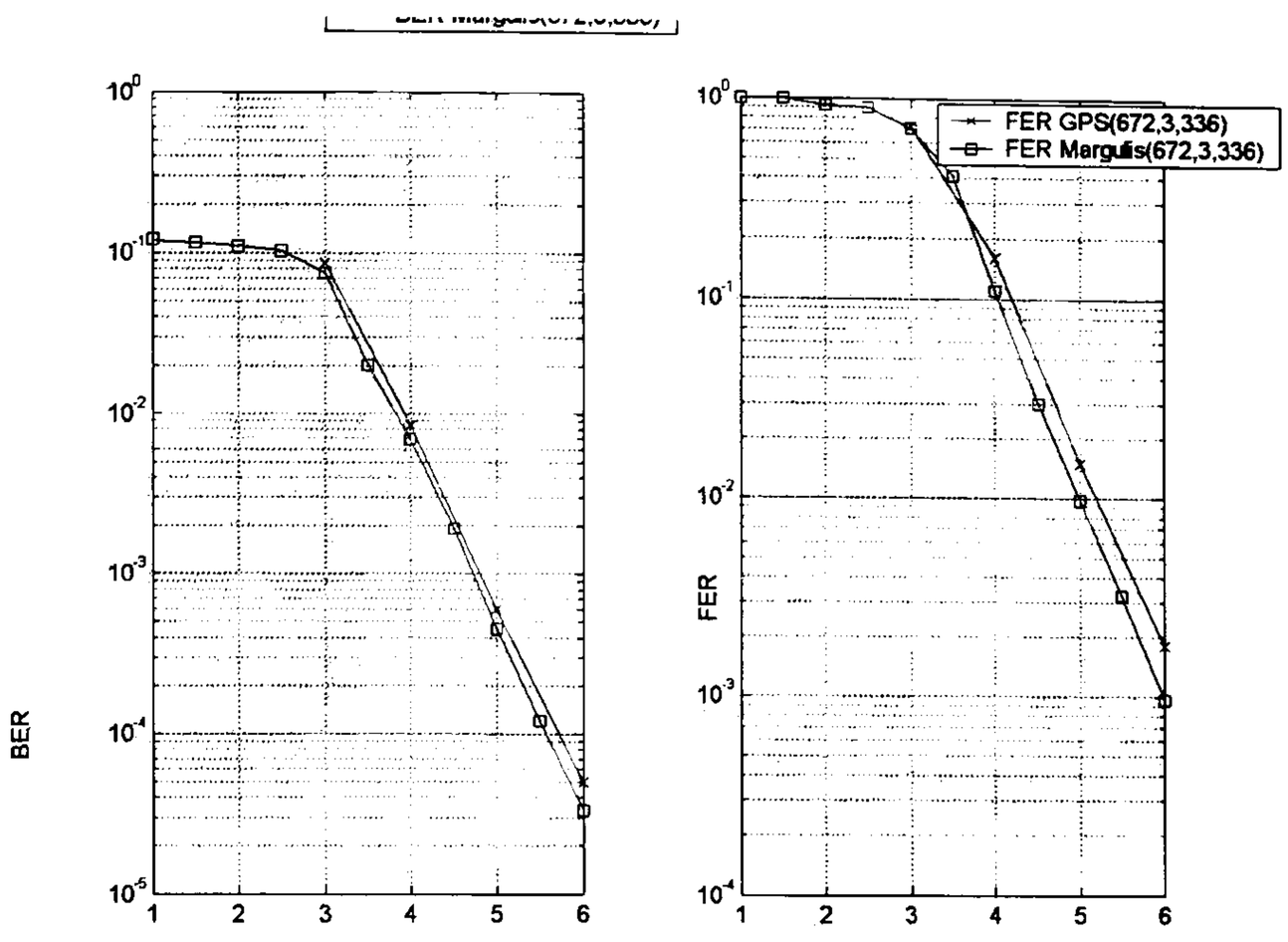

Figure 3.7: BER (left) and FER (right) performance of the GPS(672,3,336) and Margulis(672,3,336) LDPC codes. Maximum 60 iterations over the flat Rayleigh fading channel, and QPSK modulation

parameters; both constructions are half rate. For the simulation, the sum-product decoding algorithm was employed over an AWGN channel. This time, the maximum number of iterations was 60.

The GPS and the Margulis LDPC codes follow very similar BER and FER performance curves, although the Margulis ensemble outperforms the GPS ensemble by 0.1 to 0.2 dB for all E_b/N_0 values investigated of BER and FER. A comparison of the E_b/N_0 required by the GPS-LDPC codes, with length 672, to achieve a target BER of 10^{-3} when analised over the flat Rayleigh fading channel and over the AWGN channel, show a gap of 2.5 dB and 2.3 dB respectively.

Extrinsic information transfer (EXIT) charts [89] are analised in [10], with an approximation for the ensemble of LDPC codes with parameters

(n,3,k=n/2) with $\rho = 6$ and $\gamma = 3$. These are portrayed in the next section.

3.5 General construction of GPS LDPC Codes

The generalization of the procedure to construct regular GPS LDPC codes, includes as design parameters the proposed girth (g), the proposed row weight and the proposed column weight. The following rule should be considered prior to selection of the design parameters. For a GPS LDPC code with rate $1-(\rho/\gamma)$ to be realisable, the remainder of (γ·x, for x=(1,2,...,$g/2$), divided by ($g/2$), must be zero.

γ must be a positive integer with values $\gamma \geq 2$ and the girth, must be a positive integer with values $g \geq 6$. The girth must be $g \geq \gamma$. Once this variables have been assigned values, the cycle matrix, the shift matrix and the GPS-LDPC can be constructed.

The cycle matrix contains ρ+1 circulant identifiers of the type I_{D1}. If $(\rho/\gamma) \leq 2$, then the cycle matrix and the slope matrix contain g columns and $g/2$ rows. If $(\rho/\gamma) > 2$, then the cycle matrix and the slope matrix contain $(\rho/\gamma) \cdot (g/2)$ columns and $g/2$ rows.

If $(\rho/\gamma) \leq 2$, then the procedure to create the GPS LDPC matrix is the same as that one detailed previously for $\gamma = 2$ or $\gamma = 3$, except for the fact that the number of I_{D2} circulant identifiers will be increased on a per column basis according to the following rule; for the first $g/2$ columns, there will be a total of $\gamma - 2$ I_{D2} circulant identifiers, while for the rest of the columns, there will be a total of $\gamma - 1$ I_{D2} (e.g. eq. 3.10 and eq. 3.11 with $g = 12$ $\gamma = 3$ and $\rho = 4$, and $g = 16$, $\gamma = 4$ and $\rho = 5$, to construct GPS-LDPC codes with

rate $1-(3/4)=1/4=0.25$ and $1-(4/5)=1/5=0.20$ respectively).

$$\mathbf{Y}_{\mathbf{H_{qc}}qc}=\begin{bmatrix} I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D2} \\ I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D2} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} \\ I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D2} & I_{D1} & I_N & I_N & I_{D2} \\ I_{D2} & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D2} & I_{D1} & I_N & I_N \\ I_N & I_{D2} & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D2} & I_{D1} & I_N \\ I_N & I_N & I_{D2} & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} & I_{D2} & I_{D2} & I_{D1} \end{bmatrix} \tag{3.10}$$

$$\mathbf{Y}_{\mathbf{H_{qc}}qc}=\begin{bmatrix} I_{D1} & I_N & I_N & I_N & I_{D2} & I_{D2} & I_{D1} & I_{D1} & I_N \\ I_{D1} & I_{D1} & I_N & I_N & I_N & I_{D2} & I_{D2} & I_{D2} & I_N \\ I_{D2} & I_{D1} & I_N & I_N & I_N & I_N & I_{D2} & I_{D2} & I_N \\ I_{D2} & I_{D2} & I_{D1} & I_N & I_N & I_N & I_N & I_{D2} & I_N \\ I_N & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_N & I_N & I_{D1} \\ I_N & I_N & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_N & I_{D2} \\ I_N & I_N & I_{D2} & I_{D2} & I_{D1} & I_{D1} & I_N & I_N & I_{D2} \\ I_N & I_N & I_N & I_{D2} & I_{D2} & I_{D1} & I_{D1} & I_N & I_{D2} \end{bmatrix} \tag{3.11}$$

It can be noticed in eq. 3.11 that the number of columns has been reduced in such a way that the first $g/2$ columns from left to right remain the same, while the remaining $g/2$ columns have been punctured (i.e. columns 10, 11, 12, 14, 15, and 16 have been punctured) in such a way that the number of circulant identifiers per row is the same. This puncturing of columns

reduces the processing time to obtain all the invalid cycles within the cycle matrix. If this approach is considered, then the slope matrix must be punctuared as well, removing the same columns; the puncturing on this matrix has the effect of allowing a bigger number of valid shifts in the slope dababase, therefore producing GPS-LDPC matrices with shorter length for the same design parameters, compared to the results without punturing the previously mentioned columns.

As an example, consider $g = 12$, $\gamma = 3$ and $\rho = 4$ as design parameters; the rate for this code is $1 - (\rho/\gamma) = 1 - (3/4) = 0.25$. One solution for the slope matrix, considering a length $n = 2320$ ($k = 1740$) as shown in eq. 3.12.

$$\mathbf{S_H} = \begin{bmatrix} 0 & N & N & N & 3 & 0 & 0 & N \\ 0 & 0 & N & N & N & 5 & 13 & N \\ 1 & 0 & 0 & N & N & N & 45 & N \\ N & 3 & 0 & 0 & N & N & N & 0 \\ N & N & 5 & 0 & 0 & N & N & 13 \\ N & N & N & 1 & 0 & 0 & N & 105 \end{bmatrix} \tag{3.12}$$

If $(\rho/\gamma) > 2$, then the procedure to create the GPS LDPC matrix is the same as that one detailed previously for $\gamma = 2$ or $\gamma = 3$, except for the fact that the last $g/2$ columns of the cycle matrix and slope matrix, are repeated $(\rho/\gamma) - 2$ times.

As an example, consider $g = 8$, $\gamma = 3$ and $\rho = 9$ as the design parameters. The rate for such code is $1 - (3/9) = 2/3 = 0.67$. The cycle matrix for such design parameters is shown in eq. 3.13.

$$\mathbf{Y}_{\mathbf{H_{qc}}_{qc}} = \begin{bmatrix} I_{D1} & I_N & I_{D2} & I_{D1} & I_{D1} & I_N & I_{D2} & I_{D2} & I_{D1} & I_N & I_{D2} & I_{D2} \\ I_{D1} & I_{D1} & I_N & I_{D2} & I_{D2} & I_{D1} & I_N & I_{D2} & I_{D2} & I_{D1} & I_N & I_{D2} \\ I_{D2} & I_{D1} & I_{D1} & I_N & I_{D2} & I_{D2} & I_{D1} & I_N & I_{D2} & I_{D2} & I_{D1} & I_N \\ I_N & I_{D2} & I_{D1} & I_{D1} & I_N & I_{D2} & I_{D2} & I_{D1} & I_N & I_{D2} & I_{D2} & I_{D1} \end{bmatrix} \tag{3.13}$$

The complexity required to decode the GPS LDPC codes, is slightly higher compared to an LDPC code that contains only cyclic or quasi-cyclic shifts. This is because the GPS LDPC codes contains two types of matrices, (a) an identity matrix with cyclic shifts depending on the design, and (b) the null matrix, which in general adds little or no complexity to the decoding process.

Based on [85], the decoding procedure of a GPS-LDPC code can be divided into υ parallel decoding processes, which is the total number of slopes S elements in the shift matrix (i.e. the number of null N elements in the same shift matrix does not affect the complexity, since they represent null matrices in the LDPC matrix). Therefore, the delay to decode a GPS LDPC code is υ times faster compared to an LDPC code without decoding in parallel.

3.6 Flexibility of the Design Parameters of the GPS-LDPC codes

If due to design requirements, there is a need to increase the girth of the code for a particular length, or simply to increase the girth beyond 16 to make

better use of length of a longer code (i.e. the longer the length of a code, the higher the value of the girth that can be achieved, resulting in better BER and PER performance), it is only necessary to change some I_{D1} cycle identifiers for I_U cycle identifiers. This would invariably increase the number of cycles to be calculated, and also would increase the delay to analyse each temporary shift value assigned to the slopes in the slope matrix.

For this construction algorithm, the longer the length of a code, the higher the value of the girth that can be achieved, resulting in better BER and PER performance. Although there is a maximum limit in the value of the girth achievable (i.e. $g = 16$) for the new algorithm, it is only necessary to change some I_{D1} cycle identifiers for I_U cycle identifiers. The number and location of the I_{D1} cycle identifiers that need to be changed for I_U cycle identifiers depends on each particular case, and therefore no guidelines are provided in this body of work. The changes previously mentioned would invariably increase the number of cycles to be calculated, and also would increase the delay to analyse each temporary shift value assigned to the slopes in the slope matrix.

The BER performance curves displayed in Fig. 3.8 are useful to understand the behaviour of GPS-LDPC codes when their parameters are modified. For instance, the GPS (2100,3,1050) LDPC code with $g = 12$, and the GPS (2100,3,1050) LDPC code with $g = 10$, show a gap of 0.75 dB measured for a BER of 10^{-3}, which increases proportionaly with the E_b/N_0. Similar behaviour is diplayed between the GPS (1000,3,500) LDPC code with $g = 10$, and the GPS (1000,3,500) LDPC code with $g = 8$; the gap for this case if 0.75 dB measured for a BER of 10^{-2}. Finally, it is interesting to note how the

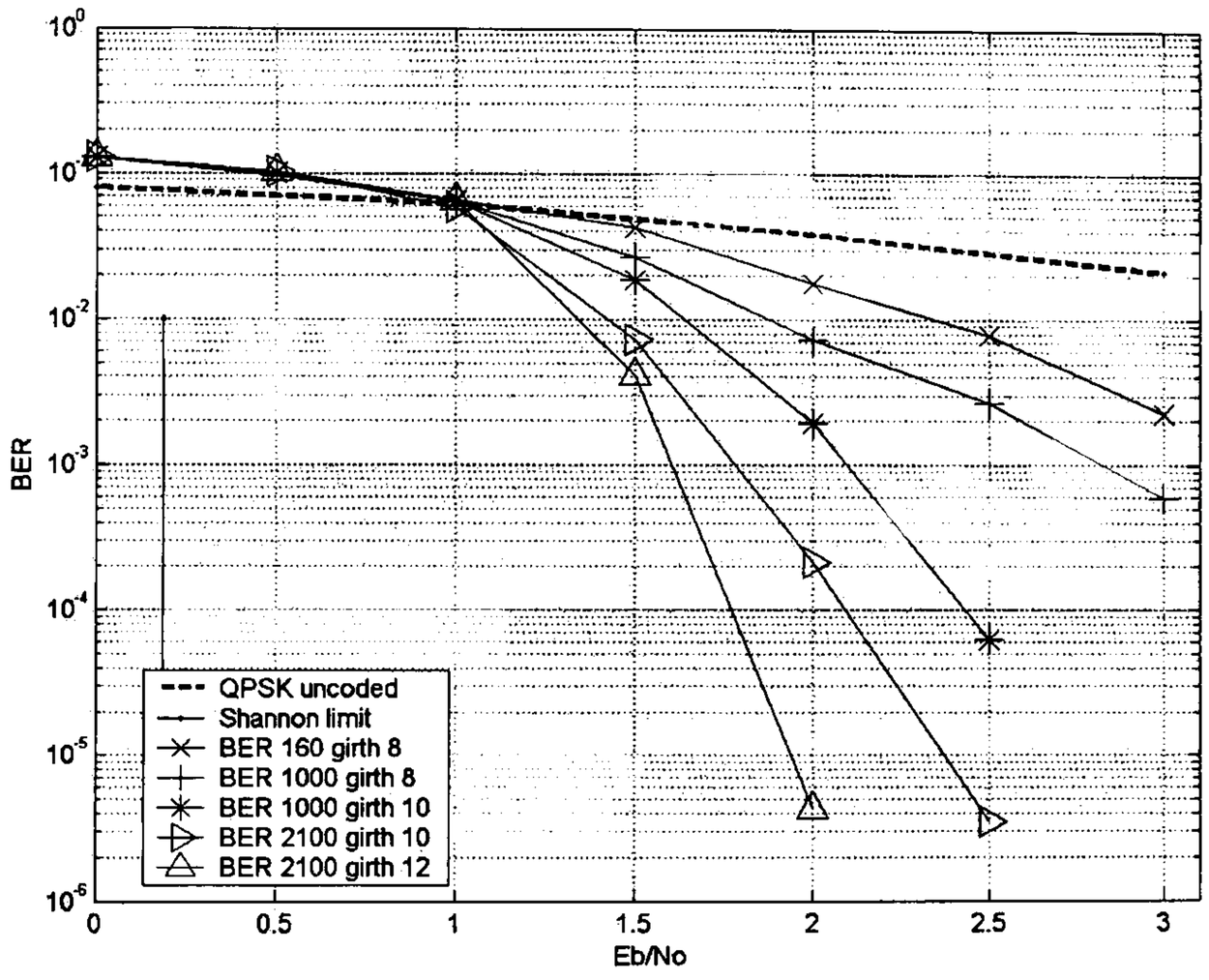

Figure 3.8: BER performance for the GPS (2100,3,1050) $g = 12$, (2100,3,1050) $g = 10$, (1000,3,500) $g = 10$, (1000,3,500) $g = 8$, and (160,3,80) $g = 8$, LPDC codes over the AWGN channel using QPSK modulation

BER performance curve of GPS-LDPC codes with different length, but the same girth as parameter, present the same slope. That is the case when the GPS (2100,3,1050) LDPC code with $g = 10$, is compared against the GPS (1000,3,500) LDPC code with $g = 10$; the same behaviour takes place when the GPS (1000,3,500) LDPC code with $g = 8$, and GPS (160,3,80) LDPC code with $g = 8$ are compared.

From the previous example, it can be concluded that the girth, and the length are linked; altering one of them will invariably affect the BER and FER performance; however, the cause of this change is particular to each parameter. Reducing the girth on one out of two codes with the same length, reduces the independence of the likelihoods after a certain number of iterations, which produces a change in the slope. On the other hand, increasing the length on one out of two codes with the same girth, increases the number of parity-check and variable nodes, improving the BER and FER performance, but since the independence of the likelihoods remains for similar number of iterations, the slope is not altered.

Also, by changing the distribution of the I_U it is possible to get GPS-LDPC codes with different values of γ. In other words, the variable and parity-check node distributions could be changed according to some design parameters to search for QC, which would vary the E_b/N_0 region where the code performs the best.

3.7 Summary

A novel algorithm to construct LDPC codes using the girth and the length of the code as design parameters, has been introduced. The codes construted with this procedure are half rate QC-LDPC codes with variable g and γ. The algorithm works in such a way that the number of operations required to calculate new LDPC codes with the same g and γ is reduced, when compared to similar QC-LPDC code constructions [18], [17], and [45]. The new GPS LDPC codes have similar performance to other random and structured LDPC codes over the AWGN channel and the flat Rayleigh fading channel, although only short code lengths were analised through BER and FER performance curves, with $n = 4368$ being the longest code. A major benefit of the new algorithm is its capacity to generate codes for a wide range of lengths, that proves impossible to achieve with other constructions, in particular, those based on well structured procedures, such as finite geometries. Although several sources suggest the requirement for LDPC matrices to contain large girth is not necessary to achieve good performance (i.e. only avoid girth 4), and that an increase in the girth may modify other characteristics of the parity-check matrix, therefore resulting in a BER performance improvement, here it has been shown that consistently, when the girth has been increased, the BER performance has improved as well; this implies that the improvement in the performance is also a function of the girth, besides other many characteristics.

An additional benefit of this algorithm is its flexibility to incorporate design constraints, to produce a wide variety of QC-LDPC codes, using more

efficiently the hardware of the channel encoder on a given communications system. For instance, a system that requires a very small delay on the channel encoding stage, would prefer to perform this operation in parallel through i shift registers directly related to i sections of the generator polynomial, or the generator matrix respectively, as proposed in [90]. This encoding is efficient because the number of sections is directly related to the number of circulants. Therefore, by maximizing the girth as a parameter for the GPS LDPC codes, the number of circulants is also maximized, with the additional benefit of improved performance due to the maximization of the girth in terms of the BER achieved for a particular E_b/N_0, and in the slope of the BER curve.

Assuming the same efficient encoding algorithm applied on a duplex or full-duplex system, with constraints on the number of shift registers to perform the channel encoding on one of the terminals, as well as requiring the same encoding rate and the same codeword length on both sides, the new algorithm has the flexibility to produce QC-LDPC codes with different girths, but with the same rate and and length.

The characterisation in dept of this ensemble my means of DE [8], provides the channel capacity achievable for an LDPC code with parameters (n,j,k=n/2) with $\rho = 6$ and $\gamma = 3$. However, LDPC codes with short lengths, which cannot achieve the same BER and FER performance as their long peers (i.e. their bit-error probability is relatively far from the channel capacity) are more desirable for systems with tight margin on the tolerated delay for channel enconding. To fulfil the criteria previously mentioned, providing acceptable BER and FER, after a few number of iterations, the parallel

concatenation of codes with different γ has been proposed in [13], [14], [91], and [92] among others; it is interesting to notice that the EXIT chart for LDPC codes with $\gamma = 2$ and $\rho = 4$, shows that such ensemble is desirable as the inner code working when concatenated in parallel, due to big improvement on the extrinsic information provided after the first iteration is accomplished by this code. Based on these ideas, it is desirable to analise and characterise the performance of the GPS LDPC codes introduced here, when concatenated in parallel.

Chapter 4

Parallel Concatenated Structured and Random LDPC codes

4.1 Purpose

The novelty in this work is based on (1) the analysis of the performance of well structured LDPC codes when concatenated in parallel, in terms of BER and FER, (2) the comparison of such performance against their random counterparts, and (3) the establishment of guidelines, if any, to secure the successful performance of this approach.

After the powerful technique proposed in [93] to construct long powerful codes from short component codes with different characteristics (i.e. coding and decoding complexity, and BER and FER performance), known as concatenated coding, turbo codes [2] make use of two fundamental ideas to

achieve outstanding performance: an encoder design that produces a code with random like properties (i.e. recursive encoders together with large pseudorandom interleavers), and a decoder design that uses an iterative decoder that exchanges SISO values.

These two fundamental ideas present in turbo coding, are considered in [13] and [14] where the concatenation of two random LDPC codes with different Mean Column Weight (MCW) is introduced. Such a scheme has proved to be an effective way to improve the bit error rate (BER) performance by increasing the redundancy whilst reducing the decoding complexity when compared to an LDPC of the same length. The use of the sum-product decoding algorithm through the joint bipartite graph shows a better performance than that of the individual codes.

Other benefits are obtained from this approach when compared to an LDPC code with the same length, such as, less decoding complexity and less delay if the decoder works in parallel, as well as less memory required since the number of elements in the concatenated LDPC matrices is smaller than that of a single LDPC matrix for the same code length. Also, no interleaver is required between the constituent encoders and decoders.

The idea behind using LDPC codes with different MCW, is to take advantage of their individual performance at different E_b/N_0 values. It is well known that low MCW LDPC codes (i.e. $2 \leq MCW \leq 2.5$) perform better in the low to medium E_b/N_0 region, while high MCW LDPC codes (i.e. $MCW \geq 2.5$) perform better in the medium to high E_b/N_0 region, for each particular code length.

It is therefore relevant to analyse the performance of other types of LDPC

codes that can be concatenated in parallel, and at the same time, can provide further improvement either in performance or reduced complexity. The parallel concatenation of half-rate LDPC codes producing codewords that are rate 1/3, is particularly desirable, since some communication systems use these coding rates as part of their standard, and especially after the introduction of adaptive hybrid Forward Error Correction (FER) / Automatic Repeat reQuest (ARQ) protocols. Three new scenarios are proposed utilising structured LDPC codes to take advantage of their efficient encoding and decoding properties, as previously stated.

The first one analyses two PG, or two EG LDPC codes. As described in [43], an EG or PG LDPC matrix is built by cyclic shifts of a circulant; this matrix defines an EG or PG LDPC code. The transpose of the EG or PG LDPC matrix defines a different EG or PG LDPC code. Therefore, by using a single matrix, two codes are defined.

The second one analyses two types of structured LDPC codes concatenated in parallel, where the first one has $\gamma = 2$ and is described in [16], while the second one has $\gamma = 3$ and is described in [88]. Both LDPC codes are regular half rate LDPC codes with good BER and FER performance.

The third one analyses two GPS LDPC codes as described in [94], where the first one has $\gamma = 2$ while the second one has $\gamma = 3$. Both LDPC codes are regular half rate LDPC codes with good BER and FER performance. Also, these codes are quasi-cyclic (QC) LDPC codes and therefore have low encoding and decoding complexity which make them an atractive option for parallel concatenation.

Extrinsic information transfer (EXIT) charts are included to further anal-

yse the performance of the LDPC codes considered. I_A represents the average mutual information between the bits on the decoder graph edges (the bits about which extrinsic L-values are passed) and the a priori L-values; I_E represents the average mutual information between the bits on the graph edges and the extrinsic L-values. VND and CND are the Variable to Node of Degree d_v ($d_v = \gamma$) and the Check to Node of Degree d_c ($d_v = \rho$) messages, respectively. To create the EXIT charts of LDPC codes, in [95], each LDPC code is seen as the concatenation of two block codes connected through an interleaver. The same approach is considered here to produce the EXIT charts of the LDPC codes used for concatenation.

Chapters 2 and 4 include a detailed description of the procedure to create these LDPC codes. Particular characteristics of the LDPC codes chosen for simulations were included in this chapter.

4.2 Parallel Concatenated Encoding

Two generator polynomials or generator matrices, gp_1 and gp_2, encode the information sequence i with k bits $b_{1,..,k}$, to produce independently the codewords c_1 and c_2 of length n_1 and n_2 respectively, where $n_1 = n_2$, and both codes have the same rate r. There is no need for an interleaver between the constituent generator polynomials, because the distribution of the parity checks that link the codeword bits in the LDPC matrices, is pseudo-random. Both generator polynomials produce codewords in systematic form, with k information bits and $n-k$ parity bits. Only the parity bits p are sent together with the input information sequence i. The diagram of the concatenated en-

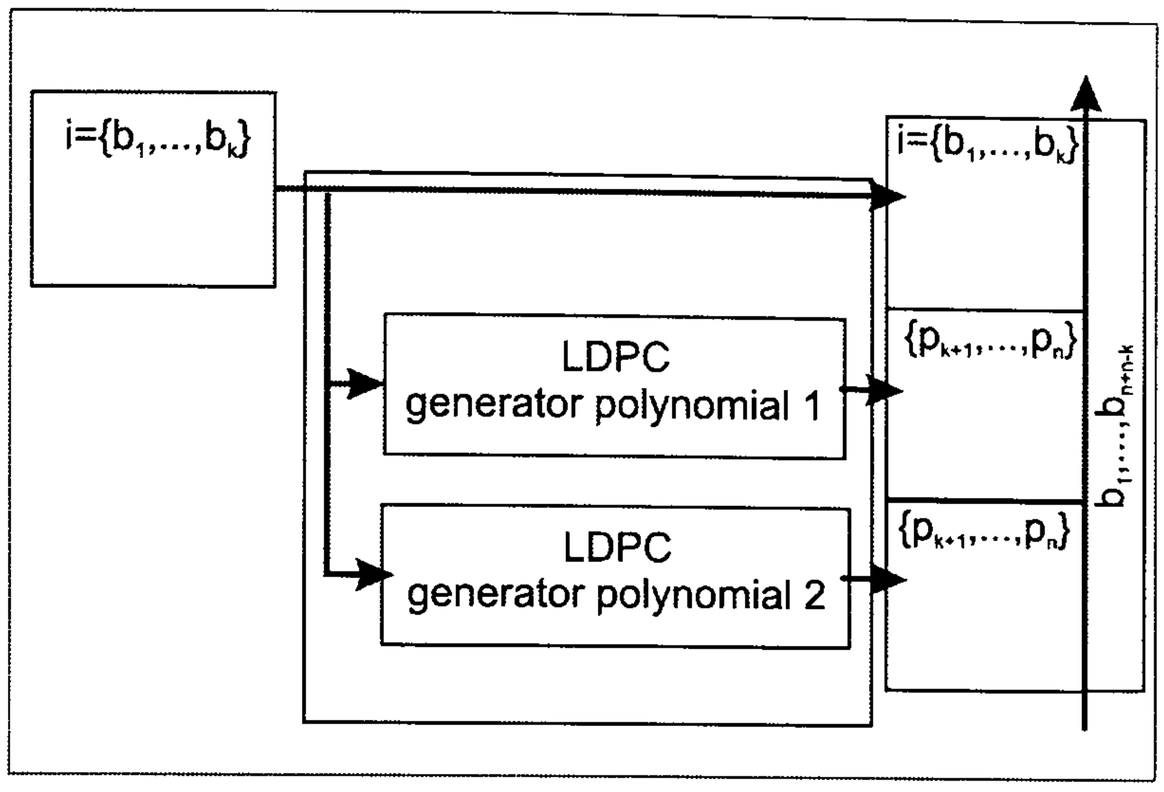

Figure 4.1: Block diagram of the encoder for the parallel concatenation of LDPC codes

coder is shown in Fig. 4.1.

4.3 Parallel Concatenated Decoding

The decoding is performed in parallel on each constituent decoder independently. The log-likelihood values of the received signal are fed into both decoders and after completion of the individual iterations, the a-posteriori log-likelihoods L_p are exchanged between decoders. The addition of the received log-likelihoods and the a-priori log-likelihoods L_a, is substracted from the a-posteriori log-likelihoods L_p, to obtain the extrinsic information in the form of log-likelihoods L_e, as shown in Fig. 4.3. This would complete one super iteration. During the first super iteration, the a-posteriori log-likelihoods are computed with no a-priori information.

The process of exchanging information between the constituent decoders continues until one of the component decoders converge to valid codewords, or a maximum number of super iterations is reached. In the latter case,

if different LDPC codes are being used, the output from the decoder with the best individual BER performance should be used to perform the hard decision, as it is considered the best estimate of the transmitted information sequence.

LDPC codes can be represented by means of bipartite graphs (Tanner graphs). The concatenation of two LDPC codes can also be represented through a bipartite graph, in such a way that the variable nodes corresponding to the information and parity bits in both codes, are related to the check nodes in both LDPC [71].

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{4.1}$$

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

The bipartite graph shown in Fig. 4.2 corresponds to the concatenation of the LDPC matrices described in eq. 4.1 and eq. 4.2. Not all the edges

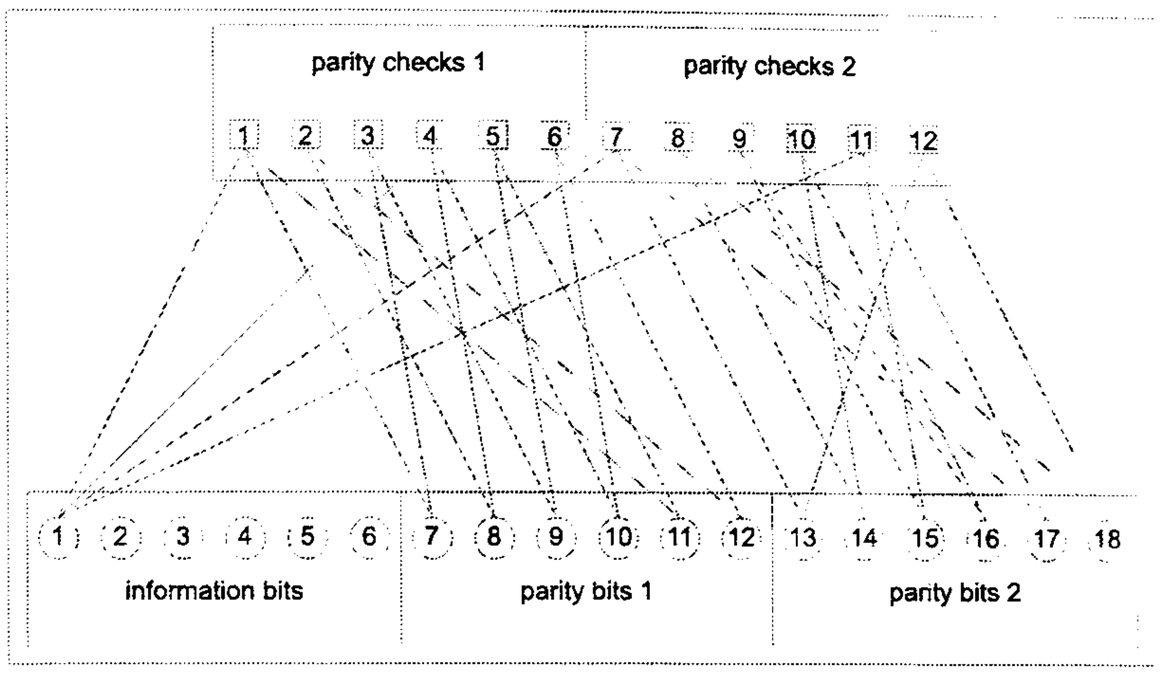

Figure 4.2: Bipartite graph corresponding to the parallel concatenation of the LDPC code matrices 4.1 and 4.2

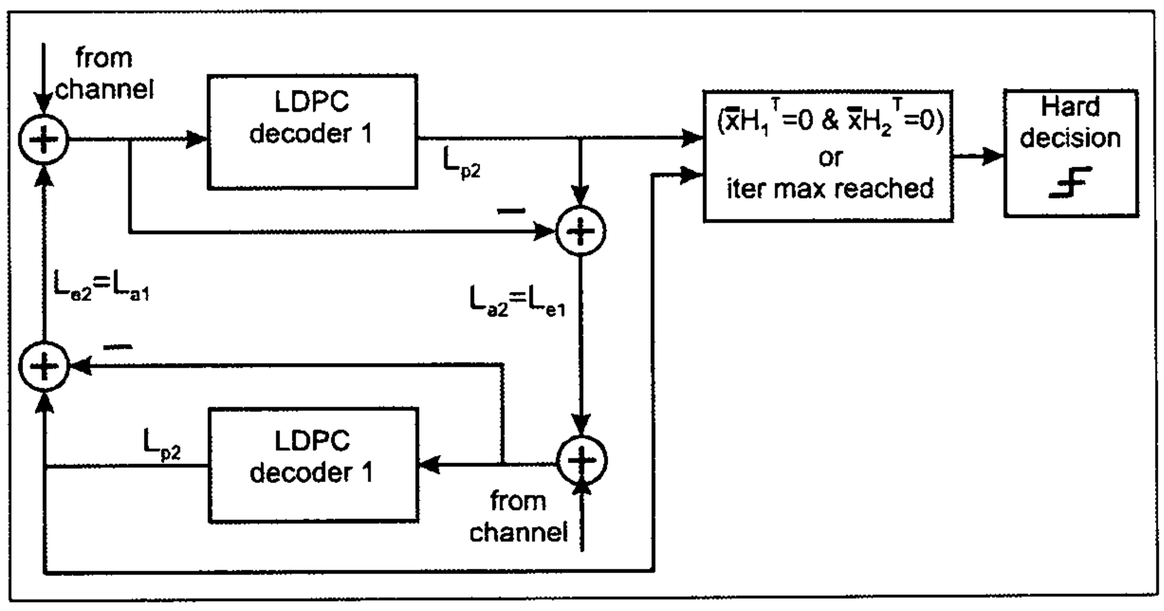

Figure 4.3: Block diagram for the decoding of parallel concated LDPC codes

joining variable nodes to check nodes have been depicted, in order to allow clarity.

4.4 Parallel Concatenated EG and PG LDPC Codes

Here, the parallel concatenation of EG and PG LDPC codes is proposed. EG and PG LDPC codes [43] are completely defined by the parity check matrix **H**. Chapter 2 contains the list of the vectors representing the circulants **E**

that generate the LDPC matrices $\mathbf{H}$, as well as the corresponding generator polynomials $\mathbf{GP}$ of $\mathbf{H}$ and $\mathbf{H^T}$, for the EG and PG LDPC codes, analysed under parallel contenation.

Fig. 4.4, and Fig. 4.5 show the EXIT chart for the type I PG(73,9,45) LDPC code. Fig. 4.6, and Fig. 4.7 show the EXIT chart for the type I PG(273,17,191) LDPC code. Fig. 4.4 shows that the first LDPC code starts to decode successfully at a lower E_b/N_0 , when compared to the second one in Fig. 4.6. Graphs Fig. 4.5 and Fig. 4.7 show that the proposed concatenation in parallel of PG and EG LDPC codes, increases very fast the mutual information (i.e. after a relatively small number of iterations), once the E_b/N_0 for each case, is high enough to show the BER turbo cliff (the turbo cliff is measured in [53] at a BER of 10^{-4}). Also, it can be seen that PG(2,2^s) and EG(2,2^s) LDPC codes with $s > 2$, have a *good start* after the first half of the first iteration, since the mutual information increase is related to γ for the I_{VND} and to ρ for the I_{CND} (i.e. as ρ and γ increase, the mutual information increases).

4.4.1 Simulation results

Both EG and PG LDPC codes share almost identical performance characteristics, and therefore the simulations include only PG LDPC codes. In [43], BER and FER performance curves of these codes show no gap between EG and PG for a similar length, after being decoded with the sum-product algorithm. Therefore, only PG constructions were considered for the parallel concatenation.

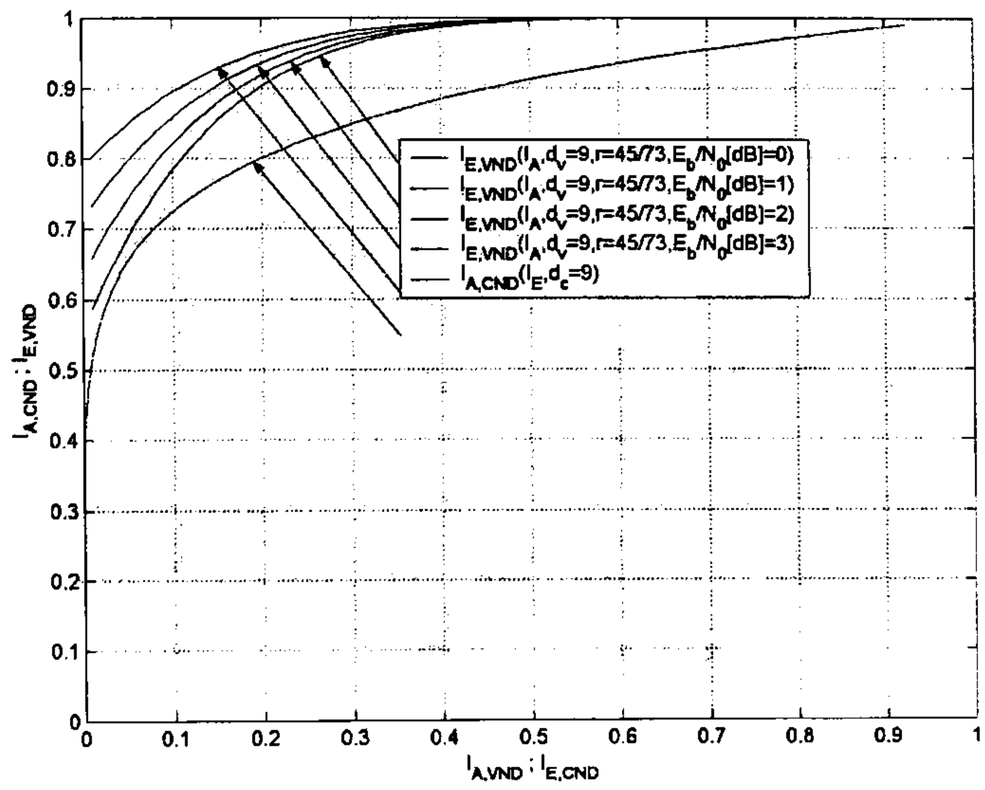

Figure 4.4: EXIT chart for the type I PG(73,9,45) LDPC code with E_b/N_0=0,1,2,3 dB.

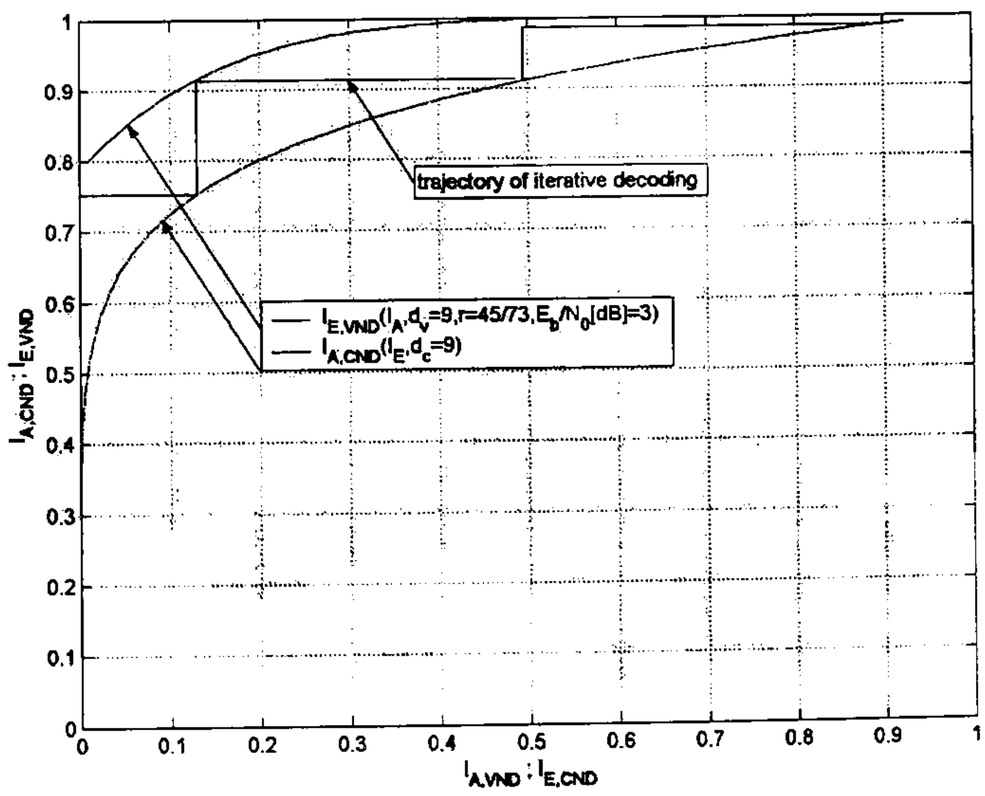

Figure 4.5: EXIT chart for the type I PG(73,9,45) LDPC code with E_b/N_0=3 dB.

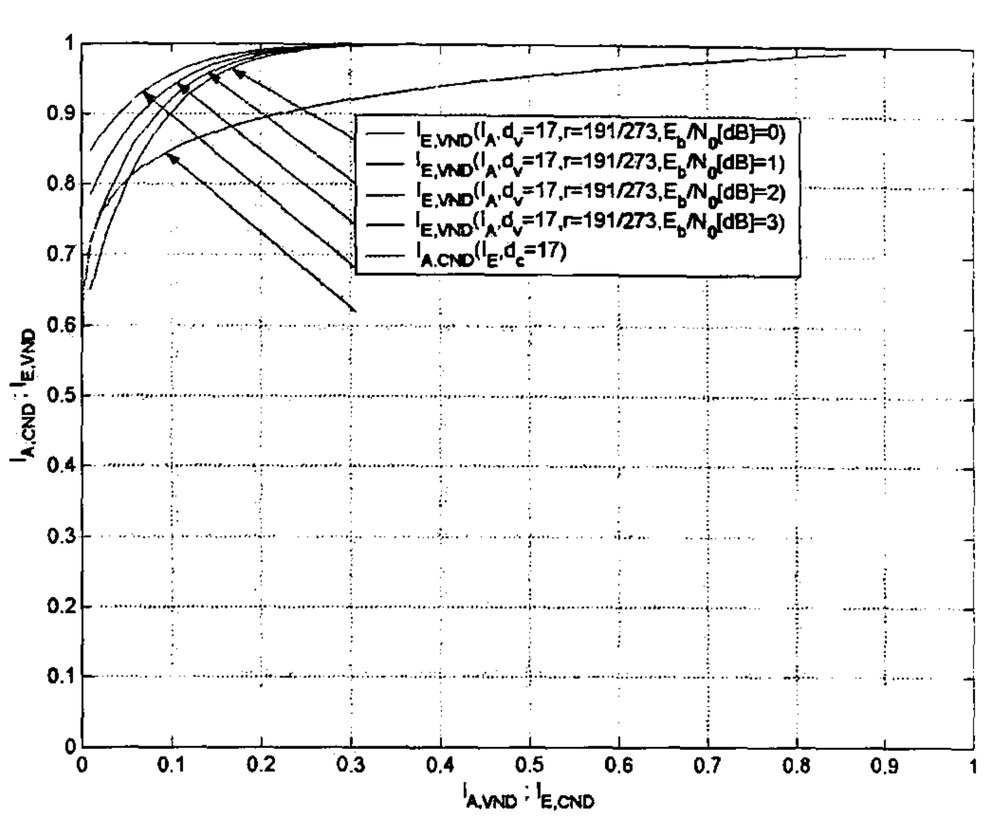

Figure 4.6: EXIT chart for the type I PG(273,17,191) LDPC code with E_b/N_0=0,1,2,3 dB.

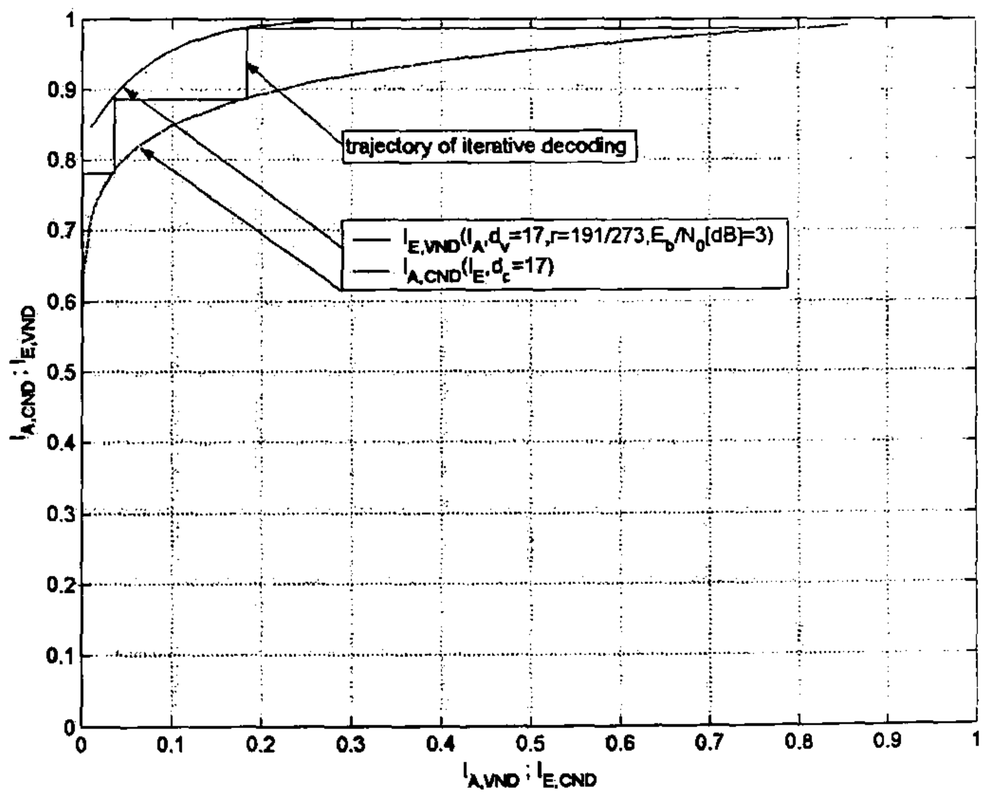

Figure 4.7: EXIT chart for the type I PG(273,17,191) LDPC code with E_b/N_0=3 dB.

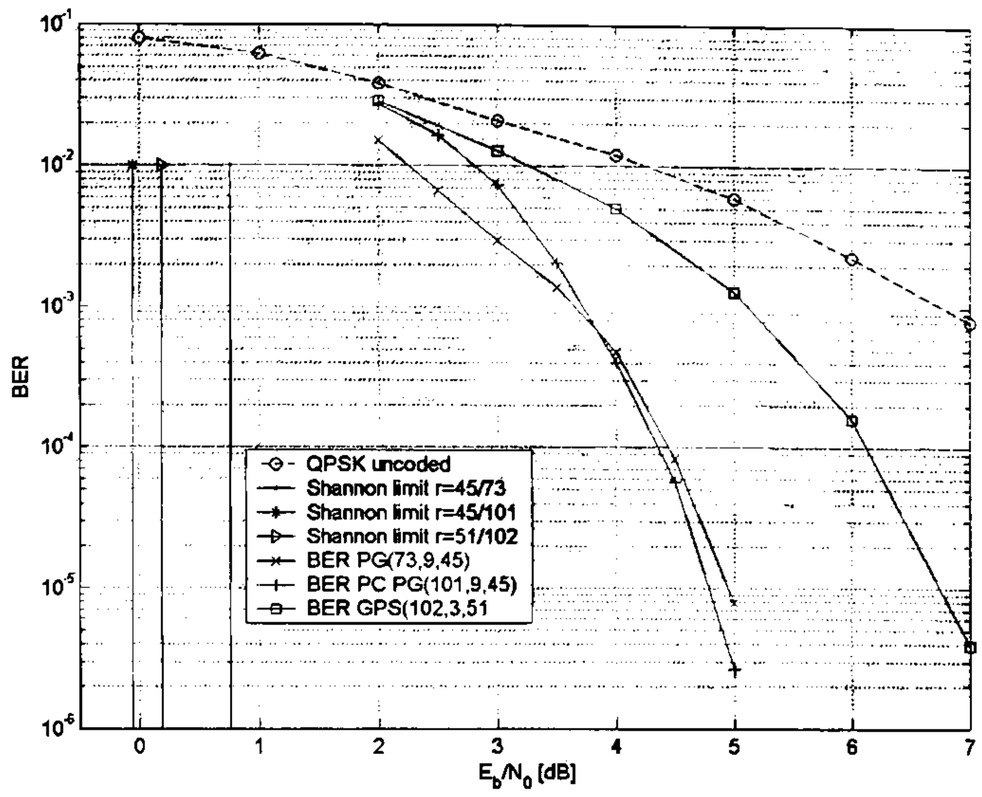

Figure 4.8: BER performance comparison between the Parallel Concatenated type I PG(101,9,45) and the GPS(102,3,51) LDPC codes over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

The codes considered were the type I PG(73,9,45) and PG(273,17,191) LDPC codes, which generate the parallel concatenated type I PG(101,9,45) and PG(355,17,191) LDPC codes. The simulations were performed over an AWGN channel with sum-product decoding. The maximum number of iterations was set at 30.

Fig. 4.8 shows how the proposed concatenation in parallel of EG and PG LDPC codes improves the BER compared to the single component after an E_b/N_0=3.8 dB has been reached.

Fig. 4.9 shows how the proposed concatenation in parallel of PG and EG LDPC codes achieves a better FER than an LDPC code with similar length and rate.

Fig. 4.10 shows how the proposed concatenation in parallel of PG and EG LDPC codes improves the BER compared to the single component after an E_b/N_0=4.2 dB has been reached.

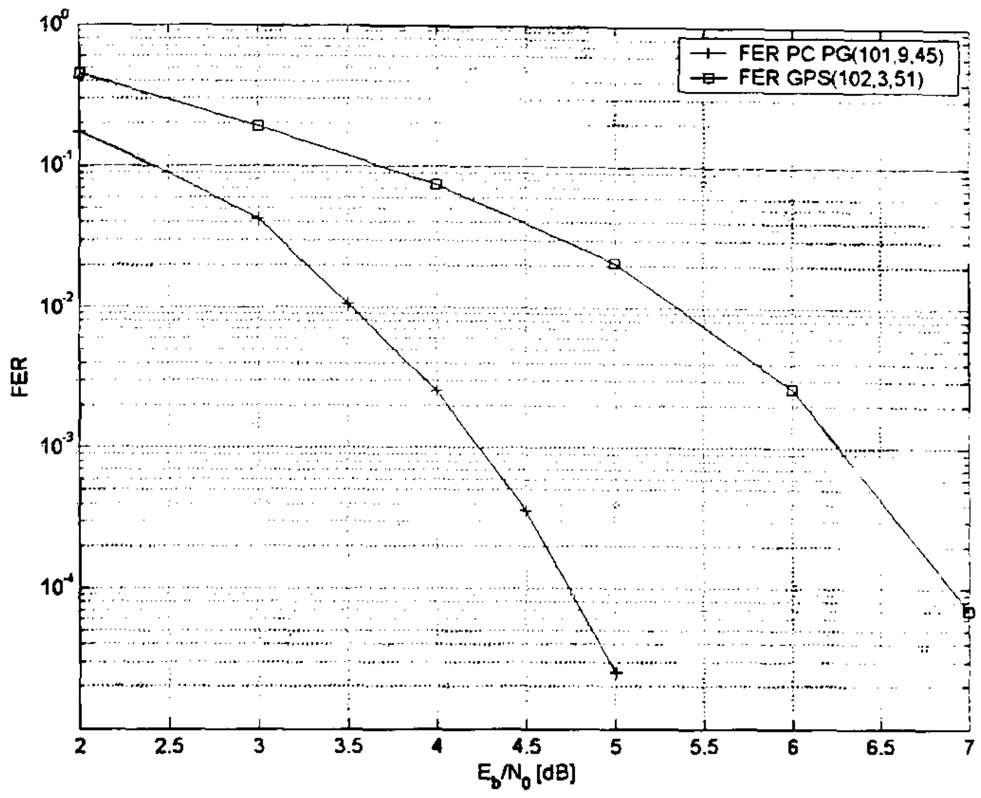

Figure 4.9: FER performance comparison between the Parallel Concatenated type I PG(101,9,45) and the GPS(102,3,51) LDPC codes over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

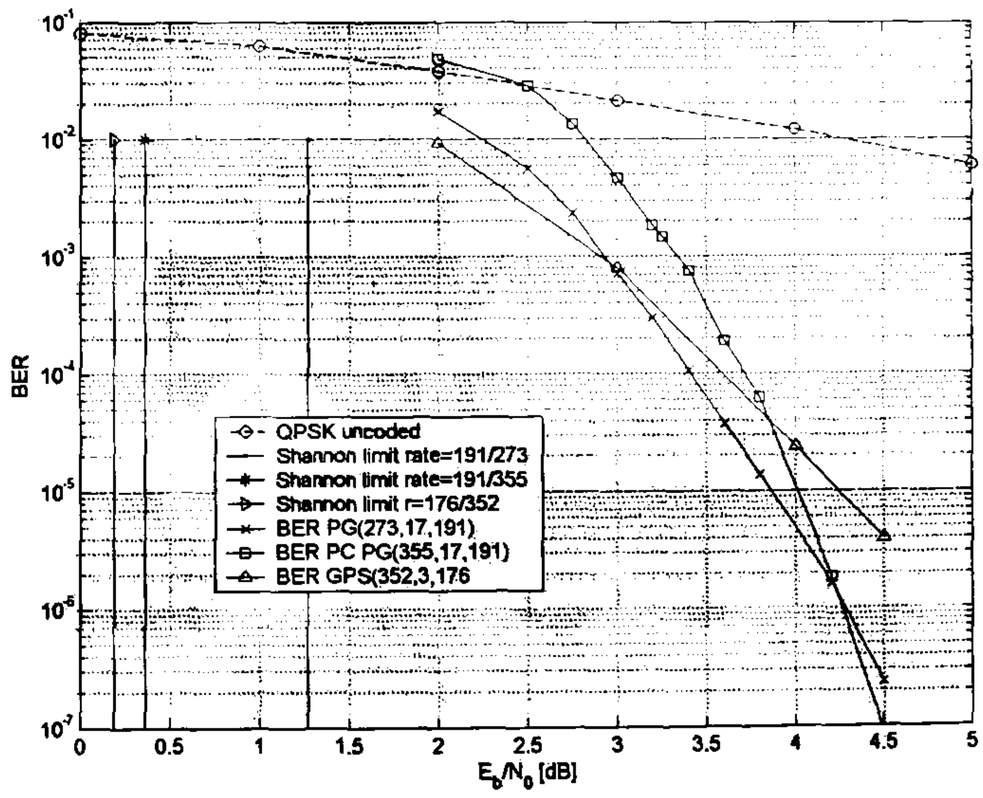

Figure 4.10: BER performance comparison between the Parallel Concatenated type I PG(355,17,191) and the GPS(352,3,176) LDPC codes over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

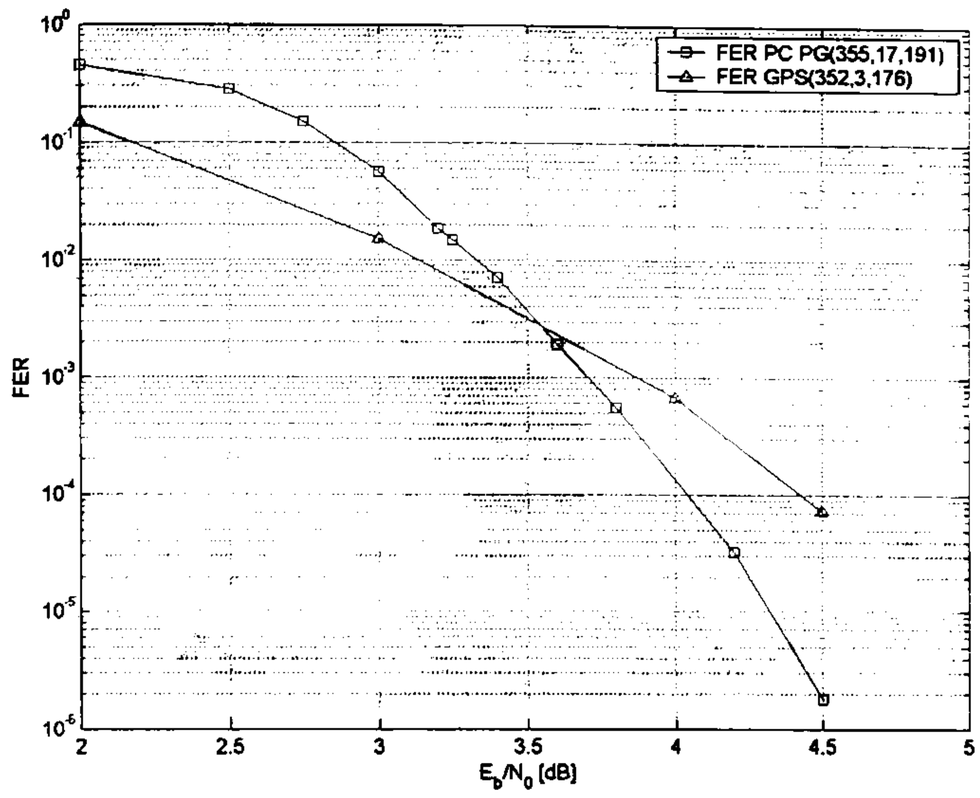

Figure 4.11: FER performance comparison between the Parallel Concatenated type I PG(355,17,191) and the GPS(352,3,176) LDPC codes over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

Fig. 4.11 shows how the proposed parallel concatenation of PG and EG LDPC codes achieves a better FER than an LDPC code with similar length and rate.

When comparing Fig. 4.8 against Fig. 4.10 it is clear that the EG and PG LDPC codes proposed for parallel concatenation do achieve a better BER performance than their constituent parts, but only after a certain E_b/N_0 has been reached. Although this is a limited performance improvement, this approach could be included in a system with Automatic Repeat Request (ARQ) if the E_b/N_0 is known by the transmitter. Rate adaptation with hybrid ARQ [96], [97], [98] is considered an important improvent for recent communication standards such as the Universal Mobile Telecommunications Service (UMTS) employing High Speed Downlink Packet Access (HSDPA) [99]. If the transmitter has knowledge of the quality of the channel (i.e.

E_b/N_0), it could decide if a probability of decoding correctly a retransmission is higher by sending the same frame with single codification, or by sending the complementary redundancy from the parallel encoder.

Other benefit comes from the FER performance graphs, since this approach forces the validation of the codewords from two different codeword sets, reducing the likelihood of approving a valid codeword, that is different from the transmitted one. Such task could be accomplished with the same LDPC matrix in the case of EG and PG LDPC codes. Through simulations under the AWGN channel, with the sum-product algorithm applied on a single LDPC decoder (i.e. no parallel concatenation), it was revealed how the FER performance decreases in the case of short code lengths of EG and PG LDPC codes due to the short Hamming distance (i.e. EG n =15 with $d_{min} = 5$, n =63 with $d_{min} = 9$, and n =255 with $d_{min} = 17$, and PG n =21 with $d_{min} = 6$, n =73 with $d_{min} = 10$, and n =273 with $d_{min} = 18$; no frame errors due to wrong codeword validation were detected for longer EG or PG LDPC codes for the E_b/N_0 values analysed); such effect is not small enough to be discarded as a contributor to the poor FER performance.

Since only one LDPC matrix is required, defined through a single vector, a reduced amount of memory to contain the LDPC matrix is required, when compared to other codes concatenated in parallel, such as [14], [13], and [15]. Based on the publication of efficient encoding [90] and efficient decoding [86], [87] techniques for Cyclic and QC LDPC codes, the application of EG and PG LDPC codes on parallel concatenated schemes, as proposed in this section, becomes feasible, although such techniques are not included in this study.

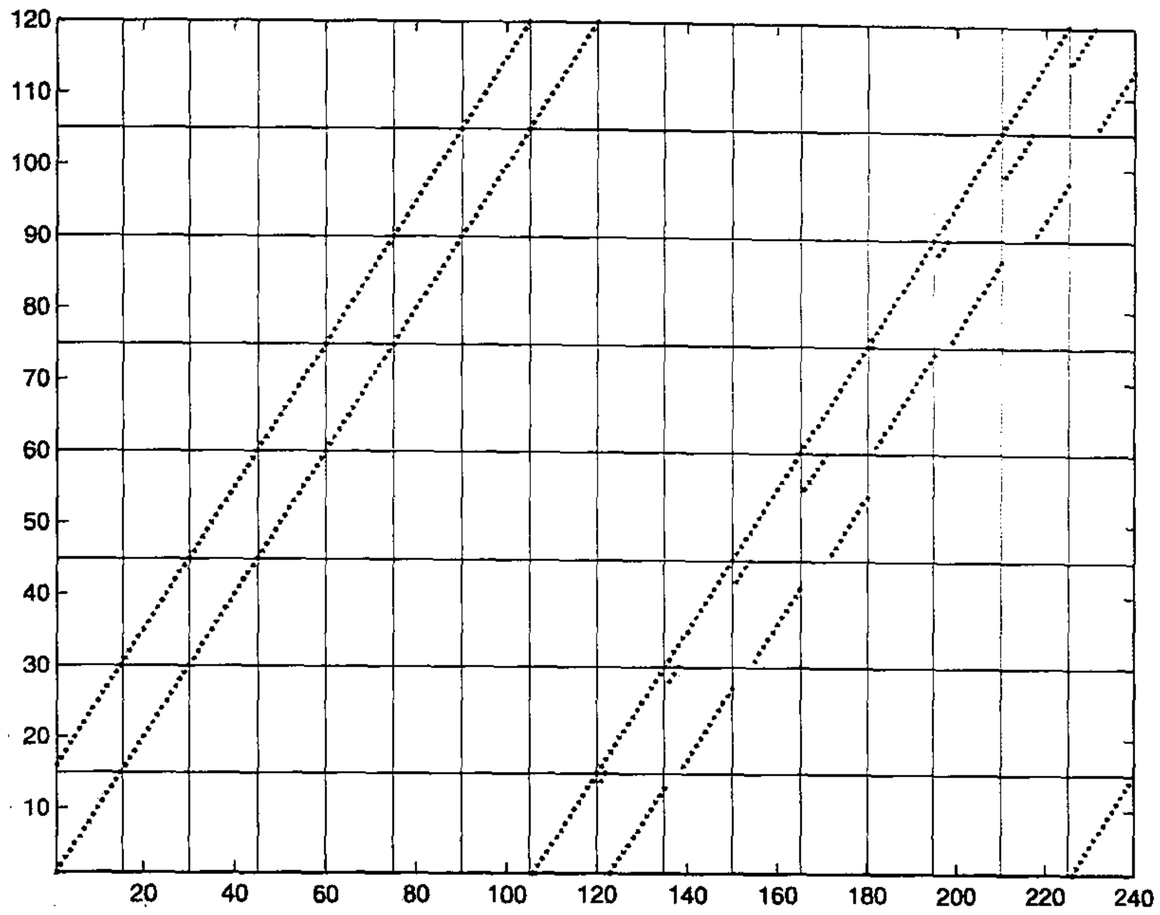

Figure 4.12: GM(240,2,120) LDPC code

4.5 Parallel Concatenated Graphical Model with $\gamma = 2$ and Margulis with $\gamma = 3$ LDPC Codes

In [16] and [17], a procedure to construct regular LDPC codes by means of graphical models with $\gamma = 2$ is presented. The construction algorithm is explained in chapter 2. Fig. 4.12 and Fig. 4.13 depict the GM(240,2,120) and GM(672,2,336) LDPC matrices respectively.

The procedure to construct regular LDPC codes by means of Cayley graphs is proposed in [88]. Such LDPC codes are half rate, and have $g = 6$, $\rho = 3$, $\gamma = 6$, and length $n = 2(q^3 - q)$ with q any prime number, resulting in the lengths shown in Table 4.1, for different values of q.

- $q = 3$ $n = 48$

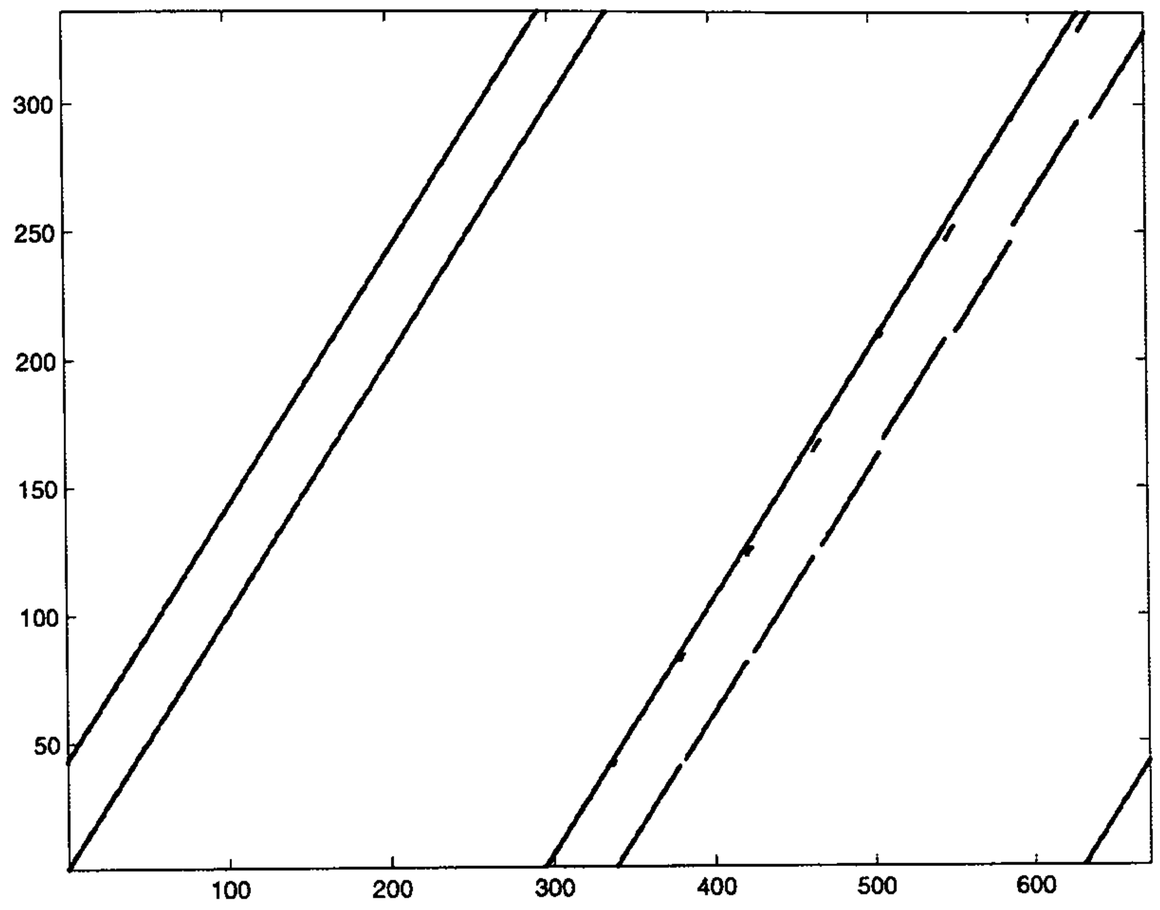

Figure 4.13: GM(672,2,336) LDPC code

q	n	k	γ	ρ	$rate$
3	48	24	3	6	0.5
5	240	120	3	6	0.5
7	672	336	3	6	0.5
11	2640	1320	3	6	0.5
13	4368	2184	3	6	0.5
17	9792	4896	3	6	0.5
19	13680	6840	3	6	0.5

Table 4.1: Basic characterisation of the Margulis LDPC codes based on Cayley graphs

- $q = 5$ $n = 240$
- $q = 7$ $n = 672$
- $q = 11$ $n = 2640$
- $q = 13$ $n = 4368$
- $q = 17$ $n = 9792$
- $q = 19$ $n = 13680$

As as example for the creation of these codes, consider $q = 3$. The finite field $\mathbf{F}_q$ when $q = 3$ is $\mathbf{F}_3$ =0,1,2. Next, the set $SL_2(\mathbf{F}_3)$ consisting of all the 2x2 matrices with elements $a_{1,1}$, $b_{1,2}$, $c_{2,1}$, $d_{2,2}$ with entries in $\mathbf{F}_3$ =0,1,2 and having determinant $ad - bc = 1$ is defined. The set $SL_2(\mathbf{F}_3)$ has order $q^3 - q = 24$ in this example. The Table 4.2 includes the values for a, b, c, and d, that conform the set of matrices $SL_2(\mathbf{F}_3)$

a	b	c	d
0	1	2	0
0	1	2	1
0	1	2	2
0	2	1	0
0	2	1	1
0	2	1	2
1	0	0	1
1	0	1	1
1	0	2	1
1	1	0	1
1	1	1	2
1	1	2	0
1	2	0	1
1	2	1	0
1	2	2	2
2	0	0	2
2	0	1	2
2	0	2	2
2	1	0	2
2	1	1	1
2	1	2	0
2	2	0	2
2	2	1	0
2	2	2	1

Table 4.2: a, b, c, and d, that conform the set of matrices $SL_2(\mathbf{F}_3)$

The bipartite graph is formed by linking the vertices in the left, represent-

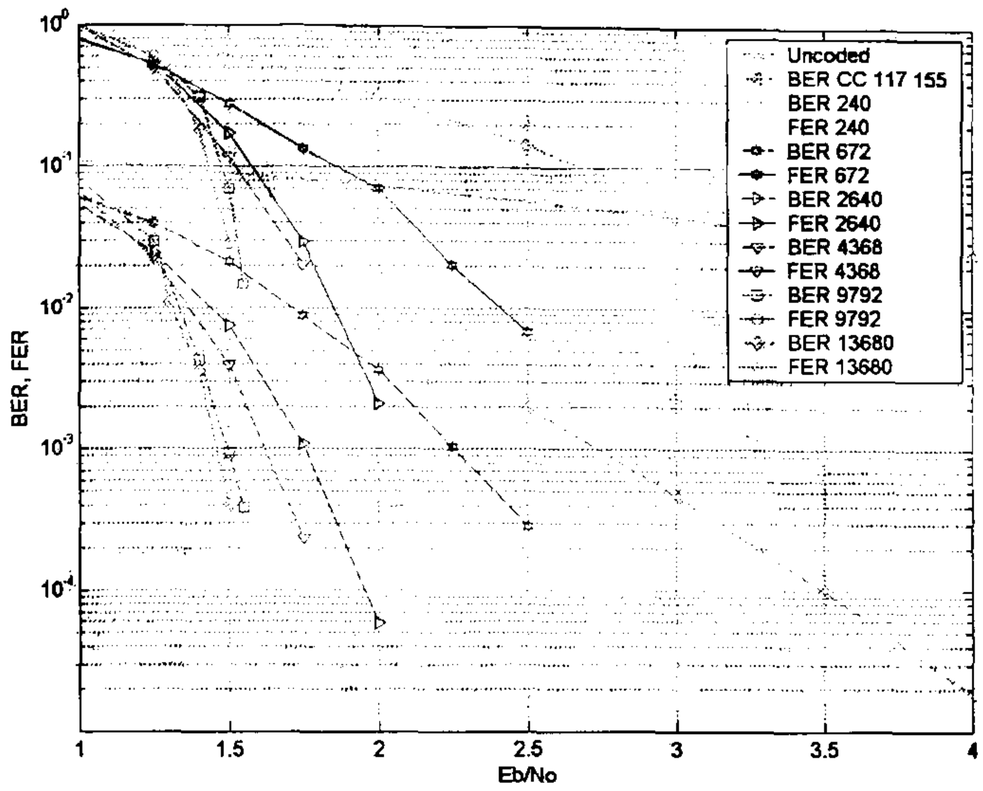

Figure 4.14: BER and FER performance for the Graphical Model (n,2,k) LDPC code with q=(5, 7, 11, 13, 17, 19) over the AWGN channel with QPSK modulation.

ing the variable nodes, with the vertices in the right, representing the check nodes. Each matrix g in the set $G = SL_2(\mathbf{F}_3)$, represents a single vertex in the bipartite graph. Two copies of G (i.e. G_{left-1} and G_{left-2}) contains the variable nodes, while one copy of G (G_{right}) contains the check nodes. An element $g \in G_{left-1}$ and an element $g \in G_{left-2}$ are connected with the elements $g \in G_{right}$, according to $g_u A^2, g_u ABA^{-1}, g_u B$ and $g_v A^{-2}, g_v AB^{-1}A^{-1}, g_v B^{-1}$ respectively, where A, A^{-1} B, and B^{-1} were defined in Chapter 2.

The resulting Margulis (48,3,24) LDPC parity check matrix is shown in the expresion 4.3.

The BER and FER performance of the LDPC codes based on the Cayley graph, also known as Margulis LDPC codes, is presented portrayed in Fig. 4.14. The comparative analysis of the Margulis LDPC codes is included in [88], and [44].

$$\mathbf{H} = \begin{bmatrix}
0010000000000000100010000100000000011000000000 \\
1000000000000010001000001000000100010000000000 \\
0100000000000000110000010000000001000100000000 \\
0000100010000100000000000000010000000000100010 \\
0000010100001000000000000001000000000000010001 \\
0001001000000010000000000000100000000000001100 \\
0000000011000000000100000000101000010000000000 \\
0000001000100000000100000000100010000100000000 \\
0000000100010000001000000001001000000100000000 \\
0100000000011000000000000000100010000000010000 \\
0010000001000100000000000000010001000000100000 \\
1000000000100010000000000000001100000000001000 \\
1000001000000000000010000100001000001000000000 \\
0100000100000000000010000100000100001000000000 \\
0010000010000000000010000000100000100000100000 \\
0000000000000100100001000100000000000001100000 \\
0000000000001000010000101000000000000100010000 \\
0000000000000011000000100100000000000010001000 \\
0001000000100001000000000100000100000000000100 \\
0000100001000000100000000100000100000000000010 \\
0000010000010000010000001000001000000000000001 \\
0000010000000000001000100000000000001100000001 \\
0001000000000000001000100000000000010010000100 \\
0000100000000000001100000000000010000100001000
\end{bmatrix} \tag{4.3}$$

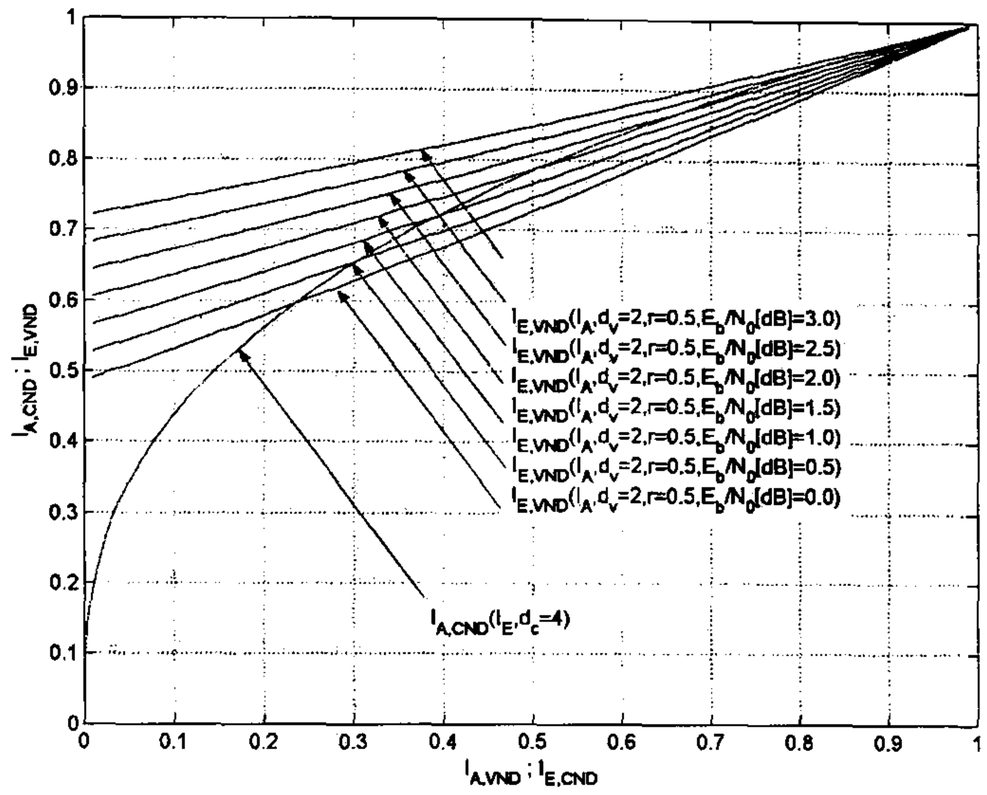

Figure 4.15: EXIT chart for the Graphical Model (240,2,120) LDPC code with E_b/N_0=0,0.5,1,1,5,2,2.5,3 dB.

Fig. 4.15, and Fig. 4.16 show the EXIT chart for the Graphical Model (240,2,120) LDPC code. Fig. 4.17, and Fig. 4.18 show the EXIT chart for the Margulis (240,3,120) LDPC code. Fig. 4.15 shows a faster increase of $I_{E,CND}$ after the first iteration, when compared to Fig. 4.17 for the same E_b/N_0, although this condition changes with the number of iterations. The higher the E_b/N_0, the faster this change will be produced. This condition is related to ρ and γ. When comparing Fig. 4.16 with Fig. 4.18 it is noticeable that the LDPC code with $\gamma = 2$ requires a higher E_b/N_0 to converge compared to the LDPC code with $\gamma = 3$; also, these graphs show for the same E_b/N_0, the LDPC code with $\gamma = 2$ requires more iterations to converge, than the code with $\gamma = 3$.

Based on Fig. 4.15 and Fig. 4.17, it becomes clear that if the decoding process is not performed in parallel, the decoder with $\gamma = 2$ should perform the first iteration, since it would produce higher quality extrinsic log-likehood

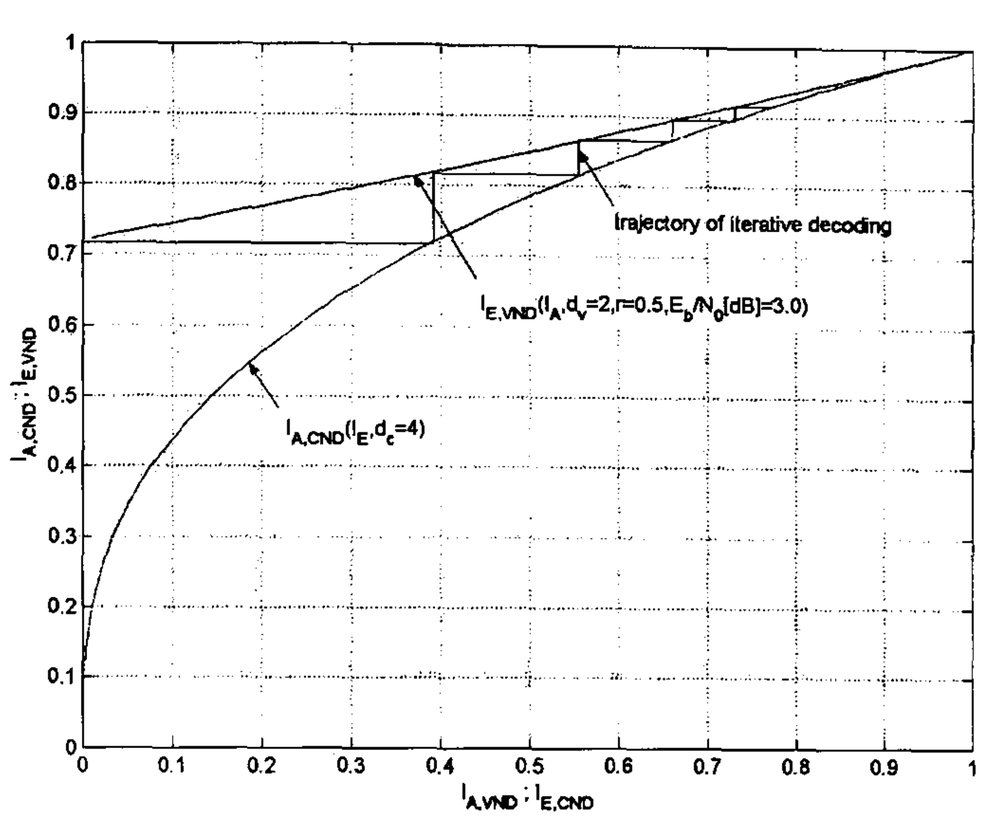

Figure 4.16: EXIT chart for the Graphical Model (240,2,120) LDPC code with E_b/N_0=3 dB.

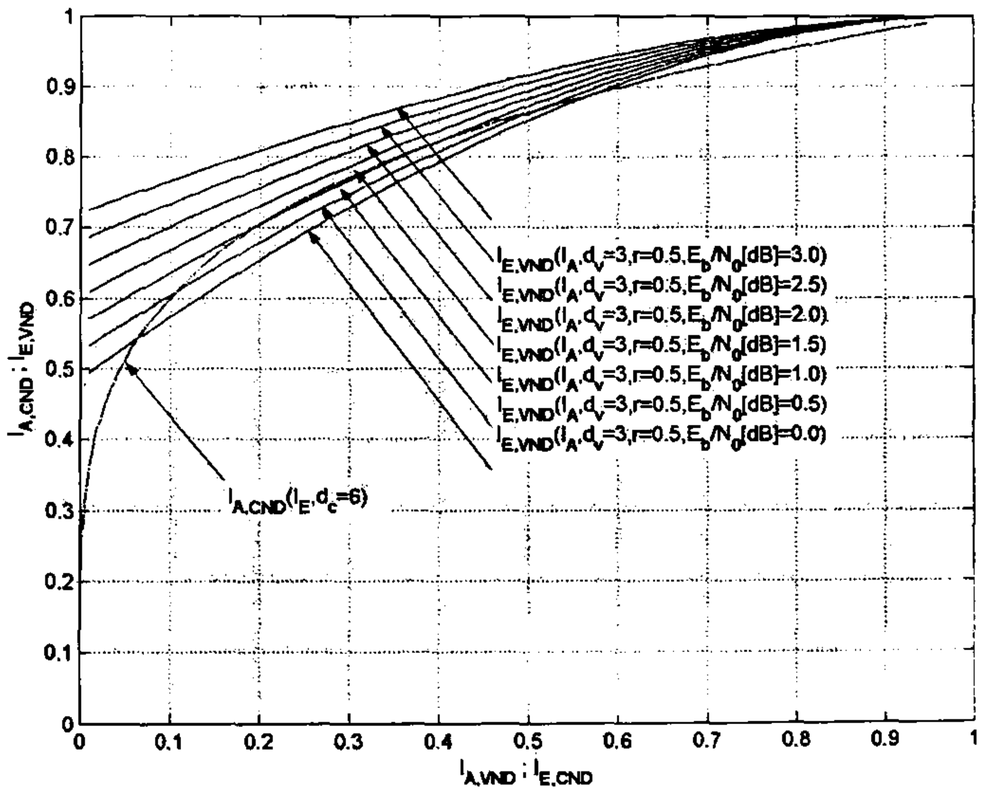

Figure 4.17: EXIT chart for the Margulis (240,3,120) LDPC code with E_b/N_0=0,0.5,1,1,5,2,2.5,3 dB.

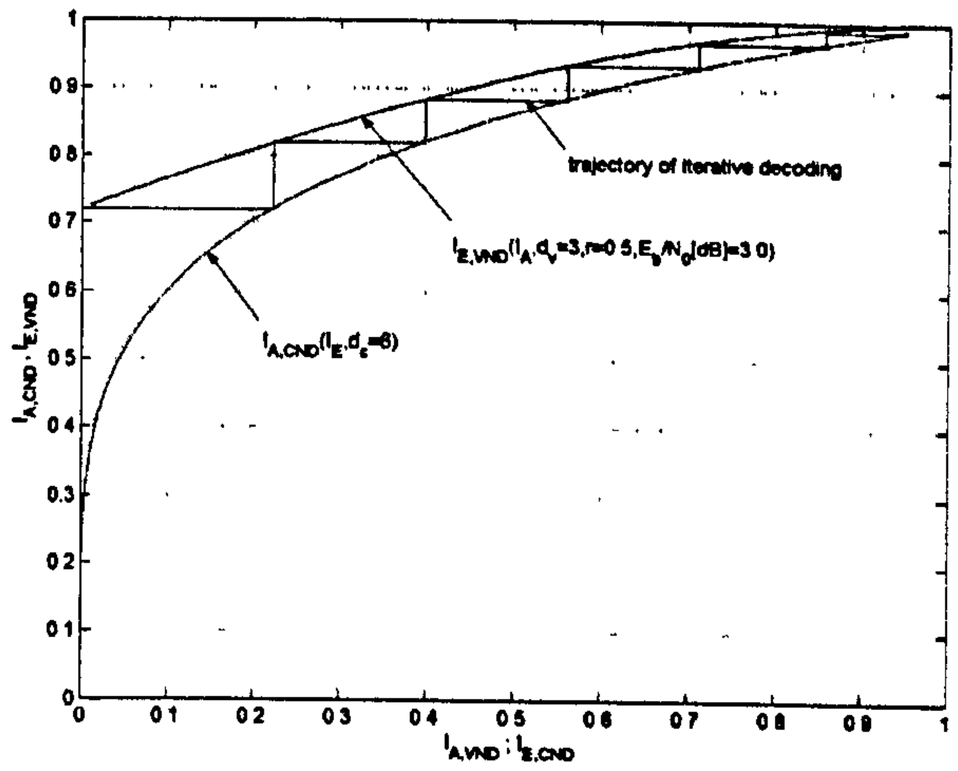

Figure 4.18: EXIT chart for the Margulis (240,3,120) LDPC code with E_b/N_0=3 dB.

values, compared to the decoder with $\gamma = 3$. These observations match similar ones presented in [15] where EXIT charts were employed to analyse the quality of the extrinsic information as a function of the signal to noise ratio.

4.5.1 Simulation results

Fig. 4.19 shows how the proposed concatenation in parallel of Margulis and GM LDPC codes improves the BER compared to the single components for an E_b/N_0 >1.5 dB, but performs worse than the Turbo Code with generators (13,15) with the same length and rate, with a 0.9 dB gap at 10^{-3}.

Fig. 4.20 shows how the proposed concatenation in parallel of Margulis and GM LDPC codes performs worse than the Turbo Code with generators (13,15) with the same length and rate, with a 0.9 dB gap at 10^{-2}.

Fig. 4.21 shows how the proposed concatenation in parallel of Margulis

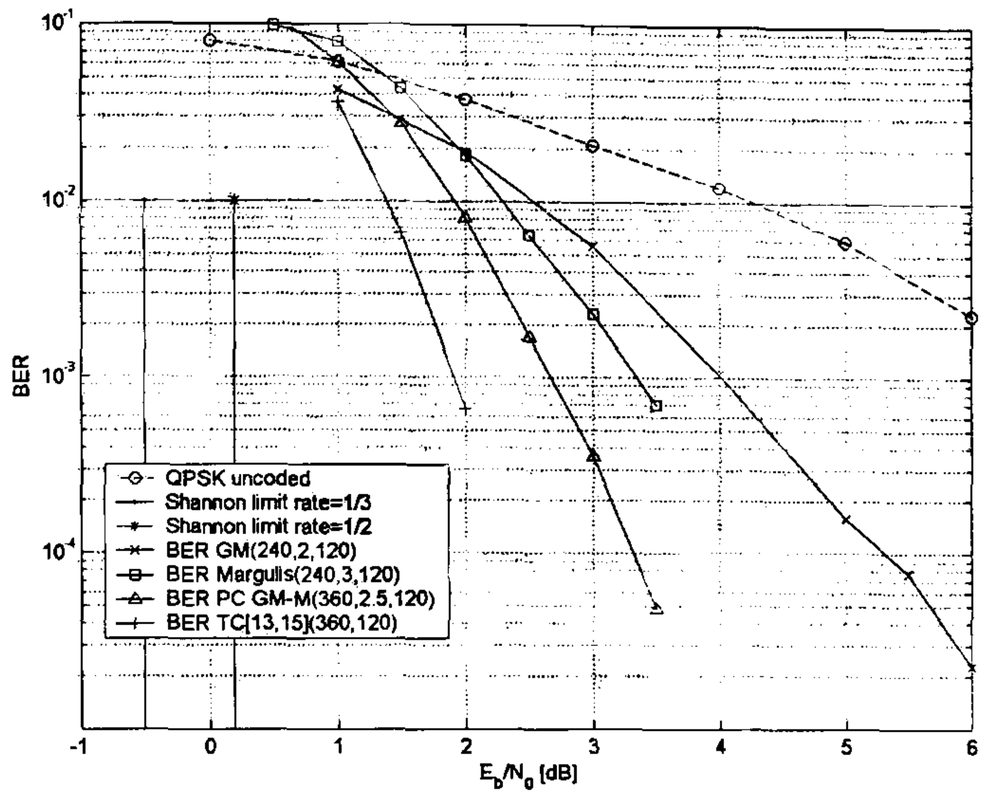

Figure 4.19: BER performance comparison between the parallel concatenation of the Graphical Model (240,2,120) with the Margulis (240,3,120) LDPC codes, and the Turbo Code (360,120) with rate=1/3 and generator polynomials [13,15] over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

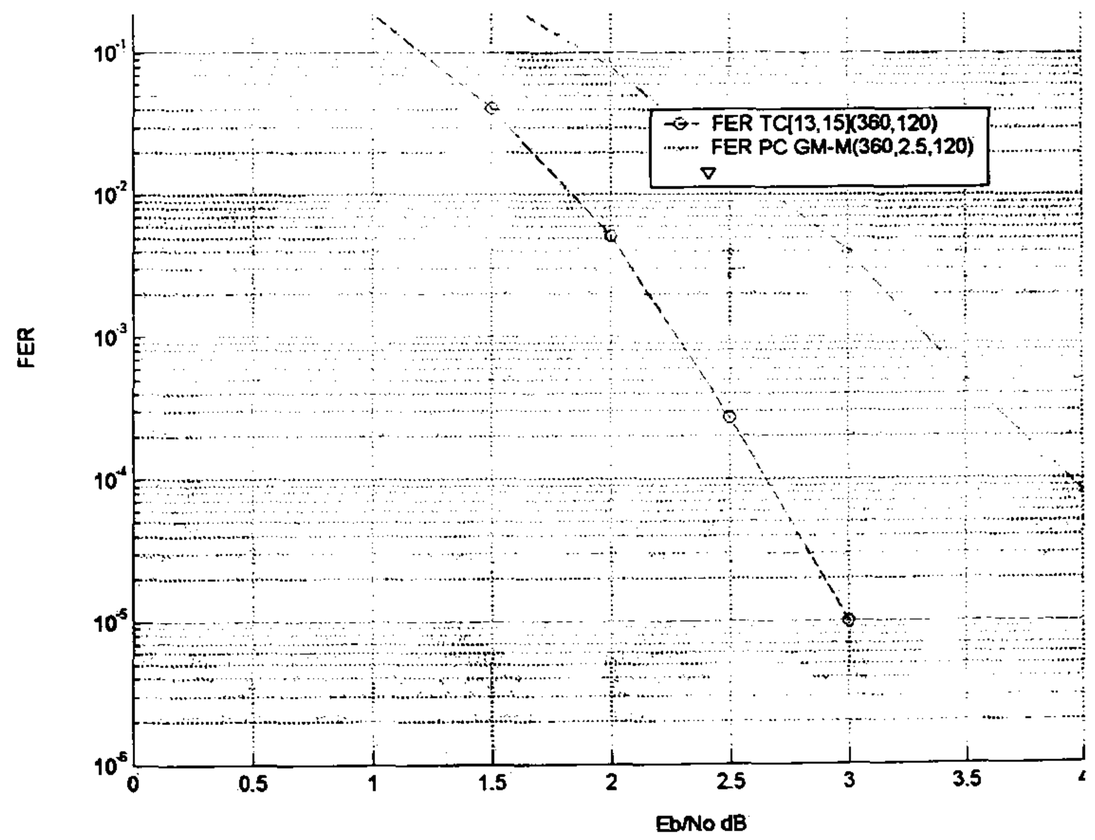

Figure 4.20: FER performance comparison between the parallel concatenation of the Graphical Model (240,2,120) with the Margulis (240,3,120) LDPC codes, and the Turbo Code (360,120) with rate=1/3 and generator polynomials [13,15] over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

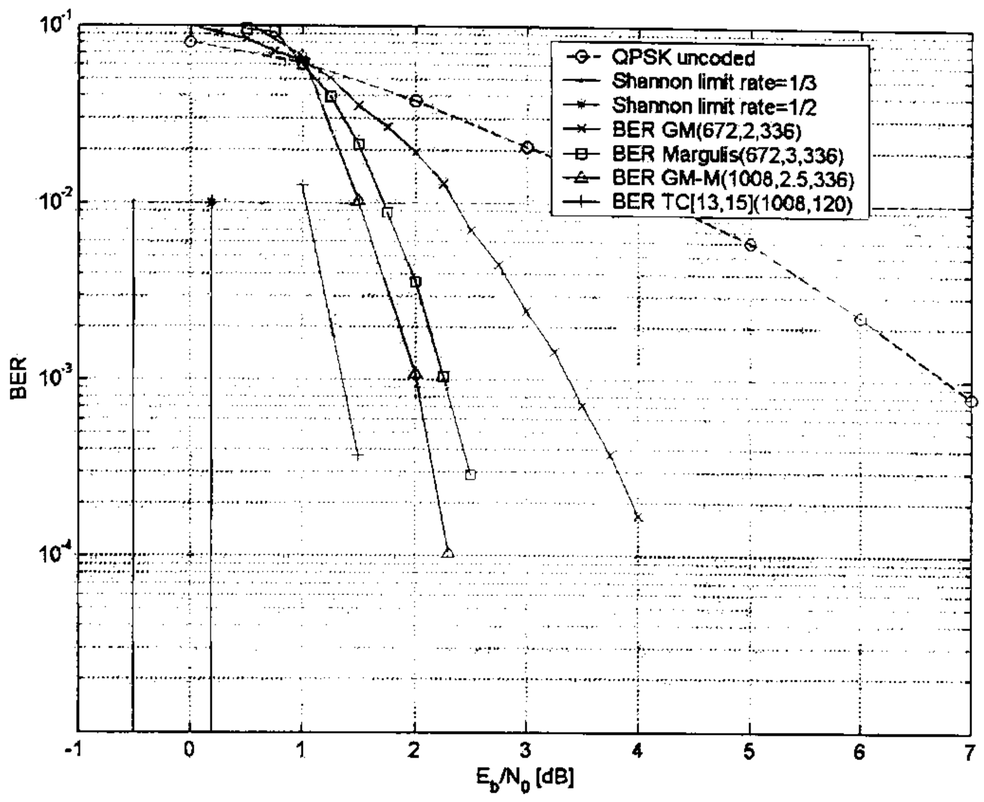

Figure 4.21: BER performance comparison between the parallel concatenation of the Graphical Model (672,2,336) with the Margulis (672,3,336) LDPC codes, and the Turbo Code (1008,336) with rate=1/3 and generator polynomials [13,15] over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

and GM LDPC codes improves the BER compared to the single components for an E_b/N_0 >1 dB, but performs worse than the Turbo Code with generators (13,15) with the same length and rate, with a 0.7 dB gap at 10^{-3}.

Fig. 4.22 shows how the proposed concatenation in parallel of Margulis and GM LDPC codes performs worse than the Turbo Code with generators (13,15) with the same length and rate, with a 0.7 dB gap at 10^{-2}.

When comparing Fig. 4.19 with Fig. 4.21 it is clear that the parallel concatenation of the Graphical Model and the Margulis LDPC codes, do achieve a better BER and FER performance than their constituent parts, after a relatively low E_b/N_0 has been reached. A benefit of the parallel concatenation of two LDPC codes with $\rho = 2$ and $\rho = 3$ is the improvement of the BER and FER performance at a lower E_b/N_0, compared to the parallel

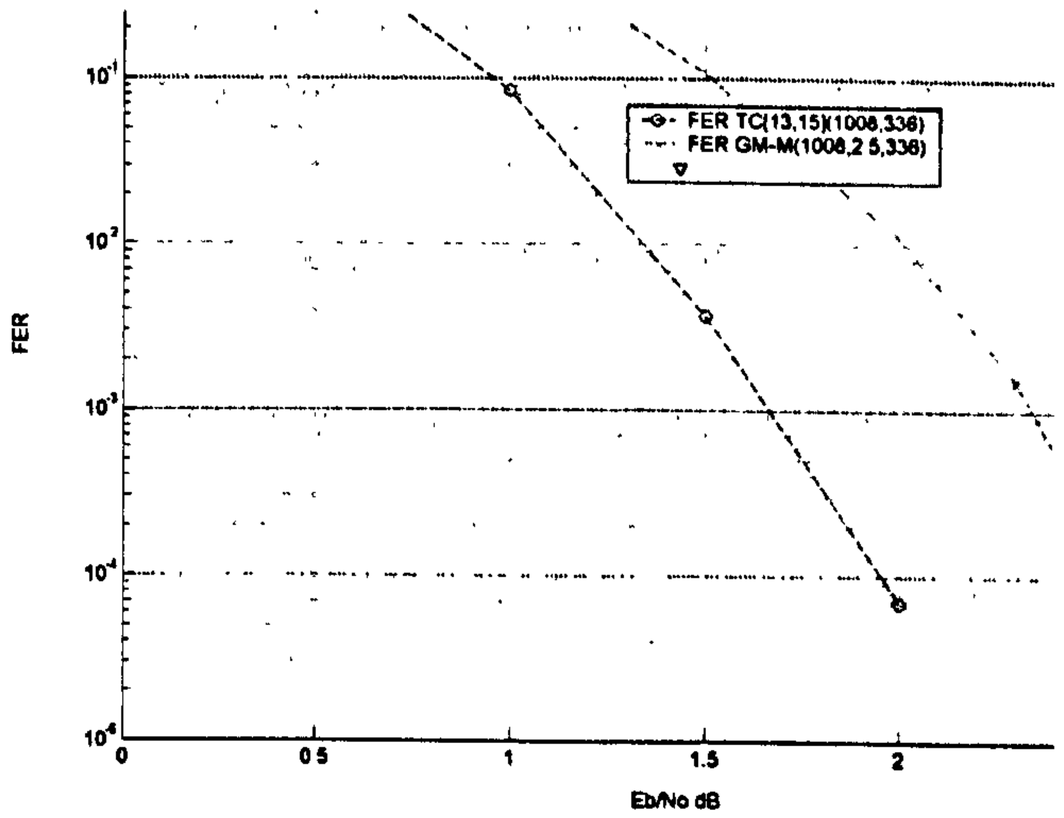

Figure 4.22: FER performance comparison between the parallel concatenation of the Graphical Model (672,2,336) with the Margulis (672,3,336) LDPC codes, and the Turbo Code (1008,336) with rate=1/3 and generator polynomials [13,15] over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

concatenation of two LDPC codes with the same $\rho = 3$, clearly depicted in the EXIT charts for such ensemble. This approach also reduces the amount of memory required to allocate the parity check matrices, when compared to a single LDPC code with the same length, rate and column weight, although this benefit is seen on any scheme with parallel concatenation. Finally, for a required BER or FER, once the E_b/N_0 required by any of the single components is reached, a reduction in the redundancy, and in the encoding and decoding processing is obtained by avoiding the concatenation of the code with $\rho = 2$.

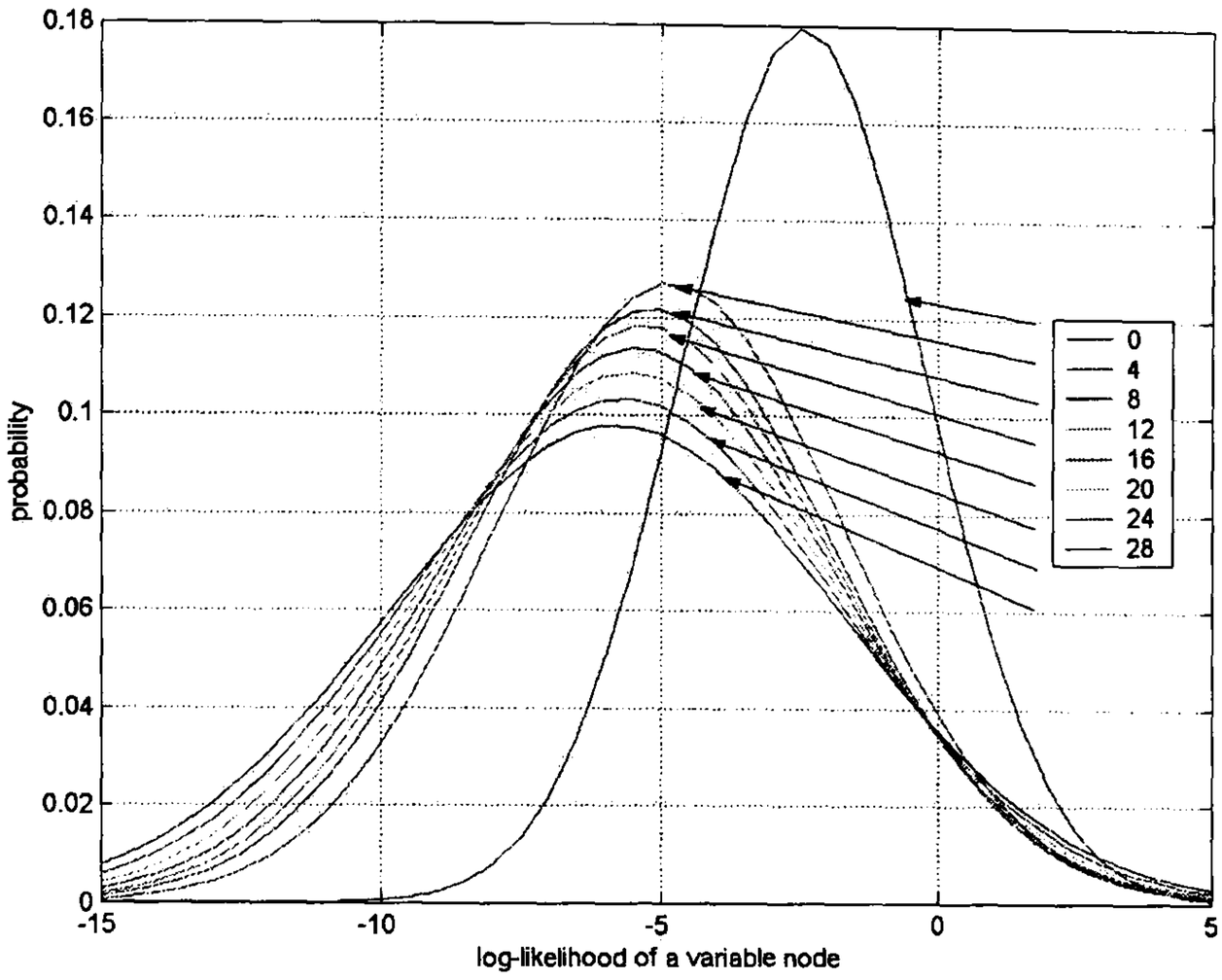

Figure 4.23: Simulation of the probability density function curves every 4 iterations, when calculating the convergence threshold for an LDPC code for a total of 28 iterations using DE and assuming an AWGN channel and QPSK modulation.

4.6 Parallel Concatenated $\gamma = 2$ and $\gamma = 3$ Girth Partition and Shift LDPC Codes

The construction of the Girth-Partition and Shift LDPC codes is presented in the previous Chapter. The GPS(240,2,120), GPS(240,3,120), GPS(672,2,336), and GPS(672,3,336) were considered to analyse their performance when concatenated in parallel. Although the GPS LDPC codes have a wide range of lengths available, the proposed lengths were chosen to match the lengths of the Margulis codes presented in the previous section, which have very limited design flexibility.

The capacity threshold [49] using DE techniques [48] for this ensemble of

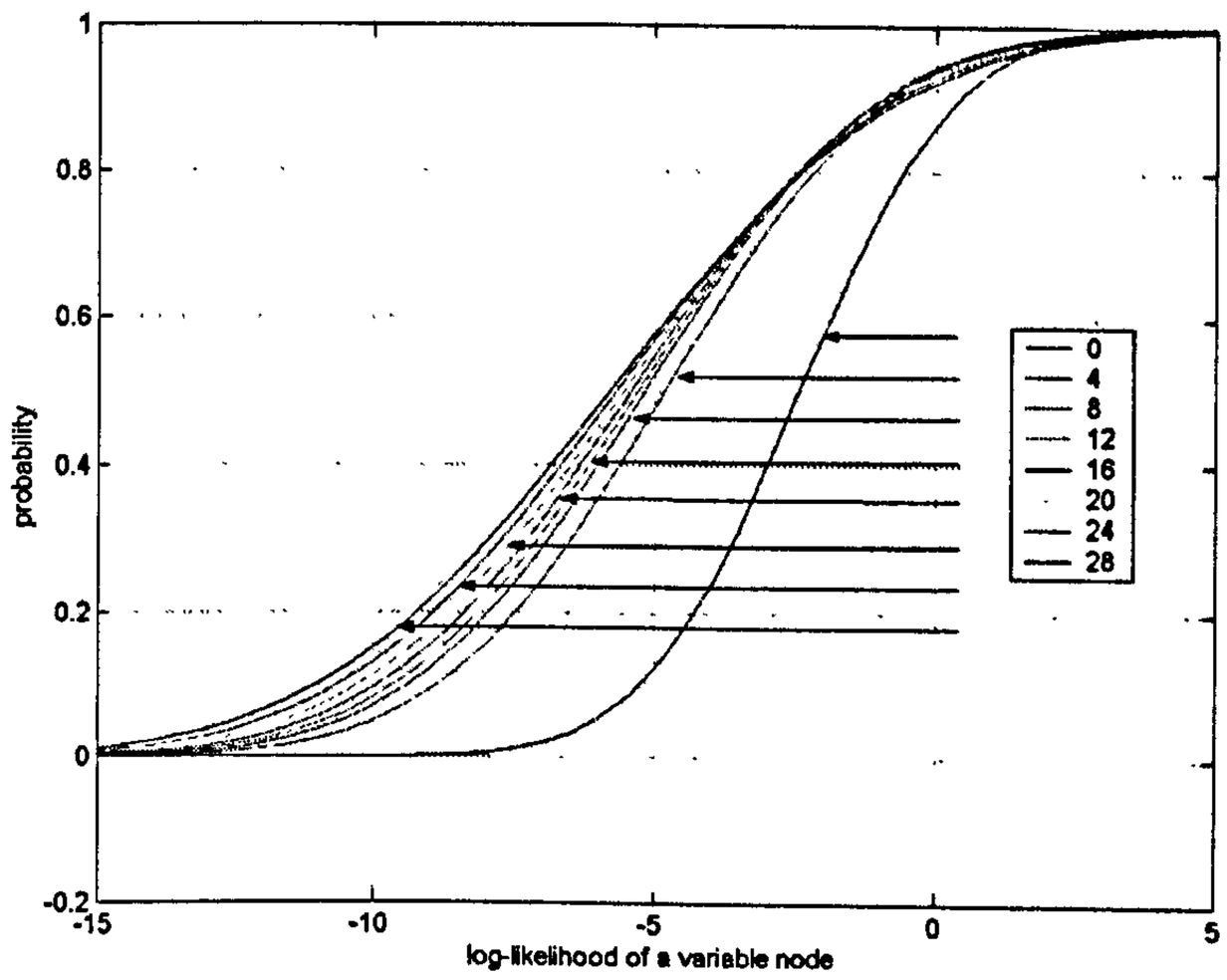

Figure 4.24: Simulation of the cumulative distribution function curves every 4 iterations, when calculating the convergence threshold for an LDPC code for a total of 28 iterations using DE and assuming an AWGN channel and QPSK modulation.

LDPC codes (i.e. the ensemble of LDPC codes with parameters (n,3,k=n/2) with $\rho = 6$ and $\gamma = 3$) is analised in depth in [8]. The threshold for the quantised Gaussian channel assuming 3-bit messages and for the Gaussian channel with a continuous message alphabet (i.e. BP) is 0.847 and 0.88 respectively. Fig. 4.23 and Fig. 4.24 simulate the probability density function and cumulative distribution function of the messages arriving at the variable nodes every four iterations, when the previous mentiones DE techniques are used.

4.6.1 Simulation results

Fig. 4.25 shows how the proposed concatenation in parallel of GPS LDPC codes improves the BER compared to the single components for an E_b/N_0 >1.5 dB, but performs worse than the Turbo Code with generators (13,15) with the same length and rate, with a 0.75 dB gap at 10^{-3}.

Fig. 4.26 shows how the proposed concatenation in parallel of GPS LDPC codes performs worse than the Turbo Code with generators (13,15) with the same length and rate, with a 0.8 dB gap at 10^{-2}.

Fig. 4.27 shows how the proposed concatenation in parallel of GPS LDPC codes improves the BER compared to the single components for an E_b/N_0 >1.5, but performs worse than the Turbo Code with generators (13,15) with the same length and rate, with a 0.55 dB gap at 10^{-3}.

Fig. 4.28 shows how the proposed concatenation in parallel of GPS LDPC codes performs worse than the Turbo Code with generators (13,15) with the same length and rate, with a 0.8 dB gap at 10^{-2}.

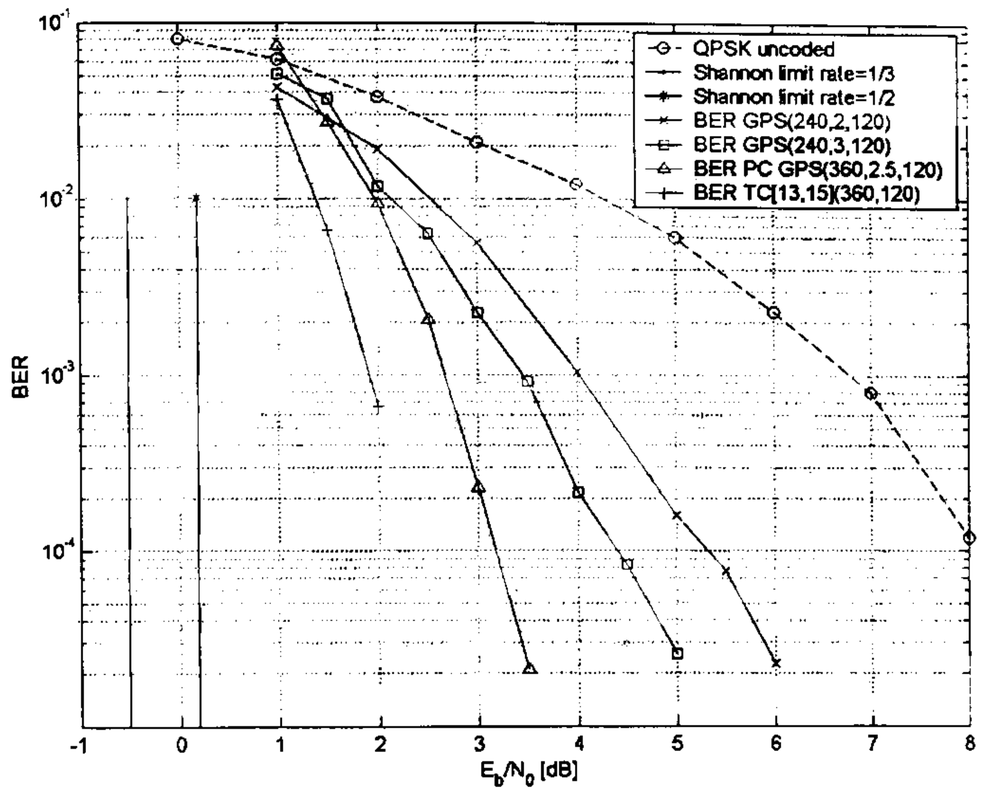

Figure 4.25: BER performance comparison between the parallel concatenation of the GPS (240,2,120) with the Margulis (240,3,120) LDPC codes, and the Turbo Code (360,120) with rate=1/3 and generator polynomials [13,15] over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

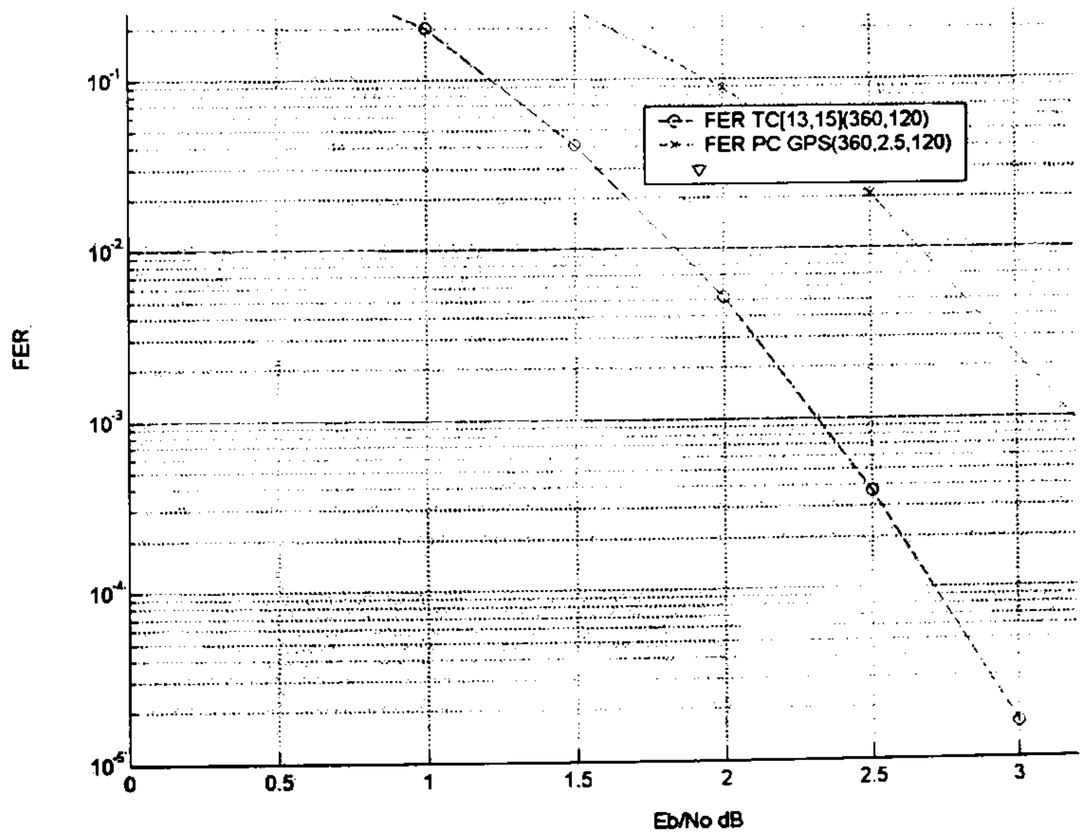

Figure 4.26: FER performance comparison between the parallel concatenation of the Graphical Model (240,2,120) with the Margulis (240,3,120) LDPC codes, and the Turbo Code (360,120) with rate=1/3 and generator polynomials [13,15] over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

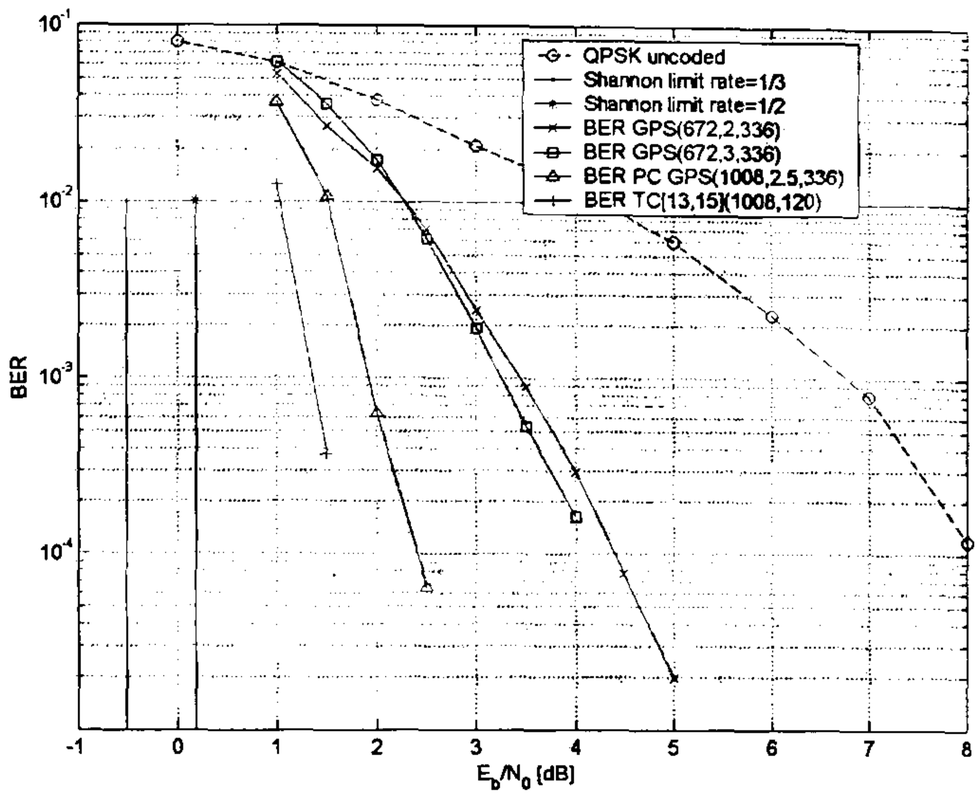

Figure 4.27: BER performance comparison between the parallel concatenation of the Graphical Model (672,2,336) with the Margulis (672,3,336) LDPC codes, and the Turbo Code (1008,336) with rate=1/3 and generator polynomials [13,15] over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

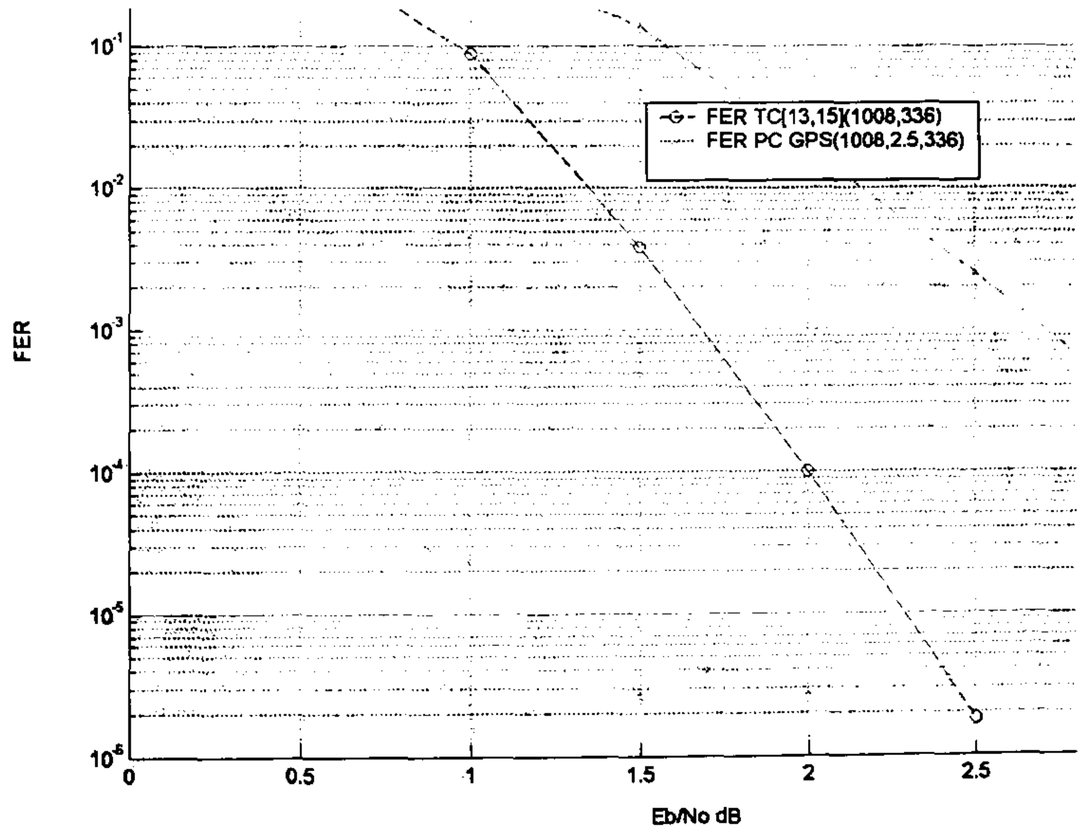

Figure 4.28: FER performance comparison between the parallel concatenation of the Graphical Model (672,2,336) with the Margulis (672,3,336) LDPC codes, and the Turbo Code (1008,336) with rate=1/3 and generator polynomials [13,15] over an AWGN channel with QPSK modulation, sum-product decoding and 30 iterations.

When comparing Fig. 4.25 and 4.26 against Fig. 4.27 and 4.28, it is clear that the parallel concatenation of the GPS LDPC codes with $\rho = 2$ and $\rho = 3$, do achieve a better BER and FER performance than their constituent parts, after a relatively low E_b/N_0 has been reached. As the length of the constituent codes is increased, the gap between the BER curve for the turbo code and the BER curve for the GPS parallel concatenation, is reduced.

Even more interesting is to note that although the Graphical Model ($\rho = 2$) and Margulis ($\rho = 3$) LDPC codes perform better in terms of BER and FER when compared to the GPS ($\rho = 2$) and ($\rho = 3$) LDPC codes respectively, when concatenated, it is the last ones that perform the best. When concatenated, the gap between the BER curves is 0.1 dB when $n = 360$ and 0.15 dB when $n = 1008$ for a BER of 10^{-3}. One explanation for this behavior comes from the bipartite graph. Although the Graphical Model ($\rho = 2$) and Margulis ($\rho = 3$) LDPC codes perform better independently, when concatenated they could create short unwanted cycles, reducing faster the independancy of the extrinsic information. If there were short cycles present in the bipartite graph for the parallel concatenation of the GPS ($\rho = 2$) and ($\rho = 3$) LDPC codes, they have the flexibility to be modified ad hoc. Such flexibility is not shared by the Margulis LDPC codes.

4.7 Conclusions

The parallel concatenation of EG and PG LDPC codes shows limited performance improvement, when compared to the performance of the constituent parts, and therefore their application should be limited to systems that re-

quire low encoding complexity and low decoding complexity in terms of processing power, low memory assigned to contain the parity-check matrix, and short frames to reduce the delay. Since these codes achieve successful decoding in a very small number of iterations after a certain E_b/N_0 has been reached, it is desirable to combine one EG or PG LDPC code, with an LDPC code whose $\rho = 2$.

The bipartite graph is an important design instrument for the selection of the codes to be concatenation in parallel, as it helps to prevent short cycles not contained in the constituent codes. Well chosen constituent codes can achieve good BER and FER performance when concatenated in parallel, even if their constituent codes perform poorly.

Specific properties of the constituent codes, observed through the use of EXIT charts, define the overall performance of the concatenated decoder scheme. Codes that contain *good* specific properties are desirable. However, not all the LDPC codes created through well structured procedures can fulfil such requirements. On the other hand, for randomly created LDPC codes, many of the desirable properties are considered as design parameters, such as the column weight, but poor structure would limit their implementation, even if their performance is excellent in terms of BER and FER [15], [13], and [14]. Instead of focussing on the creation of LDPC codes [21], or the modification of existing LDPC codes [43], the novelty of this work relies on the proposed selection of LDPC codes to be concatenated in parallel, improving the characterisation of such codes prevously published, and the analysis of their performance. This analysis is justified since they contain some of the *good* specific properties demanded by the EXIT charts to improve

the overall BER and FER performance. Such analysis is unique to this work.

In the same way some codes, or concatenation of codes perform better than others, EXIT charts, first applied to the performance analysis of an iterative demapper concatenated with a convolutional decoder, are also used to analyse other subsystems belonging to joint iterative receivers. Further analysis of the desirable properties of the subsystems involved in such joint iterative receivers can be useful, when one of the constituent part is the parallel concatenation of LDPC codes. This topic is considered in the next Chapter.

Chapter 5

Iterative Demapping, Parallel Channel Decoding and Source Decoding for Multilevel Modulation

5.1 Purpose

Traditional designs of receivers in the communications industry have independent subsystems with information flowing in only one direction, making hard decisions at each step of the process. State of the art receivers process data in an iterative manner through modified algorithms that both accept and produce soft values related to the reliability of the information (e.g. the channel decoder exchanges log-likelihood values with other subsystems). Such schemes vary according to the points where the exchange of extrin-

sic information takes place, proving to perform better in terms of BER and FER when compared to traditional designs, with a trade-off in complexity and latency. [100]

Several papers present the analysis of the iterative source and channel decoding scheme, including [101], [59], [56], [64], [54], [61], and [58], among others. Although it has proved beneficial to use any redundancy left after source encoding, or to avoid any source encoding when the redundancy at the information source is small, nonetheless, the performance achieved is inferior or similar to that of the uncoded signal when the channel conditions are poor (low E_b/N_0).

A SISO demapper is considered in [12], [11], and [102], which improves the BER and FER performance of the channel decoder when a modified convolutional decoder is employed. Previous research has proved that the gain obtained using this approach depends on the type of mapping employed. While gray mapping does not improve the performance, other mappings do (e.g. antigray mapping). This behaviour is analysed by means of EXIT charts.

The purpose of this study is to analyse the performance of well structured LDPC codes, in particular Margulis constructions, GM constructions and GPS constructions, when concatenated in parallel, and decoded jointly in a receiver with an iterative demapper and an iterative source decoder. Understanding the impact of the individual sections that conform the joint iterative receiver, may provide valuable information to understand the conditions under which this receiver can perform the best.

The first approach to understand the overall performance, consists on

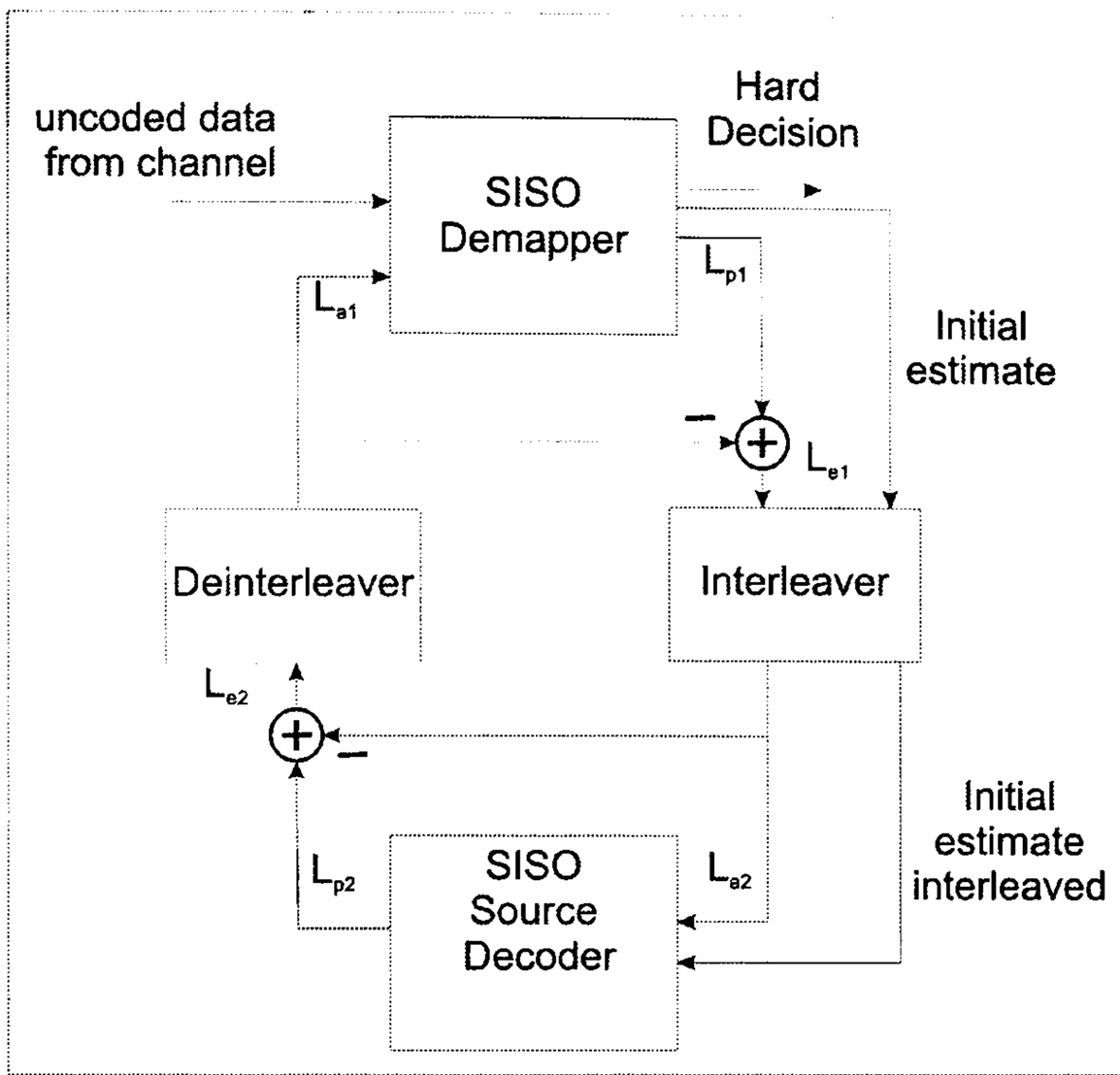

Figure 5.1: Block diagram for the joint demapping and source decoding scheme

revising the BER performance achievable without the parallel concatenated decoder. The system to be revised is a joint iterative demapper and source decoder, as depicted in Fig. 5.1. The characteristics of both sections are varied to understand their behaviour.

Although the analitical description of the iterative demapper, and the iterative source decoder has been previously published, both concepts are reproduced next.

5.2 Demapping with a-priori knowledge

The analytical description of the iterative demapper is well documented by the reference provided in [12], and here is reproduced for four and eight point

constellations, together with the general derivation.

A binary signal $S \in \{0,1\}$ is mapped into the complex symbols $X = map(S)$ according to the signal constellation of any modulation scheme required and transmitted. By doing so, all the bits related to a particular symbol become dependent on each other. The demapper needs to calculate the log-likelihood ratios of the received mapped symbols. It is the mutual dependancy of the bits linked up in the constellation symbol that is exploited in the system considered.

The demapper obtains the log-likelihood values of the received signal $Z = h * X + n$, where h represents the channel function in the time domain, n represents the additive white gaussian noise, and $*$ represents the convolution. In the case of a four points constellation such as 4QAM (i.e. where each constellation point represents two bits), the log-likelihood value of the first binary bit S_0 conditioned to the received signal Z at the output of the matched filter is,

$$\begin{aligned} L\left(s_0|Z\right) &= ln\frac{p(s_0=1|Z)}{p(s_0=0|Z)} \\ &= ln\frac{p(s_0=1,s_1=0|Z)+(s_0=1,s_1=1|Z)}{p(s_0=0,s_1=0|Z)+(s_0=0,s_1=1|Z)} \end{aligned} \tag{5.1}$$

We can assume that the probability of the individual bits forming the symbol is equal to the combined probability, as long as the symbols are independent. Based on the previous statement, the proposed scheme has to consider the independence of continuous symbols produced by the source, that would be mapped into a constellation point. If continuous source symbols are not statistically independent, such as in the case of a Markov source. the introduction of an interleaver is necessary. (e.g. in the case of a source

producing blocks of three independent symbols, but with dependence between blocks, and 8PSK mapping, an interleaver would not be necessary as long as the mapping operation was done in phase with the independent source symbols, and assuming transmission over an AWGN channel). If the independence of source symbols exists, the joint probability can be expressed as the multiplication of the individual probabilities.

$$p\left(s_0 = 1, s_1 = 1\right) = p\left(s_0 = 1\right) \cdot p\left(s_1 = 1\right) \tag{5.2}$$

By making use of Bayes' rule $p(a|b) \cdot p(b) = p(b|a) \cdot p(a)$ and the independence of source symbols in eq. 5.1, we get the expression shown in eq. 5.3.

$$\begin{aligned}
L\left(s_0|Z\right) &= ln\frac{p(Z|s_0{=}1,s_1{=}0)p(s_0{=}1,s_1{=}0)+p(Z|s_0{=}1,s_1{=}1)p(s_0{=}1,s_1{=}1)}{p(Z|s_0{=}0,s_1{=}0)p(s_0{=}0,S_1{=}0)+p(Z|s_0{=}0,s_1{=}1)p(s_0{=}0,s_1{=}1)} \\
&= ln\frac{p(Z|s_0{=}1,s_1{=}0)\cdot p(s_0{=}1)\cdot p(s_1{=}0)+p(Z|s_0{=}1,s_1{=}1)\cdot p(s_0{=}1)\cdot p(s_1{=}1)}{p(Z|s_0{=}0,s_1{=}0)\cdot p(s_0{=}0)\cdot p(s_1{=}0)+p(Z|s_0{=}0,s_1{=}1)\cdot p(s_0{=}0)\cdot p(s_1{=}1)} \\
&= ln\frac{p(s_0{=}1)\cdot[p(Z|s_0{=}1,s_1{=}0)\cdot p(s_1{=}0)+p(Z|s_0{=}1,s_1{=}1)\cdot p(s_1{=}1)]}{p(s_0{=}0)\cdot[p(Z|s_0{=}0,s_1{=}0)\cdot p(s_1{=}0)+p(Z|s_0{=}0,s_1{=}1)\cdot p(s_1{=}1)]} \\
&= ln\frac{p(s_0{=}1)}{p(s_0{=}0)} + ln\frac{p(Z|s_0{=}1,s_1{=}0)\cdot p(s_1{=}0)+p(Z|s_0{=}1,s_1{=}1)\cdot p(s_1{=}1)}{p(Z|s_0{=}0,s_1{=}0)\cdot p(s_1{=}0)+p(Z|s_0{=}0,s_1{=}1)\cdot p(s_1{=}1)} \\
&= ln\frac{p(s_0{=}1)}{p(s_0{=}0)} + ln\frac{p(Z|s_0{=}1,s_1{=}0)+p(Z|s_0{=}1,s_1{=}1)\cdot\frac{p(s_1{=}1)}{p(s_1{=}0)}}{p(Z|s_0{=}0,s_1{=}0)+p(Z|s_0{=}0,s_1{=}1)\cdot\frac{p(s_1{=}1)}{p(s_1{=}0)}} \\
&= ln\frac{p(s_0{=}1)}{p(s_0{=}0)} + ln\frac{p(Z|s_0{=}1,s_1{=}0)+p(Z|s_0{=}1,s_1{=}1)\cdot e^{ln\frac{p(s_1{=}1)}{p(s_1{=}0)}}}{p(Z|s_0{=}0,s_1{=}0)+p(Z|s_0{=}0,s_1{=}1)\cdot e^{ln\frac{p(s_1{=}1)}{p(s_1{=}0)}}} \\
&= L_a\left(s_0\right) + ln\frac{p(Z|s_0{=}1,s_1{=}0)+p(Z|s_0{=}1,s_1{=}1)\cdot e^{La(s_1)}}{p(Z|s_0{=}0,s_1{=}0)+p(Z|s_0{=}0,s_1{=}1)\cdot e^{La(s_1)}}
\end{aligned} \tag{5.3}$$

The term $ln\frac{p(s_k=1)}{p(s_k=0)}$ represents the a-priori information of the bit s_k $\forall k$; this is replaced by the log-likelihood $L_a\left(s_k\right)$ at the end of eq. 5.3.

After the same manipulation is applied to eq. 5.2, the log-likelihood value

of the first binary bit S_0 conditioned to the received signal Z at the output of the matched filter for an eight point constellation such as 8PSK (i.e. where each point represents three bits) is written according to eq. 5.4.

$$\begin{aligned} L\left(s_0|Z\right) &= L_a\left(s_0\right) \\ &+ln\frac{p(Z|s_0=1,s_1=0,s_2=0)+\cdots+p(Z|s_0=1,s_1=1,s_2=1)\cdot e^{La(s_1)}\cdot e^{La(s_2)}}{p(Z|s_0=0,s_1=0,s_2=0)+\cdots+p(Z|s_0=0,s_1=1,s_2=1)\cdot e^{La(s_1)}\cdot e^{La(s_2)}} \end{aligned} \tag{5.4}$$

The generalization of eq. 5.4 is described by eq. 5.5, where $s_{j,j=0\ldots M-1,j\neq k} \equiv b\left(i\right)$ denotes the joint event of the variables $s_{j,j=0\ldots M-1,j\neq k}$ and takes the value 0 or 1 according to the binary representation of i, where M denotes the number of points in the constellation (e.g. for M=3, k=1, and i=3, $x_0 = 1$ and $x_2 = 1$), and $c(i,h)$ takes the value 1 if the bit number h is set in the binary decomposition of i, otherwise it is 0, with

$$\begin{aligned} L\left(s_k|Z\right) &= L_a\left(s_k\right) \\ &+ln\frac{\sum_{i=0}^{2^{M-1}-1} p\Big(Z|s_k=1,s_{j,j=0\ldots M-1,j\neq k}\equiv b(i)\Big)\cdot e^{\sum_{j=0,j\neq k,c(i,h)=1}^{M-1} L_a\left(s_j\right)}}{\sum_{i=0}^{2^{M-1}-1} p\Big(Z|s_k=0,s_{j,j=0\ldots M-1,j\neq k}\equiv b(i)\Big)\cdot e^{\sum_{j=0,j\neq k,c(i,h)=1}^{M-1} L_a\left(s_j\right)}} \end{aligned} \tag{5.5}$$

$$h = \begin{cases} j, & j < k \\ j-1, & j \geq k \end{cases} \tag{5.6}$$

In [50], the EXIT charts are used to analyse the convergence threshold for a joint iterative demapping and convolutional decoding scheme. Through this study it becomes clear that when the proposed scheme is combined with gray mapping, the convergence threshold is improved compared to other

mapping schemes (i.e. the system decodes information successfully at a lower E_b/N_0 compared to other mapping schemes), but at the cost of achieving lower BER after the number of iterations is increased. In the EXIT chart, different mapping schemes are represented by straight lines with different slopes, where gray mapping results in a horizontal line (i.e. for any value of mutual information at the input of the SISO demapper, the output will always show the same mutual information value).

5.3 Hidden Markov Source estimation algorithm

In [43], a hidden Markov model is used to estimate the parameters of the source model with the received sequence z_1^k from z_1 to z_K. A hidden Markov source generates symbols, which are represented through r_k possible states. $b_k = \{r_{k-1} = i, s_k, r_k = j\}$ is the $k-th$ transition branch and $p(s, r_j|r_i)$ is the transition probability, where $s \in \{0, 1\}$ for a two state hidden Markov source with initial parameters $\lambda = \{p(s, r_j|r_i), \rho, s \in \{0, 1\}, i, j = 0, 1\}$, where ρ is the initial state of the source model. The maximum a-posteriori (MAP) algorithm applied on a trellis is also known as the BCJR algorithm. The calculation of the propagation of the probability likelihoods in the trellis is known as the forward and backward equations, described by eq. 5.7 and 5.9, respectively. $\alpha_k(i)$ represents the forward equation, while $\beta_k(i)$ reprsents the backward equation.

$$\begin{aligned}\alpha_k(i) &= P(r_k = i|z_1^k, \lambda) \\ &= \frac{\sum_{r_{k-1}} \sum_{s_k} P(r_{k-1}{=}j, s_k, r_k{=}i, z_1^{k-1}, z_k|\lambda)}{P(z_1^{k-1}|\lambda)P(z_k|P(z_1^{k-1},\lambda)} \\ &= \frac{\sum_{r_{k-1}} \sum_{s_k} \alpha_{k-1}(j)P(z_k|s_k)P(s_k, r_k{=}i|r_{k-1}{=}j,\lambda)}{P(z_k|z_1^{k-1},\lambda)}\end{aligned} \tag{5.7}$$

$$\begin{aligned}P(z_k|z_1^{k-1},\lambda) &= \sum_{r_{k-1}} \sum_{s_k} \sum_{r_k} P(r_{k-1} = j, s_k, r_k = i, z_k|z_1^{k-1},\lambda) \\ &= \sum_{r_{k-1}} \sum_{s_k} \sum_{r_k} \alpha_{k-1}(j)P(s_k, r_k = i|r_{k-1} = j,\lambda)P(z_k|s_k)\end{aligned} \tag{5.8}$$

$$\begin{aligned}\beta_k(i) = \quad & \frac{P(z_{k+1}^K|r_k{=}i,\lambda)}{P(z_{k+1}^K|z_1^k,\lambda)} \\ &= \frac{\sum_{s_{k+1}} \sum_{r_{k+1}} \beta_{k+1}(j)P(z_{k+1}|s_{k+1})P(s_{k+1}, r_{k+1}{=}j|r_k{=}i,\lambda)}{P(z_{k+1}|z_1^k,\lambda)}\end{aligned} \tag{5.9}$$

$$\begin{aligned}P(b_k) &= P(r_{k-1} = i, s_k, r_k = j|z_1^K, \lambda) \\ &= \frac{\alpha_{k-1}(i)\beta_k(j)P(s_k, r_k{=}j|r_{k-1}{=}i,\lambda)P(z_k|s_k)}{P(z_k|z_1^{k-1},\lambda)}\end{aligned} \tag{5.10}$$

$$\begin{aligned}L(s_k) &= ln\textstyle\sum_{r_{k-1}} \sum_{r_k} \alpha_{k-1}(i)\beta_k(j)P(s_k = 1, r_k = j|s_{k-1} = i, \lambda) \\ &\quad -ln\textstyle\sum_{r_{k-1}} \sum_{r_k} \alpha_{k-1}(i)\beta_k(j)P(s_k = 0, r_k = j|s_{k-1} = i, \lambda)\end{aligned} \tag{5.11}$$

The conditioned transition probability is represented by eq. 5.8. The probability of the $k-th$ branch may be computed as eq. 5.10. Finally, the extrinsic information expressed as a log-likelihood ratio, is described by eq. 5.11.

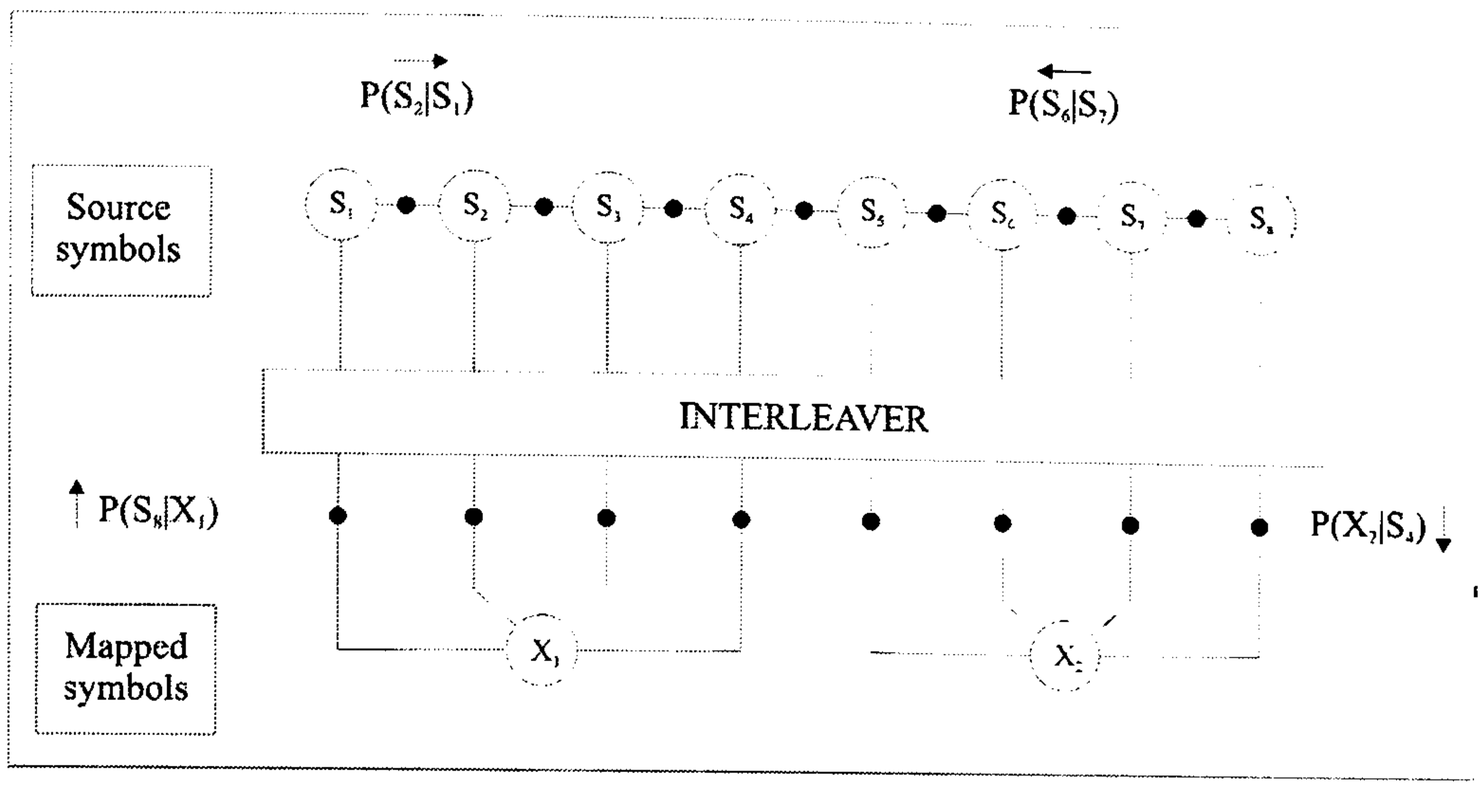

Figure 5.2: Factor graph for a joint demapping and source decoding scheme

5.4 Joint Demapping and Source Decoding

The factor graph describing the joint demapping and source decoding receiver, is shown in Fig. 5.2. The model considers only the conditional probability between adjacent source symbols.

To evaluate the performance of the this scheme, the source at the transmitter is modelled as a first order Markov source with two states, which produces the binary sequence of information bits $S_0, S_1, ..., S_N \in 0, 1$. The state transition probabilities are displayed in Table 5.1.

Initial State	Final State	Transition Probability
State 1	State 1	p_1
State 1	State 2	$1 - p_1$
State 2	State 2	p_2
State 2	State 1	$1 - p_2$

Table 5.1: State Transition Probabilities

Since the BCJR algorithm is better understood on a trellis, Fig. 5.3 shows the two state trellis model for the proposed Markov Source. The initial state of the source is unknown at the receiver and therefore equal probability has

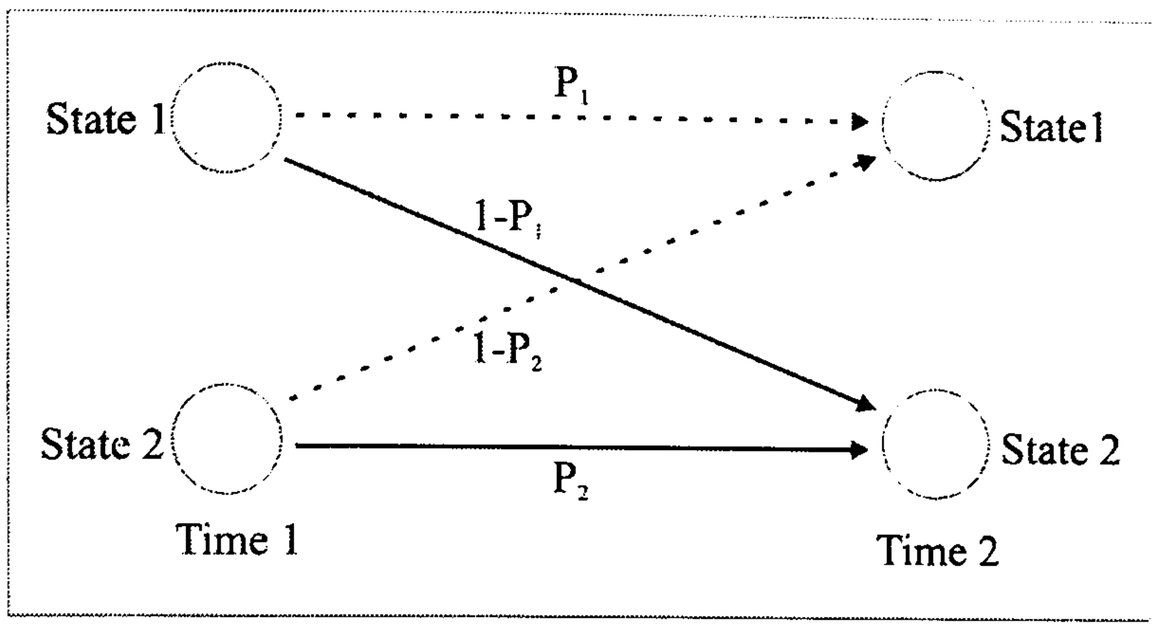

Figure 5.3: Trellis for the First order Markov Source model with two states

been assumed for α_0 and β_K.

It is stated in [12] that the performance of the demapper depends on the particular mapping constellation and the maximum number of iterations set up, or according to a different criteria, that it depends on the mutual information $I(X;Z)$, and the number of iterations set up. X represents the transmitted constellation sybol, and Z represents the same constellation symbols after being affected by the AWGN channel. Gray mapping has the highest mutual information among all the possible constellations for each one of the constellation points; however, no improvement can be obtained even after many iterations. This behaviour is due to the low correlation between bits that represent the constellation symbol. To complete the first iteration, the decoder sends its estimate to the demapper, but since the bits in the QPSK symbol are independent from each other, no a-posteriori information generated by one of the bits can be useful to improve the estimate of the other bit. Constellations that show a lower mutual information, correspond to constellations whose symbols do not provide maximum independence between the constituent bits. This, on the other hand, improves the BER

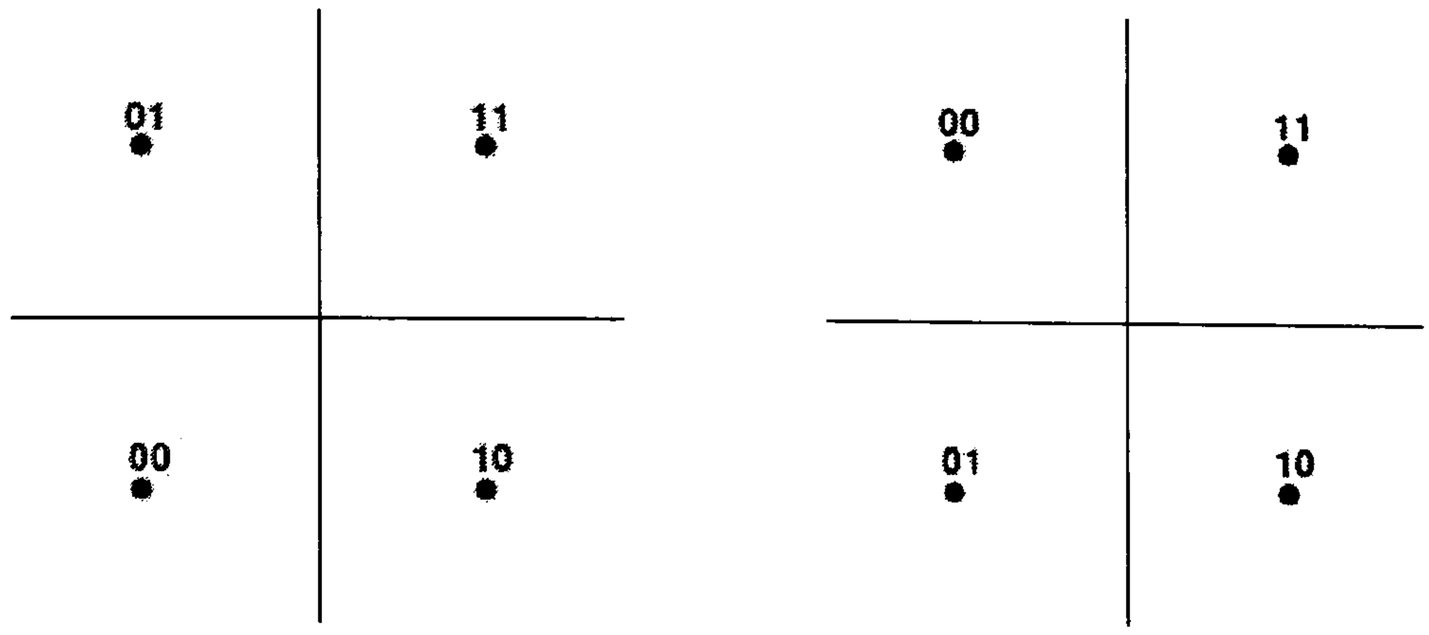

Figure 5.4: QPSK constellations mapping, left:gray, right:antigray

performance compared to the case when gray mapping is considered, but only after a certain number of iterations.

Gray and antigray mapping were chosen for the simulations, as the means to study the point previously indicated, on the joint demapper and source decoder. Fig. 5.4 show the gray and antigray constellation for QPSK.

The size of the frame for the simulation is 256 bits per frame. A pseudorandom interleavers of size 16x16 provides independence to the sequence of bits, at least for the first few iterations. Since no channel coding is considered here, there is no reason to increase the frame length, especially since the only reason behind this simulation verify the behaviour of the iterative demapper, and of the iterative source decoder. QPSK modulation was considered, and there the BER also represents the Source Symbol Error Rate (SSER). The Transition Probability (TP) matrix of the Markov source was varied according to eq. 5.12.

$$A = \begin{bmatrix} 0.85 & 0.15 \\ 0.15 & 0.85 \end{bmatrix} \tag{5.12}$$

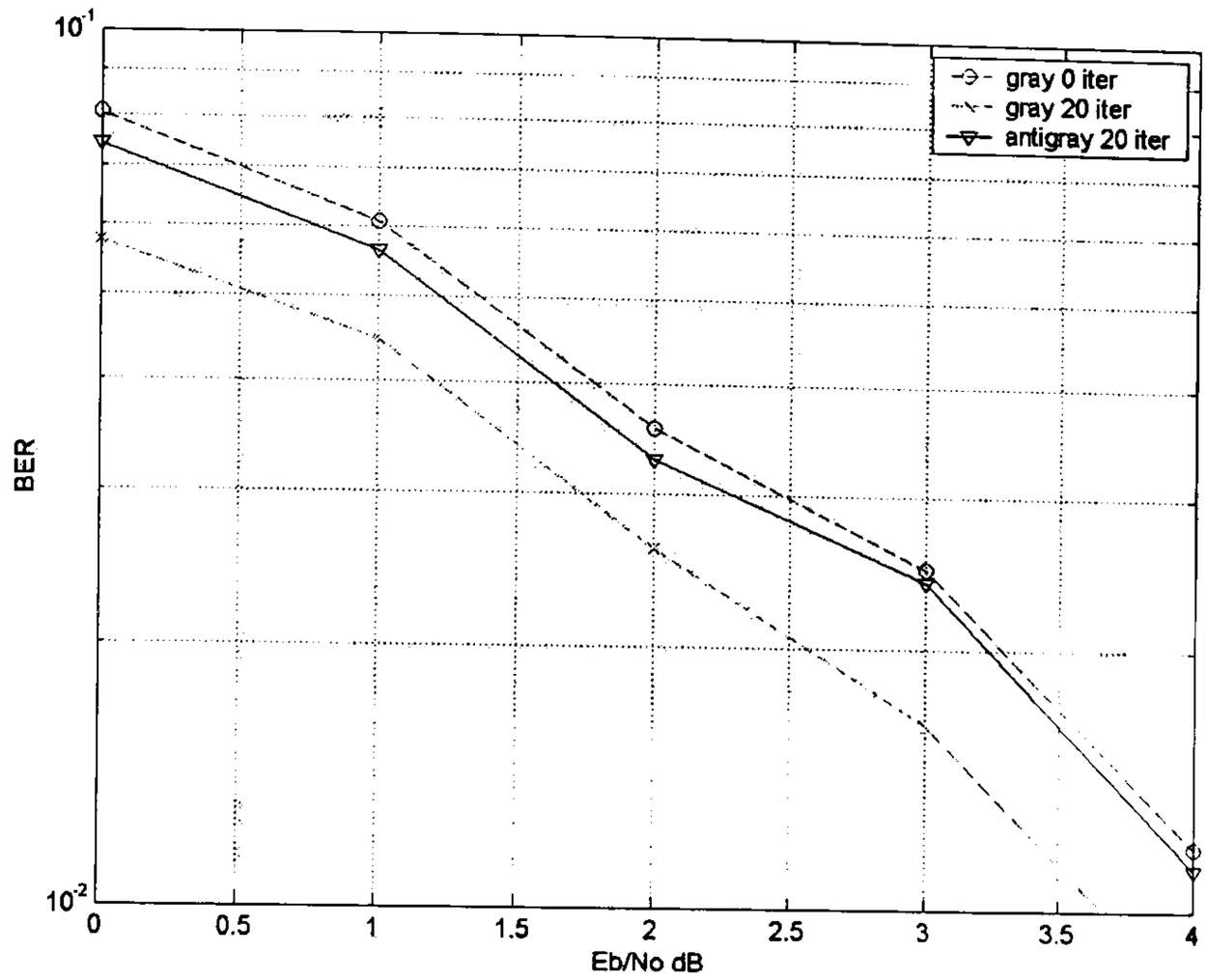

Figure 5.5: BER with length=256 QPSK modulation

To show the impact of the mapping on the BER performance, Fig. 5.5 and Fig. 5.6 characterise the BER and FER performance, respectively, when the length of the frame is 256, including gray and antigray mapping.

The graphs show that the gray mapping does not provide any improvement to the BER. The BER after 20 iterations is same as the BER after the first iteration; therefore, is only the source decoder that provides such improvement. For the antigray mapping, there is some improvement through the iterations, due to the reasons previously explained. This results match those ones previously published for a joint iterative demapper and iterative convolutional decoding.

The second stage of this analysis considers the incorporation of the GPS(1000,3,500)LDPC code, in a joint demapper, channel decoder and source decoder under the AWGN channel with gray and antigray modulation. The block diagram of

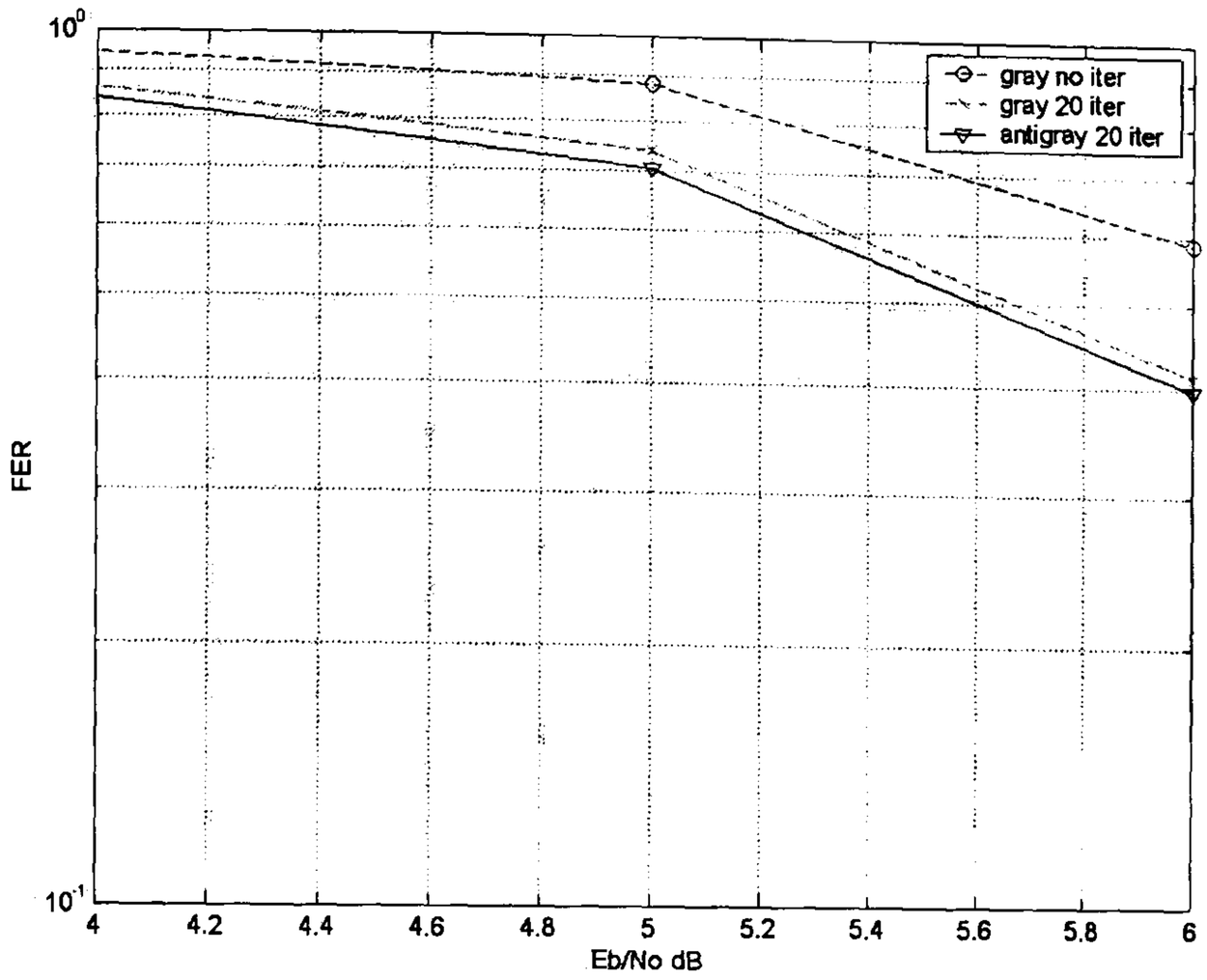

Figure 5.6: FER with length=264 QPSK modulation

this system is depicted in Fig. 5.7. For every iteration of the receiver, the log-likelihood values of the demapper, channel decoder and source decoder are calculated and exchanged in a sequencial manner.

To show the impact of the channel decoder, Fig. 5.8 and Fig. 5.9 characterise the BER and FER performance, including gray and antigray mapping. After 20 iterations, the BER and FER graphs show that the impact of the channel decoder is not considerably modified with the use of the extrinsic information provided by the demapper with antigray mapping, and therefore the system with gray mapping show the best performance.

The last stage of this analysis considers the incorporation of the GPS(240,3,120)LDPC code concatenated in parallel with the GPS(240,2,120) LDPC code, in a joint parallel concatenated channel decoder and source decoder under the AWGN channel with gray modulation. Antigray modulation is not considered since

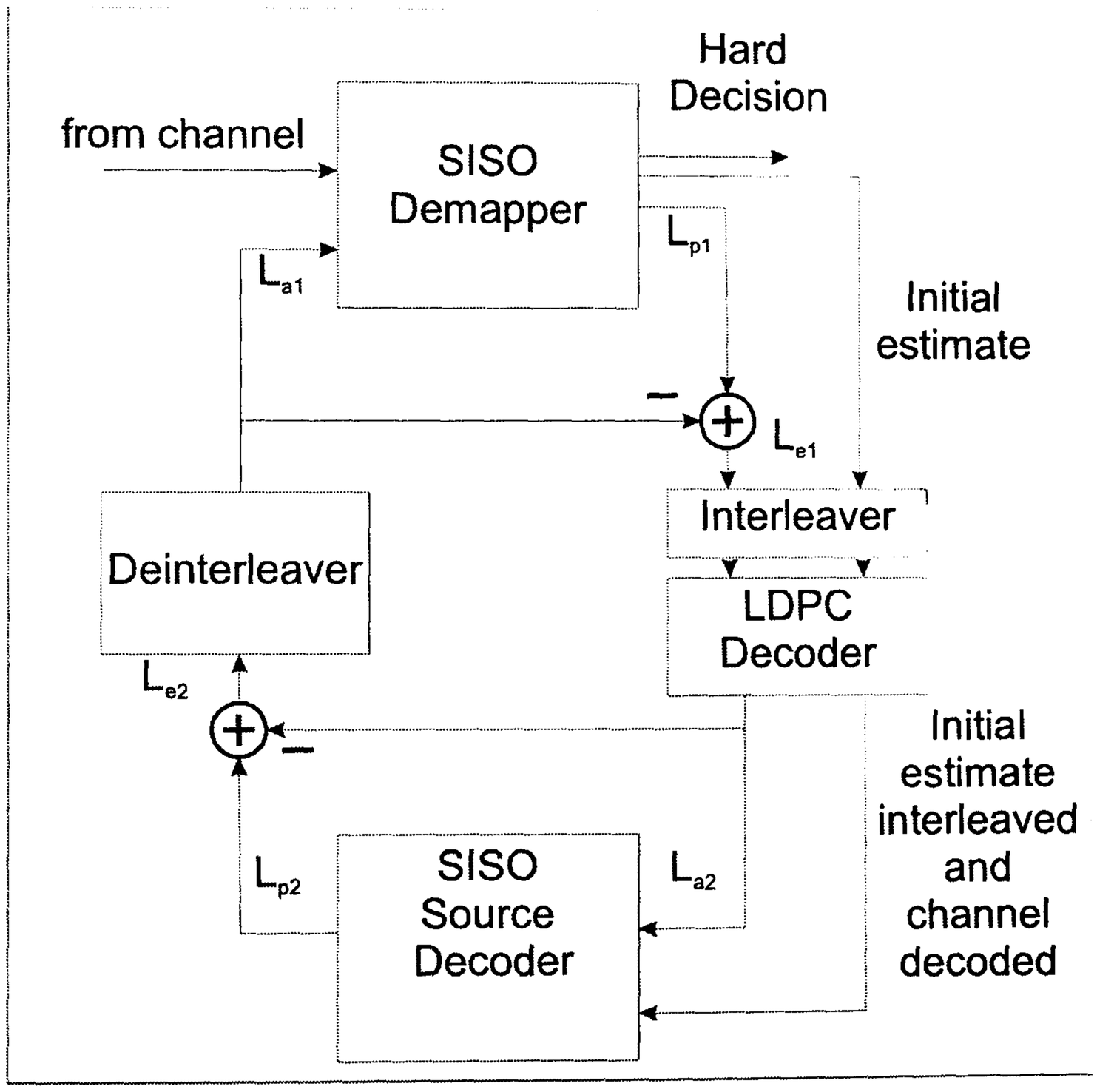

Figure 5.7: Block diagram for a joint demapping, channel decoding and source decoding scheme

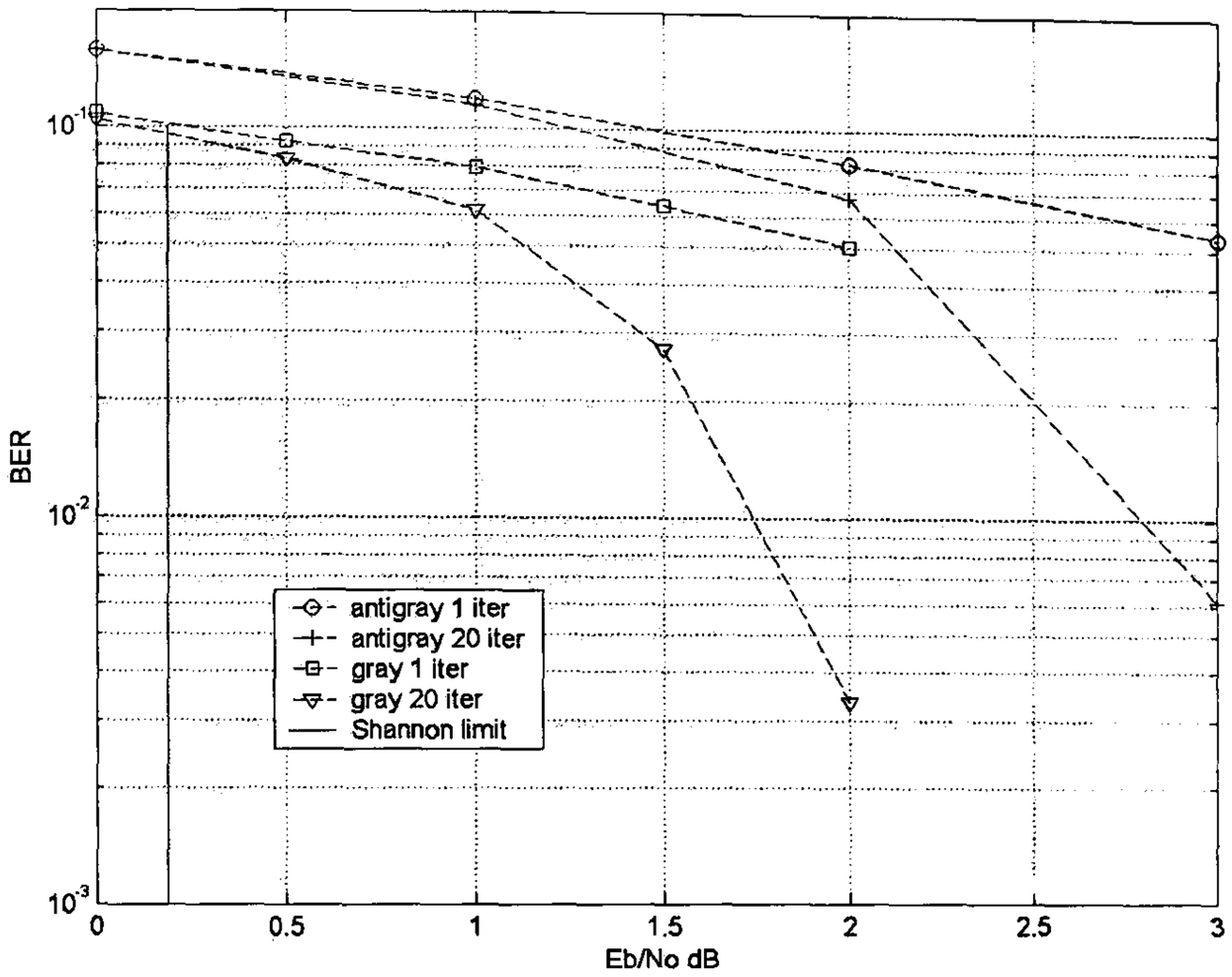

Figure 5.8: BER GPS(1000,3,500) QPSK modulation

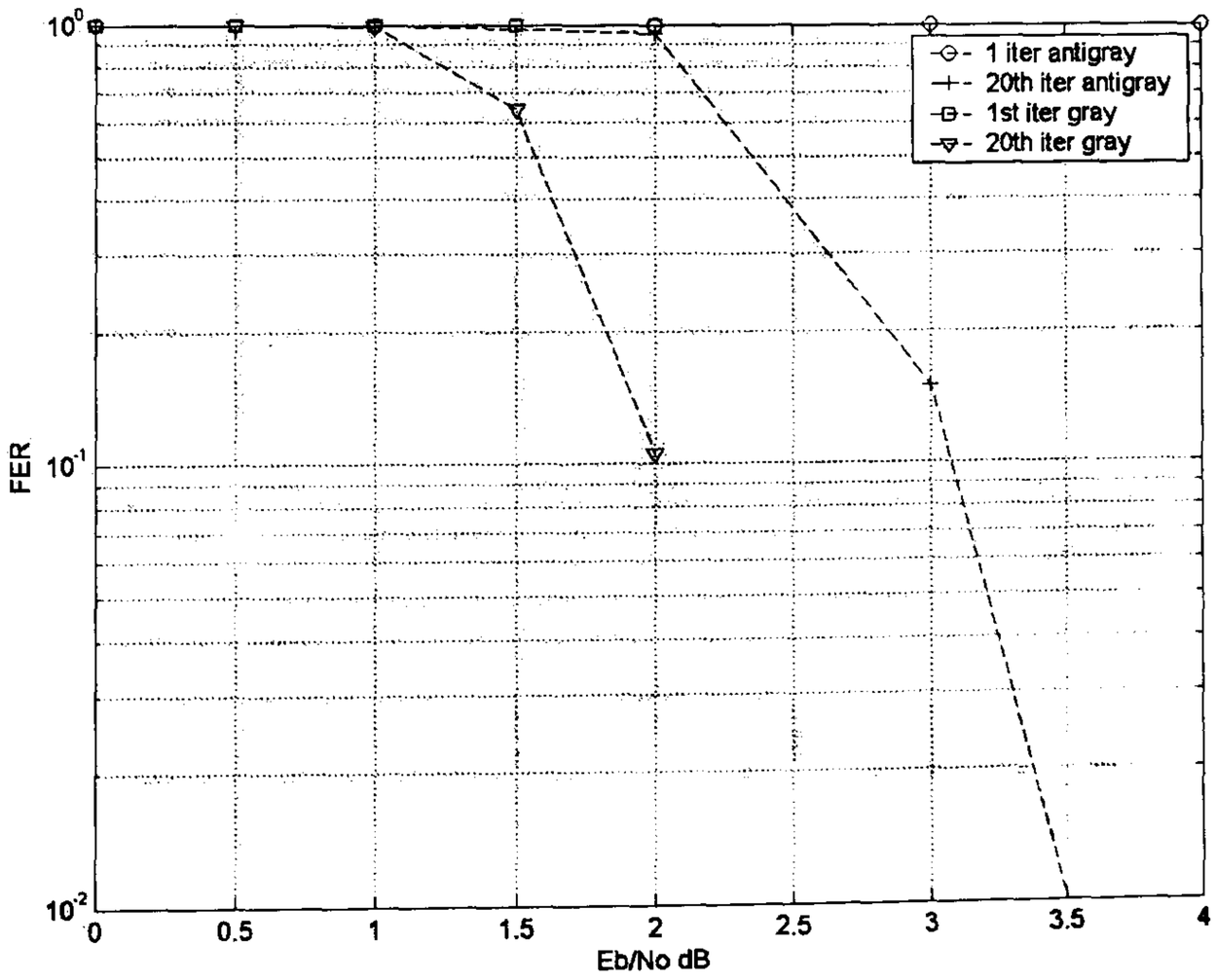

Figure 5.9: FER GPS(1000,3,500) QPSK modulation

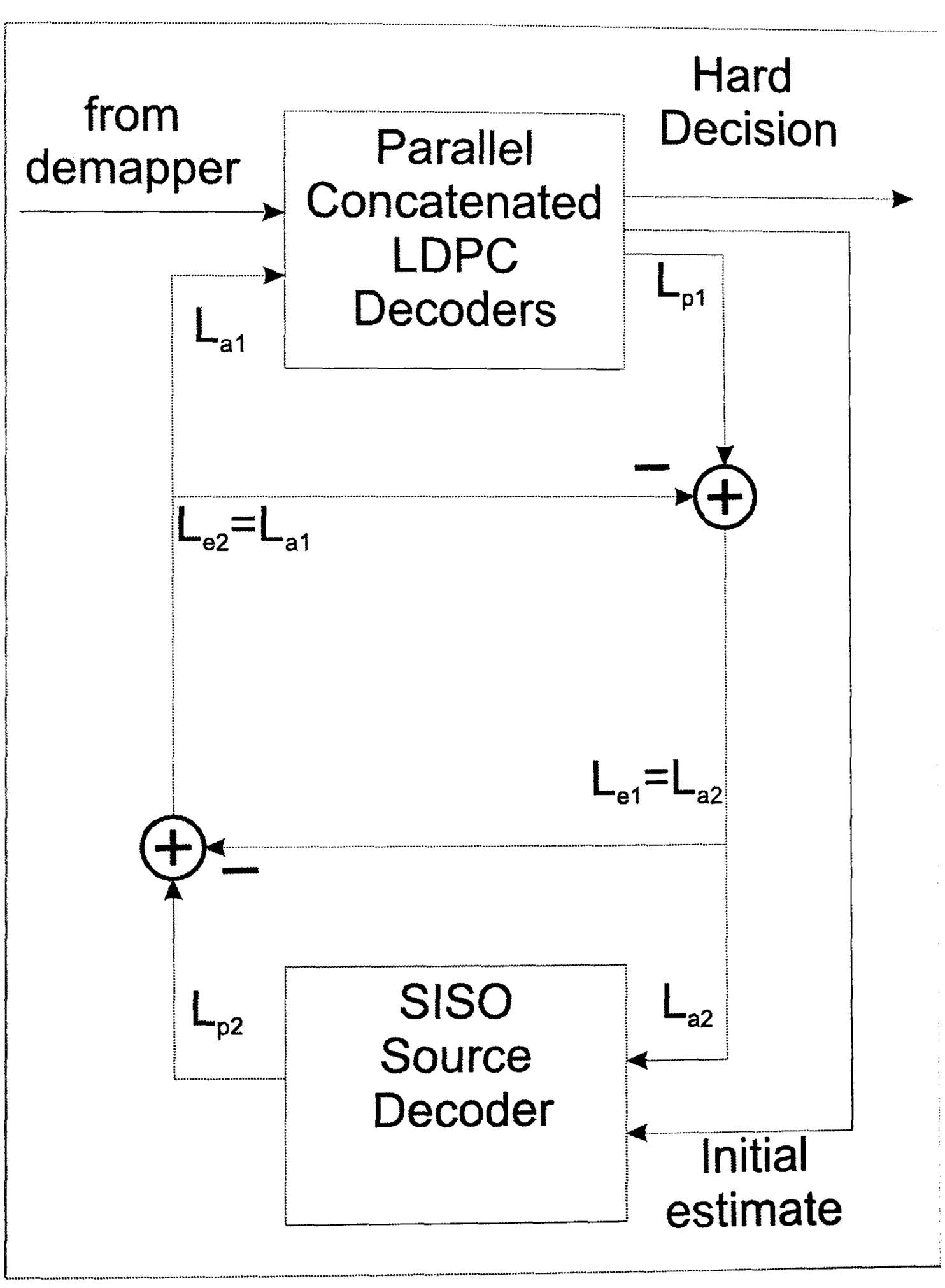

Figure 5.10: Block diagram for a joint demapping, channel decoding and source decoding scheme

it already proved to be ineffective when the performance of the single LDPC decoder was introduced. The BER and FER performance is shown in Fig. and Fig. respectively. Every 5 iterations between the parallel concatenated LDPC codes, the log-likelihood values of the demapper, nd source decoder are calculated and exchanged in a sequencial manner. A tota of 30 iterations were predefined. The overall performance of this scheme is poor compared to a system that does not exchange information with the source decoder. This can be proved, since the performance of the joint scheme improves as the redundancy of the source is decreased, meaning that the estimates from the source are sending incorrect likelihoods.

5.5 Summary

The analysis of the parallel concatenation of GPS LDPC codes with column weight 2 and 3, together with an iterative source decoder and an iterative demapper, has been characterised through simulations. The impact of the constellation used for the demapper do not show an improvement in the performance achieved by the LDPC decoder. On the contrary, the little redundancy sent back as likelihoods can reduce the overall performance. An adecuate weighting of the likelihoods could provide benefitial and should be analysed in future work. The impact of the source decoder is not considerable, but as the redundancy of the source is reduced, the overall performance improves. Once more, there seem to be an inadequate weighting of the likelihoods. This is proved after the last scheme performs five iterations for the concatenated decoder before exchanging information with the source decoder.

Chapter 6

Conclusion

In this thesis, a new procedure to construct Quasi-Cyclic LDPC codes with variable girth and variable length as design paramenters has been studied. Also, the parallel concatenation of structured LDPC codes has been characterised. Finally, the impact of the source decoder and the iterative demapper is revised, when used in a joint iterative receiver together with a parallel concatenated decoder. In this final chapter the conclusions of this research are presented and directions are proposed for future work.

6.1 Thesis Summary

The novel algorithm to construct Girth-Partition and Girth LDPC codes using the girth and the column weight of the code as design parameters, has been introduced. These codes are half rate QC-LDPC codes. The algorithm reduces the number of operations required to find LDPC codes with the constraints imposed for the design parameters selected, when compared to

similar QC-LPDC code constructions [18], [17], and [45]. The new GPS LDPC codes have similar performance to other random and structured LDPC codes over the AWGN channel and the flat Rayleigh fading channel. The flexibility of the algorithm to generate codes for a wide range of lengths is of great benefit, since that is not possible to achieve with other constructions based on finite geometries. The new algorithm has the flexibility to produce QC-LDPC codes with different girths, but with the same rate and and length.

The parallel concatenation of EG and PG LDPC codes shows limited performance improvement, when compared to the performance of the constituent parts, and therefore their application should be limited to systems that require low encoding complexity and low decoding complexity in terms of processing power, low memory assigned to contain the parity-check matrix, and short frames to reduce the delay. Since these codes achieve successful decoding in a very small number of iterations after a certain E_b/N_0 has been reached, it is desirable to combine one EG or PG LDPC code, with an LDPC code whose $\rho = 2$.

The novelty of this work relies on the proposed selection of LDPC codes to be concatenated in parallel, improving the characterisation of such codes previously published, and the analysis of their performance. This analysis is justified since they contain some of the *good* specific properties demanded by the EXIT charts to improve the overall BER and FER performance. Such analysis is unique to this work.

The analysis of the parallel concatenation of GPS LDPC codes with column weight 2 and 3, together with an iterative source decoder and an iterative demapper, has been characterised through simulations. The impact of

the constellation used for the demapper do not show an improvement in the performance achieved by the LDPC decoder, considering antigray demapping for QPSK modulation; this is not the case when the decoder is an iterative convolutional decoder. The little redundancy sent back by the iterative demapper with antigray mapping, as likelihoods, reduces the overall performance when the scheme considers an LDPC decoder. An adecuate weighting of the likelihoods could prove benefitial and should be analysed in future work. The impact of the source decoder is not considerable, but as the redundancy of the source is reduced, the overall performance improves. Once more, there seem to be an inadequate weighting of the likelihoods.

6.2 Statements of Originality

The main contributions of this work are:

1. The introduction of a new procedure to construct LDPC codes with girth and column weight as a design parameter.

2. The characterisation of the new ensemble of GPS LDPC codes.

3. The aditional characterisation of well structured LDPC codes included in previous chapters, my means of EXIT charts.

4. The characterisation of well structured LDPC codes when concatenated in parallel, over AWGN and flat Rayleigh fading channels.

5. The BER and FER performance characterisation for a joint iterative receiver with an interative demapper, and interative source decoder and

a parallel concatenated channel decoder using specific LDPC codes.

6.3 Suggested directions of Future Work

Based on EXIT charts, minor modifications to the algorithm presented to create GPS LDPC codes can provide new LDPC code ensembles, useful to improve the BER and FER performance.

The parallel concatenation of EG and PG LDPC codes with other LDPC codes whose column weight is two, should be analysed, since the EXIT charts indicate good BER and FER performance for such scheme.

The analysis of the joint iterative receiver with an interative demapper, and interative source decoder and a parallel concatenated channel decoder should be enhanced through DE techniques, including EXIT charts.

Based on the parallel concatenation of non-binary LDPC codes studied in [49] the characterisation of such scheme is still basic and further research is required.

The impact of lazy scheduling [46] for the sum-product decoding has to be further characterised for the different types of LDPC codes. The impact of such technique on parallel concatenated LDPC codes is still an open question.

In [103] an improved Max-Log-MAP algorithm is introduced for Turbo codes. BER performance curves demostrate that it achieves near to Log-MAP algorithm performance with a reduced complexity. The performance and characterisation through DE when applied to LDPC codes could provide good performance.

The analysis of parallel concatenated LDPC codes on multiuser detection

techniques, together with Hybrid ARQ needs to be addressed.

References

[1] T. L.Hanzo and B.L.Yeap, *Turbo coding, turbo Equalisation and Space-Time Coding for Transmission over Fading Channels*, Wiley and Sons, Eds. Wiley and Sons, 2002.

[2] A. G.Berrou and P.Thitimajshima, "Near shannon limit error-correcting coding: Turbo codes," *in Proc. Int. Conf. Comm.*, pp. 1064–1070, 1993.

[3] R. Gallager, "Low-density parity-check codes," Master's thesis, M.I.T. Press, Cambridge, 1963.

[4] N.Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linkping University, Sweden, 1996.

[5] H. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, pp. 28–41, 2004.

[6] R.M.Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, Sept 1981.

[7] A.P.Worthen and W.E.Stark, "Unified design of iterative receivers using factor graphs," *on Information Theory, IEEE Transactions*, vol. 47, pp. 843–849, Feb 2001.

[8] R. T.J.Richardson, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 599–618, Feb. 2001.

[9] T. S.Y.Chung and R.Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation," *Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation*, vol. 47, pp. 657–670, Feb. 2001.

[10] S.R.Kollu and H.Jafarkhani, "On the exit chart analysis of low-density parity-check codes," *IEEE Global Telecommunications Conference*, vol. 3, p. 6, 28 Nov - 2 Dec 2005.

[11] J. S.T.Brink and R.Yan, "Iterative demapping for qpsk modulation," *Electronics Letters*, vol. 34, pp. 1459–1460, July 1998.

[12] ——, "Iterative demapping and decoding for multilevel modulation," *IEEE Global Telecommunications Conference GLOBECOM*, vol. 1, pp. 579–584, Nov 1998.

[13] H. Behairy and S.-C.Chang, "Parallel concatenated gallager codes," *Electronics Letters*, vol. 36, pp. 2025–2026, Nov 2000.

[14] H.Behairy and S.C.Chang, "Parallel concatenated gallager codes for cdma applications," *IEEE Global Telecommunications Conference*, vol. 2, pp. 1002–1006, Nov. 2001.

[15] H. Behairy and S.C.Chang, "On the design, simulation and analysis of parallel concatenated gallager codes," *IEEE International conference on Communications ICC*, vol. 3, pp. 1850–1854, May 2002.

[16] H. Z. J.M.F. Moura and J. Lu, "Structured low-density parity-check codes," *IEEE Signal Proc. Magazine*, pp. 42–45, Jan 2004.

[17] H. Zhang and J. Moura, "The design of structured regular ldpc codes with large girth," in *IEEE GLOBECOM*, vol. 7, 2003, pp. 4022–4027.

[18] ——, "Geometry based designs of ldpc codes," in *IEEE International conference on Communications*, vol. 2, 2004, pp. 762–766.

[19] F. L.R.Bahl, J.Cocke and J.Raviv, "Optimum decoding of linear codes for minimizing symbol error rate," *IEEE Transactions Information theroy*, pp. 284–287, March 1974.

[20] S.Lin and D.J.Costello, *Error Control Coding*, K. Bradley, Ed. Pearson Prentice Hall, 2004.

[21] T. Richardson and M. Shokrollahi, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[22] D. R.J.McEliece and J.Cheng, "Turbo decoding as an instance of pearl's "belief propagation" algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 140–152, 1998.

[23] A. T.Etzion and A.Vardy, "Which codes have cycle-free tanner graphs," *IEEE Transactions on Information Theroy*, vol. 45, pp. 2173–2181, 1999.

[24] I. T. D.Changyan, D.Proietti and R.L.Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Transactions on Information theory*, vol. 48, pp. 1570–1579, June 2002.

[25] M.Schwartz and A.Vardy, "On the stopping distance and the stopping redundancy of codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 922–932, March 2006.

[26] a. K.-G. J.H.Weber, "Stopping set analysis for hamming codes," *IEEE Information Theory Workshop*, pp. 244–247, 2005.

[27] D.J.C.MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, pp. 399–432, Mar. 1999.

[28] I. L.Chen, J.Xu and S.Lin, "Near-shannon-limit quasi-cyclic low-density parity-check codes," *IEEE Transactions on Communications*, vol. 52, pp. 1038–1042, Jul. 2004.

[29] Y.-J. B.Ammar, B.Honary and S.Lin, "Construction of low-density parity-check codes based on balanced incomplete block designs," *IEEE Transactions on Information Theory*, vol. 50, pp. 1257–1268, Jun. 2004.

[30] K.-G. I.Djurdjevic, J.Xu and S.Lin, "Construction of low-density parity-check codes on reed-solomon codes with two information symbols," *IEEE Communications Letters*, vol. 7, pp. 317–319, Jul 2003.

[31] B.Vasic and O.Milenkovic, "Combinatorial construction of low-density parity-check codes for iterative decoding," *IEEE Transactions on Information Theory*, vol. 50, pp. 1156–1176, June 2004.

[32] Y.-S. H.Tang, J.Xu and K.Abdel-Ghaffar, "On algebraic construction of gallager and circulant low density parity check codes," *IEEE Transactions on Information Theory*, vol. 50, pp. 1269–1279, June 2004.

[33] I. L.Chen, L.Lan and K.Abdel-Ghaffar, "An algebraic method for constructiong quasi-cyclic ldpc codes," *Proceedings International Symposium Information Theory Application, Parma, Italy*, pp. 535–539, Oct. 2004.

[34] S. H.Tang, J.Xu and K.Abdel-Ghaffar, "Codes on finite geometries," *IEEE Transactions on Information Theory*, vol. 51, pp. 572–596, Feb. 2005.

[35] S.J.Johnson and S.R.Weller, "Construction of low-density parity-check codes from kirkman triple systems," *IEEE Global Telecommuniations Conference GLOBECOM*, vol. 2, pp. 970–974, Nov. 2001.

[36] B. B.Ammar and S.Lin, "Construction of low density parity check codes: a combinatoric design approach," *IEEE International Symposium on Information Theory, Proceedings*, p. 311, 2002.

[37] I. V. K.Jon-Lark, U.N.Peled and S.Friedland, "Explicit construction of families of ldpc codes with no 4-cycles," *IEEE Transactions on Information Theory*, vol. 50, pp. 2378–2388, Oct. 2004.

[38] P.Razaghi and W.Yu, "Bilayer ldpc codes for the relay channel," *IEEE International Conference on Communications*, vol. 4, pp. 1574–1579, June 2006.

[39] M.C.Davey and D.MacKay, "Low-density parity-check codes over gf(q)," *IEEE Communications Letters*, vol. 2, pp. 165–167, June 1998.

[40] K. B.M.Kurkoski and K.Kobayashi, "Density evolution for gf(q) ldpc codes via simplified message-passing sets," *Information Theory and Applications Workshop*, pp. 237–244, 2007.

[41] Z.Yang and L.Tong, "On the error exponent and the use of ldpc codes for cooperative sensor networks with misinformed nodes," *IEEE Transactions on Information Theory*, vol. 53, pp. 3265–3274, Sept 2007.

[42] A. S. A.Thangaraj, A.R.S.Dihidar and J-M.Merolla, "Applications of ldpc codes to the wiretap channel," *IEEE Transactions on Information Theory*, vol. 53, pp. 2933–2945, Aug 2007.

[43] S. Y.Kou and M.P.C.Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, pp. 2711–2736, 2001.

[44] G.A.Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, pp. 71–78, 1982.

[45] H. Zhang and J. Moura, "Large-girth ldpc codes based on graphical models," in *SPAWC*, 2003, pp. 100–104.

[46] E. D.Levin and S.Litsyn, "Lazy scheduling for ldpc decoding," *IEEE Communications Letters*, vol. 11, pp. 70–72, Jan 2007.

[47] S. E.Sharon and J.Goldberger, "Efficient serial message-passing schedules for ldpc decoding," *IEEE Transactions on Information Theory*, vol. 53, pp. 4076–4091, Nov. 2007.

[48] X.Wei and A.N.Akansu, "Density evolution for low-density parity-check codes under max-log-map decoding," *Electronics Letters*, vol. 37, pp. 1125–1126, 30 Aug. 2001.

[49] V.Rathi and R.Urbanke, "Density evolution, thresholds and the stability condition for non-binary ldpc codes," *IEE Procedings on Communications*, vol. 152, pp. 1069–1074, Dec. 2005.

[50] S. Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, pp. 1727–1737, Oct. 2001.

[51] P. I.Land and J.Sayir, "Bounds on information combining for the accumulator of repeat-accumulate codes without gaussian assumption," *Proceedings, International Symposium on Information Theory*, p. 443, 27 June-2 July 2004.

[52] S. I.Land and J.B.Huber, "Bounds on information combining," *IEEE Transactions on Information Theory*, vol. 51, pp. 612–619, Feb 2005.

[53] G. S.T.Brink and A.Ashikhmin., "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions of Communications*, vol. 52, pp. 670–678, April 2004.

[54] J. Garcia-Frias, "Joint source-channel decoding of correlated sources over noisy channels," *Proceedings Data Compression Conference*, pp. 283–292, March 2001.

[55] J. G.Buch, F.Burkert and B.Kukla, "To compress or not to compress?" *Global Communications Conference BLOBECOM*, pp. 198–203, Nov 1996.

[56] N. Grtz, "Iterative source-channel decoding using soft-in/soft-out decoders," *Proceedings, IEEE International Symposium on Information Theory*, p. 173, June 2000.

[57] J.Hagenauer, "Source-controlled channel decoding," *IEEE Transactions on Communications*, vol. 43, pp. 2449–2457, Sept. 1995.

[58] M. a. R. P.Strauch, C.Luschi, "Low complexity source controlled channel decoding in a gsm system," *IEEE Proceedings on Acoustics, Spech and Signal Processing ICASSP*, vol. 5, pp. 2571–2574, March 1999.

[59] Q.Chen and K.P.Subbalakshmi, "An integrated joint source-channel decoder for mpeg-4 coded video," *IEEE 58th Vehicular Technology Conference*, vol. 1, pp. 347–351, Oct 2003.

[60] A. H.Persson and T.Ottosson, "Utilizing soft information in image decoding," *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications 2003 PIMRC*, vol. 3, pp. 2678–2682, Sept. 2003.

[61] J. L.Yun and Y.Wu, "Ldpc-based joint source-channel coding scheme for multimedia communications," *The 8th International Conference on Communications Systems, ICCS*, vol. 1, pp. 337–341, Nov. 2002.

[62] B. Sklar, *Digital Communications Fundamentals and Applications*, R. Kernan, Ed. B. Goodwin, 2001.

[63] N. J.B.Anderson, T.Eriksson, "On the bcjr algorithm for rate distortion source coding," *IEEE Transactions on Information Theory*, vol. 53, pp. 3201–3207, Sept 2007.

[64] C.Giuseppe and M.Fresia, "Joint source-channel coding: a practical approach and an implementation example," *Information Theory and Applications Workshop*, pp. 63–72, 2007.

[65] D.A.Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, pp. pp. 854–867, Sept. 1952.

[66] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, A.-W. Longman, Ed. Addison-Wesley Publishing Company, 1994.

[67] M.Singh and I.J.Wassell, "Comparison between serial and parallel concatenated channel coding schemes using continuous phase modulation over awgn and fading channels," in *CIC' 2001*, 2001.

[68] M. P. R. T.Souvignier, A.Friedmann and J.K.Wolf, "Turbo decoding for pr4: parallel versus serial concatenation," *IEEE International Conference on Communications (ICC)*, vol. 3, pp. 1638–1642, 1999.

[69] H. Behairy and S.-C.Chang, "Analysis and design of parallel concatenated gallager codes," *Electronics letters*, vol. 38, pp. 1039–1040, Aug. 2002.

[70] J.Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, M. Kaufmann, Ed. San Mateo, CA, 1988.

[71] B. F.R.Kschischang and H.A.Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–512, Feb 2001.

[72] D.J.C.MacKay and R.M.Neal, "Near shannon limit performance of low-density parity-check codes," *Electronic Letters*, vol. 32, pp. 1645–1646, Aug. 1996.

[73] T. D.Sridhara and R.M.Tanner, "Low density parity check matrices from permutation matrices," *Proceedings of 2001 Conference on Infor-*

mation Sciences and systems, John Hopkins University, Baltimore, p. 142, March 2001.

[74] R.M.Tanner, "Spectral graphs for quasi-cyclic ldpc codes," *IEEE International Symposium on Information Theory, Washington, DC*, p. 226, Jun 2001.

[75] J. S. L.Chen, I.Djurdjevic and K.Abdel-Ghaffar, "Construction of qc-ldpc codes based on the minimum-weight codewords of rs codes," *Proceedings IEEE International Symposium on Information Theory*, p. 239, June-July 2004.

[76] L. L. J.Xu, L.Chen and S.Lin, "Construction of low-density parity-check codes by superposition," *IEEE Transactions on Communications*, vol. 53, pp. 243–251, Feb. 2005.

[77] R.M.Tanner, "A transform theory for a class of group-invariant codes," *IEEE Transactions on Information theory*, vol. 34, pp. 725–775, July 1988.

[78] L. B. L.Lan, T.Ying-Yu and B.Honary, "New constructions of quasi-cyclic ldpc codes based on special classes of bidb for the awgn and binary erasure channels," *IEEE Transactions on Communications*, vol. 56, pp. 39–48, Jan. 2008.

[79] A. B.Honary and B.Ammar, "Construction of well-structured quasi-cyclic low-density parity check codes," *IEE Proceedings - Communications*, vol. 152, pp. 1081–1085, Dec. 2005.

[80] G. S.Honary, N.Pandya and B.Honary, "Migration to capacity approaching codes for digital video broadcasting," *IEE Procedings - Communications*, vol. 152, pp. 1103–1107, Dec. 2005.

[81] H. S. Y.Kou, J.Xu and K.Abdel-Ghafar, "On circulant low density parity check codes," *IEEE International symposium on Information Theory*, p. 200, 2002.

[82] M.P.C.Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Transactions on Information Theory*, vol. 50, pp. 1788–1793, Aug. 2004.

[83] W.W.Peterson and E.J.Weldon, *Error Correcting Codes*, M. Cambridge, Ed. MIT Press, 1972.

[84] B. S.Tong, S.Zhang and X.Wang, "Fast encodable and decodable irregular repeat accumulate codes from circulant permutation matrices," *IEEE Electronics Letters*, vol. 43, pp. 48–49, Jan 2007.

[85] C. W.Zhongfeng and K.K.Parhi, "Area efficient decoding of quasi-cyclic low density parity check codes," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 49–52, 2004.

[86] W.Zhongfeng and J.Qing-wei, "Low complexity, high speed decoder architecture for quasi-cyclic ldpc codes," *IEEE International Symposium on Circuits and Systems*, vol. 6, pp. 5786–5789, May 2005.

[87] Z.Wang and Z.Cui, "Low-complexity high-speed decoder design for quasi-cyclic ldpc codes," *IEEE Transactions on Very Large Scale Integrtion Systems*, vol. 15, pp. 104–114, Jan. 2007.

[88] J.Rosenthal and P.O.Vontobel, "Constructions of ldpc codes using ramanujan graphs and ideas from margulis," *Proceedings of 38th Allerton conference on Communication, Control and Computing*, vol. 12, pp. 248–257, Oct. 2000.

[89] Y.Lin and W.E.Ryan, "Design of ldpc-coded modulation schemes," *Proceedings 3rd International Turbo Symposium*, pp. 551–552, Sep 2003.

[90] L. S. Z.Li, L.Chen and W.Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Transactions on Communications*, vol. 54, pp. 71–81, Jan 2006.

[91] J.C.Serrato and T.O'Farrell, "Parallel concatenated gallager codes using euclidean and projective geomery ldpc codes," in *The Annual London Conference on Communications, UCL*, 13-14 Sept 2004.

[92] ——, "Structured parallel concatenated ldpc codes," in *The Annual London Conference on Communications, UCL*, 14-15 Sept. 2006.

[93] J. G.D.Forney, *Concatenated Codes.* MIT Press, Cambridge, 1966.

[94] J.C.Serrato, "Girth partition and shift," *Electronic Letters*, Under revision.

[95] S. Brink, "Convergence of iterative decoding," *Electronics letters*, vol. 35, pp. 806–808, May 1999.

[96] J.Hu and W.W.Lu, "Open wireless architecture - the core to 4g mobile communications," *Proceedings of ICCT*, pp. 1337–1342, 2003.

[97] W.W.Lu, "4g mobile research in asia," *IEEE Communications Magazine*, pp. 104–106, March 2003.

[98] S. S.Ryoo and C.S.Ahn, "Rate adaptation with hybrid arq based on cross layer information for satellite communication systems," *IEEE International Symposium on Consumer Electronics*, pp. 165–168, Sept 2004.

[99] B. M.Assaad and D.Zeghlache, *TCP Performance over UMTS-HSDPA System.* Springer Netherlands, Jan. 2005.

[100] J.C.Serrato and T.O'Farrell, "Joint demapping and source decoding for multilevel modulation," in *IEEE Wireless Communications and Networking Conference*, 3-6 April 2006.

[101] R. K. I. Kozintsev and K. Ramchandran, "A factor graph framework for joint source-channel decoding of images," *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systemas, and Computers*, vol. 1, pp. 321–325, Oct 1999.

[102] T.C.Hewavithana and M. Brookes, "Soft decisions for dqpsk demodulation for the vitervi decoding of the convolutional codes," *IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), vol. 4, pp. 17–20, April 2003.

[103] B. S.Talakoub, L. Sabeti and M.Ahmadi, "An improved max-log-map algorithm for turbo decoding and turbo equalization," *IEEE transactions on Instrumentation and Measurement*, vol. 56, pp. 1058–1063, june 2007.