UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Dipartimento di Elettronica Informatica e Sistemistica

Dottorato di Ricerca in Ingegneria Elettronica, Informatica e delle Telecomunicazioni - XIX Ciclo

SSD: ING-INF/03 - Telecomunicazioni

Iterative Decoding Methods Based on Low-Density Graphs

(Metodi di Decodifica Iterativa Basati su Grafi Sparsi)

Tesi di:
Ing. Enrico Paolini

Relatore:
Chiar.mo Prof. Ing. Marco Chiani

Coordinatore:
Chiar.mo Prof. Ing. Paolo Bassi

Anno Accademico 2006/2007

To my grandparents,
Bice, Enzo, Silvana and Paolino

Contents

List of Tables

List of Figures

Acronyms

APP *a posteriori* probability

Bi-AWGN binary-input additive white Gaussian noise

BEC binary erasure channel

BP belief propagation

BSC binary symmetric channel

CCSDS Consultative Committee for Space Data Systems

CLBuEC constant length burst erasure channel

CND check node decoder

DE differential evolution

D-GLDPC doubly-generalized LDPC

eIRA Extended irregular repeat-accumulate

ESA European Space Agency

ESOC European Space Operation Center

EXIT extrinsic information transfer

GeIRA generalized extended irregular repeat-accumulate

GLDPC generalized low-density parity-check

i.i.d. independent, identically distributed

IRA irregular repeat-accumulate

LDPC low-density parity-check

LR likelihood ratio

MAP	maximum a posteriori
NASA-JPL	NASA Jet Propulsion Lab
SPC	single parity-check
VND	variable node decoder

Introduction

This thesis investigates channel coding schemes that can be represented as low-density bipartite graphs, and which are decoded through iterative algorithms based on the *belief-propagation* (BP) principle [1]. By definition, a bipartite graph is any graph whose nodes are partitioned into two disjoined subsets, such that any edge can connect only two nodes belonging to different subsets [2]. The number of edges connected to a node is said the node *degree*. For each of the two disjoint subsets, the sequence of node fractions associated to the node degrees is said the (node-oriented) *degree distribution*. If the fraction of edges towards the nodes of a certain degree is used, instead of the fraction of nodes with that degree, then the edge-oriented degree distribution is obtained. By definition, a *cycle* of length l is a path in the bipartite graph, composed of l edges, starting and ending on the same node. The graph *girth* is length of the shortest cycle in the graph. From the point of view of BP decoding, bipartite graphs with large girth are preferable. The bipartite graph representing the code is sparse in that the number of edges in the bipartite graph is a relatively small fraction of the total number of possible edges connecting the nodes of different subsets. The sparseness of the bipartite graph is the key feature for developing iterative decoding algorithms whose complexity is manageable even for very long codeword lengths (up to thousands of bits).

The so far most investigated class of such codes is represented by low-density parity-check (LDPC) codes, that were first proposed in [3] in 1963, and then rediscovered only about forty years later (see e.g. [4, 5, 6, 7, 8]), after the invention of turbo codes [9]. For LDPC codes, the two disjoint sets of nodes are called *variable nodes* and *check nodes*, where the variable nodes represent the encoded bits and the check nodes the parity-check equations involving such bits. Thus, a variable node V is connected to a check node C if and only if the bit corresponding to V is involved in the parity-check equation corresponding to C. LDPC codes have been shown to exhibit excellent performance, under iterative BP decoding, over a wide range of communication channels, achieving extraordinary coding gains with moderate decoding complexity.

Asymptotically in the codeword length, LDPC codes exhibit a threshold phenomenon. In fact, if the noise level is smaller than a certain *decoding threshold* (which depends on the bipartite graph properties) then it is possible to achieve an arbitrarily small bit error probability under iterative decoding, as the codeword length tends to infinity. On the contrary, for noise level larger than the threshold, the bit error probability is always larger than a positive constant, for any codeword length [7, 8]. On the binary-input additive white gaussian noise (Bi-AWGN) channel, this threshold value is defined in terms of signal-to-noise ratio, on the binary symmetric channel (BSC) in terms of error probability, on the binary erasure channel (BEC) in terms of erasure probability. There are two main tools for asymptotic *analysis* of LDPC codes, i.e. for evaluating the decoding threshold associated to a given degree distribution: differential evolution [7] and extrinsic information transfer (EXIT) charts [10].

One of the features that makes LDPC codes very attractive is the possibility to design, for several transmission channels, the bipartite graph degree distribution for the variable and check nodes in order to obtain a decoding threshold which is extremely close to the channel capacity [11]. For given code rate and node degrees, the threshold optimization is usually performed by means of numerical optimization tools, especially differential evolution (DE) [12]. In the special case of the BEC, where the transmitted bits are either correctly received or “lost” independently with some erasure probability, it was also shown that it is possible to design sequences of degree distributions, known as *capacity-achieving sequences* [13] whose threshold converges to the channel capacity.

While the asymptotic design and analysis of LDPC codes is so far mostly understood, there still is a knowledge gap with respect to the design of *finite length* LDPC codes. In fact, it has not be possible up to date to design finite length LDPC codes having both very good waterfall performance and very good error floor performance: finite length LDPC codes with a degree distribution associated to a close-to-capacity decoding threshold, though characterized by very good waterfall performance, usually exhibit a bad error floor performance, due to poor minimum distance [14, 15]. Then, the design of finite length LDPC codes mostly relies on the search for the best compromise between the two regions of the performance curve, by carefully constructing the bipartite graph. Among the several techniques that have been proposed they are quite effective those based on finite geometries [5], on the progressive-edge-growth (PEG) construction [16], on the irregular repeat-accumulate (IRA) construction [17], on circulant permutation matrices [18] and on protographs [19]. These techniques, or their combinations, lead to codes with good code properties (in terms, for instance, of girth of the bipartite graph and possibility to perform the encoding procedure efficiently).

The attempt to improve the compromise between waterfall performance and error floor has recently inspired the study of more powerful, and somewhat more complex, coding schemes. This is the case of generalized LDPC (GLDPC) codes [20] and doubly-generalized LDPC (D-GLDPC) codes [21] that, together with standard LDPC codes, are the coding techniques subject of this thesis. The basic idea behind GLDPC and D-

GLDPC codes is to bridge the classical coding techniques and the modern ones (based on sparse graphs and BP decoding), by letting classical linear block codes be nodes of the bipartite graph. In GLDPC codes, generic linear block codes are used as check nodes. In D-GLDPC codes, which represent a wider class of codes than GLDPC codes, the generalization consists in using generic linear block codes both as variable nodes and as check nodes.

The above mentioned coding techniques are studied in this thesis assuming as communication channel the BEC. This communication channel has gained a certain interest in the last few years for a number of reasons. First, when considering transmission on the BEC, it is sometimes possible to study the problems analytically instead of by extensive simulation. Second, it was proved in recent works that LDPC codes optimized for transmission on the BEC usually exhibit good performance also when used for transmission on error channels, like the BSC or the Bi-AWGN channel [22]. In general, codes having a good compromise between waterfall and error floor performance on the BEC, exhibit the same good compromize also on other channels. Third, when performing asymptotic code analysis by EXIT chart, it is sometimes possible to use the EXIT function for the check nodes, evaluated on the BEC, instead of the actual EXIT function, and still obtain an accurate threshold analysis. Fourth, codes designed for transmission on erasure channels (especially burst erasure channels) are interesting for a wide range of heterogeneous applications, including wireless communications, data storage and packet-level forward erasure correction in space applications (this specific application is currently under investigation within the CCSDS by ESA/ESOC and NASA-JPL [23]). The thesis outline is described next.

In Chapter 1, the coding techniques investigated in the thesis, namely LDPC, GLDPC and D-GLDPC codes, are described, and the basic notation is introduced. In this chapter, the rules for developing the parity-check matrix of GLDPC codes and D-GLDPC codes from the code bipartite graph, and the decoding algorithm for such codes on the BEC are also described.

In Chapter 2, a method for the analysis of GLDPC codes and D-GLDPC codes on the BEC, based on the concept of extrinsic information transfer (EXIT) chart, is described. This method permits to overcome the problem consisting in the impossibility to evaluate the EXIT function for the check or variable component codes, in situations where the information functions or split information functions for the component code are unknown. According to the proposed technique, GLDPC codes and D-GLDPC codes where the generalized check and variable component codes are *random* codes with minimum distance at least 2, are considered. A method is then developed which permits to obtain the EXIT chart for the overall GLDPC or D-GLDPC code, by evaluating the expected EXIT function for each check and variable component code. This technique is then combined with differential evolution (DE) algorithm in order to generate some optimal GLDPC and D-GLDPC edge distributions. Numerical results on long, random codes, are presented which reveal that D-GLDPC codes can outperform standard LDPC codes and GLDPC codes in terms of both waterfall performance and error floor.

In Chapter 3, the well-known *stability condition* for LDPC codes on the BEC is extended to GLDPC codes and D-GLDPC codes. It is proved that, in both cases, the stability condition only involves the component codes with minimum distance 2. The stability condition for GLDPC codes is always expressed as an upper bound to the decoding threshold. This is not possible for D-GLDPC codes, unless all the generalized variable nodes have minimum distance at least 3. Furthermore, a condition called *derivative matching* is defined, which is sufficient for a GLDPC or D-GLDPC code to achieve the stability condition with equality. If this condition is satisfied, the threshold of D-GLDPC codes (whose generalized variable nodes have all minimum distance at least 3) and GLDPC codes can be expressed in closed form.

A family of LDPC degree distributions, whose decoding threshold on the binary erasure channel (BEC) admits a simple closed form, is studied in Chapter 4. These degree distributions are a subset of the check regular distributions (i.e. all the check nodes have the same degree) fulfilling the derivative matching condition, and are referred to as p-positive distributions. Achieving the stability condition with equality, the threshold of a p-positive distribution is simply expressed by $[\lambda'(0)\rho'(1)]^{-1}$. Besides this closed form threshold expression, the p-positive distributions exhibit three additional properties. First, for given code rate, check degree and maximum variable degree, they are in some cases characterized by a threshold which is extremely close to that of the best known check regular distributions, under the same set of constraints. Second, the threshold optimization problem within the p-positive class can be solved in some cases with analytic methods, without using any numerical optimization tool. This is indeed the most interesting feature of this class of codes. Third, these distributions can achieve the BEC capacity. The last property is shown by proving that the well-known binomial degree distributions belong to the p-positive family.

Chapter 5 moves to consider finite length LDPC codes and their design over non-memory-less erasure channels. In this chapter, a simple, general-purpose and effective tool for the design of LDPC codes for iterative correction of bursts of erasures is presented. The design method consists in starting from the parity-check matrix of an LDPC code and developing an optimized parity-check matrix, with the same performance on the memory-less erasure channel, and suitable also for the iterative correction of single bursts of erasures. The parity-check matrix optimization is performed by an algorithm called *pivot searching and swapping* (PPS) algorithm, which executes permutations of carefully chosen columns of the parity-check matrix, after a local analysis of particular variable nodes called stopping set pivots. This algorithm can be in principle applied to any LDPC code. If the input parity-check matrix is designed for achieving good performance on the memory-less erasure channel, then the code obtained after the application of the PSS algorithm provides good joint correction of independent erasures and single erasure bursts. Numerical results are provided in order to show the effectiveness of the PSS algorithm when applied to different categories of LDPC codes.

Concluding remarks are presented in Chapter 6.

Chapter 1

Generalizing Low-Density Parity-Check Codes: GLDPC and D-GLDPC Codes

1.1 Introduction

Low-density parity-check (LDPC) codes [3] have been shown to exhibit excellent asymptotic performance over a wide range of channels, under iterative decoding [7, 11, 5, 24, 18, 25]. It has been proved that irregular LDPC codes are able to asymptotically achieve the binary erasure channel (BEC) capacity for any code rate [26, 13]: this means that, for any code rate R and for any small $\epsilon > 0$, it is possible to design an edge degree distribution (λ, ρ) such that $\int_0^1 \rho(x)\mathrm{d}x / \int_0^1 \lambda(x)\mathrm{d}x = 1 - R$, whose threshold is $p^* = (1-\epsilon)(1-R)$. Examples of capacity achieving (sequences of) degree distributions are the heavy-tail poisson sequence [26] and the binomial sequence [27].

It is well known that this very good asymptotic performance in terms of decoding threshold does not necessarily correspond to a satisfying finite length performance. In fact, finite length LDPC codes with good asymptotic threshold, though typically characterized by good waterfall performance, are usually affected by high error floors [14, 28, 29]. This phenomenon has been partly addressed in [15], where it is proved that all the so far known capacity approaching LDPC degree distributions, all characterized by $\lambda'(0)\rho'(1) > 1$, are associated to finite length LDPC codes whose minimum distance is a sublinear (more precisely, logarithmic) function of the codeword length N. When considering transmission on the BEC, the low weight codewords induce small stopping sets [30], thus resulting in high error floors.

The (so far not overcome) inability in generating LDPC codes, with threshold close to capacity and good minimum distance properties as well, is one of the main reasons for investigating more powerful (and complex) coding schemes. Examples are generalized LDPC (GLDPC) codes and doubly-generalized LDPC (D-GLDPC) codes. In GLDPC codes, generic linear block codes are used as check nodes in addition to the traditional single parity-check (SPC) codes, as explained in Section 1.2. First introduced in [20], GLDPC codes have been more recently investigated, for instance, in [31, 32, 33, 34, 35, 36, 37, 38, 39, 40]. Recently introduced in [21, 41, 42], D-GLDPC codes represent a wider class of codes than GLDPC codes. The generalization consists in using generic linear block codes as variable nodes in addition to the traditional repetition codes (see Section 1.2). Linear block codes used as check or variable nodes will be called *component codes* of the D-GLDPC code. The check nodes represented by component codes which are not SPC codes will be called *generalized check nodes*, and their associated codes *generalized check component codes*. Analogously, the variable nodes represented by component codes which are not repetition codes will be called *generalized variable nodes*, and their associated codes *generalized variable component codes*. The ensemble of all the check nodes will be referred to as the check node decoder (CND), and the ensemble of all the variable node variable node decoder (VND). In this thesis, only binary check and variable component codes will be considered, so that the overall LDPC, GLDPC or D-GLDPC code is a binary code.

A description of the concepts of GLDPC and D-GLDPC code is provided in Section 1.2. In Section 1.3 the rule for obtaining the parity-check matrix of a D-GLDPC code is explained. Then, in Section 1.4 the decoding algorithm for D-GLDPC codes on the BEC is described. Both Section 1.3 and Section 1.4 include LDPC and GLDPC codes as special instances. Finally, in Section 1.5 it is provided evidence that the structure and the properties of a D-GLDPC code depend on the representation chosen for its generalized variable nodes.

1.2 LDPC codes, GLDPC codes and D-GLDPC codes

A traditional LDPC code of length N and dimension K is usually graphically represented by means of a sparse bipartite graph, known as Tanner graph [20], characterized by N *variable nodes* and a number $M \geq N - K$ of *check nodes*. Each edge in the graph can only connect a variable node to a check node, and the number of edges is small with respect to the total number of possible edges $M \cdot N$. According to this representation, the variable nodes have a one-to-one correspondence with the encoded bits of the codeword, and each check node represents a parity-check equation involving a certain number of encoded bits. The degree of a node is defined as the number of edges connected to the node. Thus, the degree of an LDPC variable node is the number of parity constraints the corresponding encoded bit is involved in, and the degree of an LDPC check node is the number of bits involved in the corresponding parity-check equation.

A degree-n check node of a standard LDPC code can be interpreted as a length-n

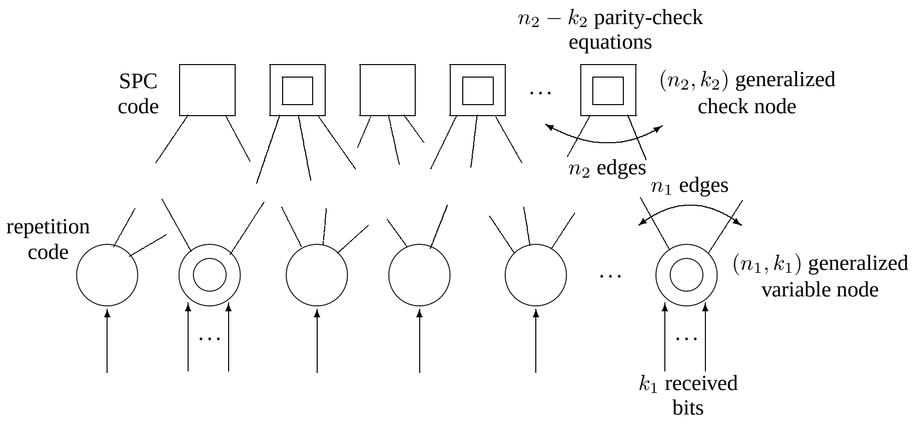

$n_2 - k_2$ parity-check equations
SPC code
(n_2, k_2) generalized check node
n_2 edges
n_1 edges
repetition code
(n_1, k_1) generalized variable node
k_1 received bits

Figure 1.1: Structure of a D-GLDPC code.

SPC code, i.e. as a $(n, n-1)$ linear block code. Analogously, a degree-n variable node can be interpreted as a length-n repetition code, i.e. as a $(n, 1)$ linear block code, where the information bit corresponds to the bit received from the communication channel. A first step towards the generalization of LDPC codes consists in letting some of (or eventually all) the check nodes, be generic (n, k) linear block codes: the corresponding code structure is known as GLDPC code. An (n, k) generalized check node is connected to n variable nodes, and corresponds to $n - k$ parity-check equations. Then, for GLDPC codes, the number of parity check equations is no longer equal to the number of check nodes. If there are N_C check nodes, then the number of parity-check equations is $M = \sum_{i=1}^{N_C}(n_i^C - k_i^C)$, where n_i^C and k_i^C are, respectively, the length and the dimension of the i-th check component code.

The generalized check nodes are characterized by better error or erasure correction capability than SPC codes, and they can be favorable from the point of view of the code minimum distance. The drawback of using generalized check nodes is an overall code rate loss, which might be not compensated by their high correction capability [34]. This makes GLDPC codes with uniform check node structure (i.e. composed only of generalized nodes, e.g. only of $(7, 4)$ Hamming codes), despite being characterized by a minimum distance linearly growing up with the codeword length N [31], quite poor in terms of decoding threshold. This poor threshold, which was evaluated in [34] on the BEC assuming bounded distance decoding at the check nodes, does not improve to competitive values even if MAP decoding is performed at the check nodes [37], as it will be shown in Section 2.5.

The second generalization step consists in introducing variable nodes different from repetition codes. The corresponding code structure is known as D-GLDPC code [21, 41], and is represented in Fig. 1.1. An (n, k) generalized variable node is connected to n check nodes, and receives its k information bits from the communication channel. Thus, k of the N encoded bits for the overall D-GLDPC code are received by the (n, k) generalized variable node, and interpreted by the variable node as its k information bits. In a D-GLDPC code, the number of parity check equations is still given by $M = \sum_{i=0}^{N_C} (n_i^C - k_i^C)$, as for a GLDPC code. Moreover, if N_V is the number of variable nodes, the codeword length is $N = \sum_{j=1}^{N_V} k_j^V$, where k_j^V is the dimension of the j-th variable component code.

1.3 Parity-Check Matrix Construction for D-GLDPC Codes

Let us consider a D-GLDPC code with a given sparse bipartite graph, and let us suppose that the bipartite graph is composed of N_V variable nodes, indexed from 1 to N_V, connected to N_C check nodes, indexed from 1 to N_C. For simplicity, we assume that there are no multiple connections between each pair of connected variable and check nodes. The $(N_C \times N_V)$ sparse binary matrix whose element (i, j) is equal to 1 if the i-th check node is connected to the j-th variable node, and is equal to 0 otherwise, is called *adjacency matrix* of the bipartite graph, denoted by $\mathbf{A}$. The adjacency matrix has a one-to-one correspondence with the bipartite graph. The method for obtaining the parity-check matrix of a D-GLDPC with a certain bipartite graph consists in starting from the adjacency matrix of the graph, and then modifying it in two successive steps, as explained in [21].

Let $\mathbf{H}_i^C$ be a $((n_i^C - k_i^C) \times n_i^C)$ parity-check matrix representation for the i-th check node; if the check node is a standard SPC code, then $\mathbf{H}_i^C$ is a row vector of length n_i^C. Let $\mathbf{G}_j^V$ be a $(k_j^V \times n_j^V)$ generator matrix representation for the j-th variable node, and let $\mathbf{G}_j^{V,T}$ be its transposed matrix; if the variable node is a standard repetition code, then $\mathbf{G}_j^V$ is a row vector of length n_j^V. The two steps for obtaining the parity-check matrix $\mathbf{H}$ of the D-GLDPC code from its adjacency matrix $\mathbf{A}$ are described next (for more details we refer to [21]).

- *Row expansion.* For $i = 1, \ldots, N_C$, in row i of $\mathbf{A}$ substitute to each 0 an all-zero column vector of length $n_i^C - k_i^C$, and to each 1 a specific column of $\mathbf{H}_i^C$. Let the obtained $((\sum_{i=1}^{N_C} (n_i^C - k_i^C)) \times N_V)$ binary matrix be $\tilde{\mathbf{H}}$.

- *Column expansion.* For $j = 1, \ldots, N_V$, in column j of $\tilde{\mathbf{H}}$ substitute to each 0 an all-zero row vector of length k_j^V, and to each 1 corresponding to the same check node, the same row of $\mathbf{G}_j^{V,T}$.

The binary matrix with $M = \sum_{i=1}^{N_C} (n_i^C - k_i^C)$ rows and $N = \sum_{j=1}^{N_V} k_j^V$ columns, obtained at the end of the above described two-step procedure, is the parity-check matrix $\mathbf{H}$ for the overall D-GLDPC code. If the code is a standard LDPC code, the parity-check matrix $\mathbf{H}$ is equal to the adjacency matrix $\mathbf{A}$. If the code is a GLDPC code (i.e.

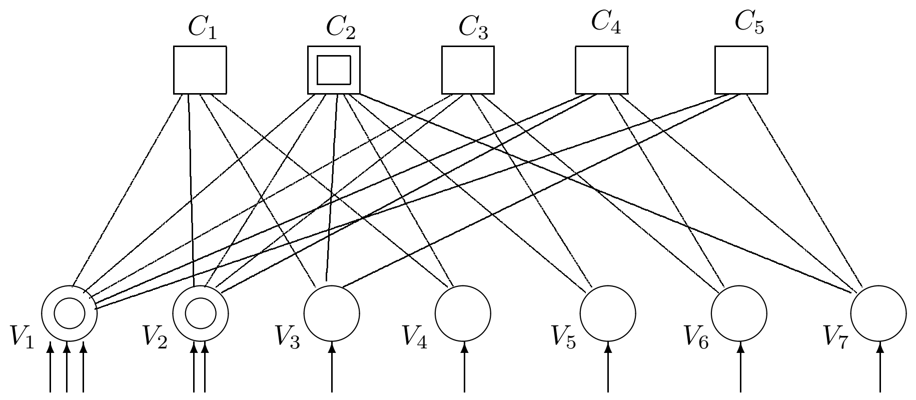


Figure 1.2: Bipartite graph for a $(10, 3)$ D-GLDPC code.

there are generalized check nodes, but there are no generalized variable nodes), then the parity-check matrix $\mathbf{H}$ is equal to $\tilde{\mathbf{H}}$.

Example 1.1. *Consider the* $(10, 3)$ *D-GLDPC code with the bipartite graph depicted in Fig. 1.2, with a* $(5, 3)$ *generalized variable node (*V_1*), a* $(4, 2)$ *generalized variable node (*V_2*) and a* $(6, 3)$ *generalized check node (*C_2*). Suppose that the generator matrices for the two generalized variable nodes, in systematic representation, are*

$$\mathbf{G}_1^V = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and

$$\mathbf{G}_2^V = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix},$$

and that the parity-check matrix for the generalized check node is

$$\mathbf{H}_2^C = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

The adjacency matrix for this code is:

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

where the bold entries emphasize that the second row corresponds to a generalized check node. By applying the row expansion step, *we obtain:*

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

where now the bold entries emphasize that the first and the second columns correspond to generalized variable nodes, and the underlined entries emphasize that, in these two columns, the second, third and fourth rows correspond to the same check node (C_2). By applying the column expansion step *we finally obtain the D-GLDPC code parity check matrix:*

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

1.4 Decoding of D-GLDPC Codes on the BEC

In this section, the message-passing iterative decoding algorithm for D-GLDPC codes on the BEC is described. The iterative decoding algorithm for standard LDPC codes and for GLDPC codes can be obtained as a special instance of the general algorithm presented next. For a detailed description of the iterative decoding algorithm of D-GLDPC codes on the BI-AWGN channel we refer to [21]).

In order to properly describe the decoding algorithm for D-GLDPC codes on the BEC, some notation is needed. The check nodes connected to the variable node V_j are denoted by $C^{j,1}, C^{j,2}, \ldots, C^{j,n_j^V}$, while their indexes are denoted by $c^{j,1}, c^{j,2}, \ldots, c^{j,n_j^V}$. The socket of the check node $C^{j,w}$, connected to the w-th socket of V_j, is indexed by $t^{j,w}$. Analogously, the variable nodes connected to the check node i are denoted by $V^{i,1}, V^{i,2}, \ldots, V^{i,n_i^C}$, while their indexes are denoted by $v^{i,1}, v^{i,2}, \ldots, v^{i,n_i^C}$. The socket of the variable node $V^{i,w}$, connected to the w-th socket of C_i, is indexed by $s^{i,w}$.

The erasure probability of the BEC is q, and the erased bits are denoted by the symbol "?". The transmitted codeword is $\mathbf{b} = [\mathbf{b}_1, \ldots, \mathbf{b}_{N_V}]$ and the sequence received from the BEC is $\mathbf{y} = [\mathbf{y}_1, \ldots, \mathbf{y}_{N_V}]$, where $\mathbf{b}_j = [b_{j,1}, \ldots, b_{j,k_j^V}]$ and $\mathbf{y}_j = [y_{j,1}, \ldots, y_{j,k_j^V}]$, both of size k_j^V, correspond to the j-th variable node. It is $y_{j,t} = b_{j,t}$ with probability $1-q$ and $y_{j,t} =$ "?" with probability q. The number of "?" symbols in a sequence $\mathbf{d}$ is denoted by $\mathcal{E}(\mathbf{d})$.

At the generic decoding iteration ℓ, some of the information bits for the j-th variable are known and the other ones are unknown. These information bits are denoted by $\mathbf{u}_j^\ell = [u_{j,1}^\ell, \ldots, u_{j,k_j^V}^\ell]$ where, for $t = 1, \ldots, k_j^V$, $u_{j,t}^\ell$ may be equal to 0/1 (in case it is known) or to "?" (in case it is unknown). The encoded bits $\mathbf{x}_j^\ell = [x_{j,1}, \ldots, x_{j,n_j^V}]$ that correspond to $\mathbf{u}_j^\ell$ are given by $\mathbf{u}_j^\ell\, \mathbf{G}_j^V$, where $0/1 \cdot ? = ? \cdot 0/1 = 0/1 + ? = ? + 0/1 = ?$.

The *a priori* information that the j-th variable node receives from the check nodes is denoted by $\mathbf{a}_j = [a_{j,1}, \ldots, a_{j,n_j^V}]$. Correspondingly, the *a priori* information that the i-th check node receives from the variable nodes is denoted by $\mathbf{z}_i = [z_{i,1}, \ldots, z_{i,n_i^C}]$.

The iterative decoding algorithm for D-GLDPC codes on the BEC is described next. It is a generalization of the iterative decoder for standard LDPC codes presented in [26]. The decoding algorithm is described next under the hypothesis that MAP decoding is performed at each check and variable component code. Different decoding algorithms (e.g. bounded distance decoding, see Section 2.1.3) may be considered.

Iterative Decoding Algorithm for D-GLDPC codes on the BEC

- *Initialization.* Set $\ell = 1$. For $j = 1, \ldots, N_V$, set $\mathbf{r}_j^\ell = \mathbf{y}_j$ and set $\mathbf{a}_j^\ell$ equal to the all-"?" word of length n_j^V. For $j = 1, \ldots, N_V$, consider the $(k_j^V \times (n_j^V + k_j^V))$ code with code with generator matrix $[\mathbf{G}_j^V \mid \mathbf{I}_{k_j^V}]$, and run MAP decoding on the BEC for this code, assuming $[\mathbf{a}_j^\ell | \mathbf{r}_j^\ell]$ as received. Let the decoded sequence be $[\mathbf{x}_j^\ell | \mathbf{u}_j^\ell]$.

- *Horizontal step.* For $i = 1, \ldots, N_C$, for $w = 1, \ldots, n_i^C$, set $z_{i,w}^\ell = x_{v^{i,w}, s^{i,w}}^\ell$. Run MAP decoding on the BEC for the (n_i^C, k_i^C) check node for the received sequence $\mathbf{z}_i^\ell$. Let the decoded sequence be $\mathbf{e}_i^\ell = [e_{i,1}^\ell, \ldots, e_{i,n_i^C}^\ell]$.

- *Vertical step.* For $j = 1, \ldots, N_V$, set $\mathbf{r}_j^\ell = \mathbf{u}_j^\ell$. For $j = 1, \ldots, N_V$, for $w = 1, \ldots, n_j^V$, set $a_{j,w}^\ell = e_{c^{j,w}, t^{j,w}}^\ell$. For $j = 1, \ldots, N_V$, consider the $(k_j^V \times (n_j^V + k_j^V))$ code with code with generator matrix $[\mathbf{G}_j^V \mid \mathbf{I}_{k_j^V}]$, and run MAP decoding on the BEC for this code, assuming $[\mathbf{a}_j^\ell | \mathbf{r}_j^\ell]$ as received sequence. Let the decoded sequence be $[\mathbf{x}_j^\ell | \mathbf{u}_j^\ell]$.

- *Stopping criterion step.* If $\mathcal{E}(\mathbf{u}^\ell) = 0$, exit and declare a success. Else, if $\ell = \ell_{\max}$, exit and declare a failure. Else, if $\ell \geq 2$ and $\mathcal{E}(\mathbf{u}^\ell) = \mathcal{E}(\mathbf{u}^{\ell-1})$ exit and declare a failure. Else, go to the horizontal step.

In the first half of each decoding iteration (*horizontal step*), the i-th (n_i^C, k_i^C) check node receives n_i^C messages from the variable nodes, stored into $\mathbf{z}_i$: some of them are known messages (i.e. "0" or "1" messages, with infinite reliability) some others are erasure messages (i.e. "?" messages). MAP decoding is then performed at the check node in order to recover its unknown encoded bits, resulting in $\mathbf{e}_i$. After the check node has completed its decoding procedure, a known message is sent through the w-th socket, to the variable node with index $v^{i,w}$ if the bit $e_{i,w}$ is known, while an erasure message is sent if $e_{i,w}$ is still unknown.

In the second half of each decoding iteration (*vertical step*), the generic (n_j^V, k_j^V) variable node receives n_j^V messages from the check nodes, stored into $\mathbf{a}_j$: some of them are erasure messages, some others are known messages. The computation performed by an (n_j^V, k_j^V) variable node is analogous to that of a check node, with the difference that, at each iteration, some of the information bits might be known as well as some of the encoded bits. In order to exploit the partial knowledge of the information bits $\mathbf{u}_j$, MAP decoding is performed on the extended generator matrix $[\mathbf{G}_j^V \,|\, \mathbf{I}_{k_j^V}]$, where $\mathbf{G}_j^V$ is the chosen $(k_j^V \times n_j^V)$ generator matrix for the variable node, and $\mathbf{I}_{k_j^V}$ is the $(k_j^V \times k_j^V)$ identity matrix. After MAP decoding has been performed, some of the previously unknown information and encoded bits for the variable node might be recovered. A known message is then sent through the w-th socket, to the check node with index $c^{j,w}$ if the bit $x_{j,w}$ is known, while erasure messages is sent if $x_{j,w}$ is still unknown.

The algorithm is stopped as soon as there are no longer unknown encoded bits for the overall code (in this case decoding is successful), or when unknown encoded bits for the overall code still exist, but either a maximum number of iterations $\ell_{\max}$ has been reached, or in the last iteration no encoded bits where recovered. It the latter two cases, a decoding failure is declared.

In all the simulation results presented in this thesis, it is assumed $\ell_{\max} = \infty$.

1.5 Dependence of D-GLDPC Codes on the Generalized Variable Nodes Representation

A very important property of D-GLDPC codes is that the structure and the properties of the code depend on the representation chosen for its generalized variable nodes. This fact is addressed next with an example.

Let us consider the $(8, 3)$ D-GLDPC code with the bipartite graph depicted in Fig. 1.3. Such bipartite graph is composed of five degree-3 standard check nodes (length-3 SPC codes), five degree-2 standard variable nodes (length-2 repetition codes), and one $(5, 3)$ generalized variable node. Let the code book for the $(5, 3)$ generalized variable node be $\{00000, 10011, 01001, 00110, 11010, 10101, 01111, 11100\}$. Two possible (3×5) generator matrix representations that correspond to this code book are

$$\mathbf{G_1} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{1.1}$$

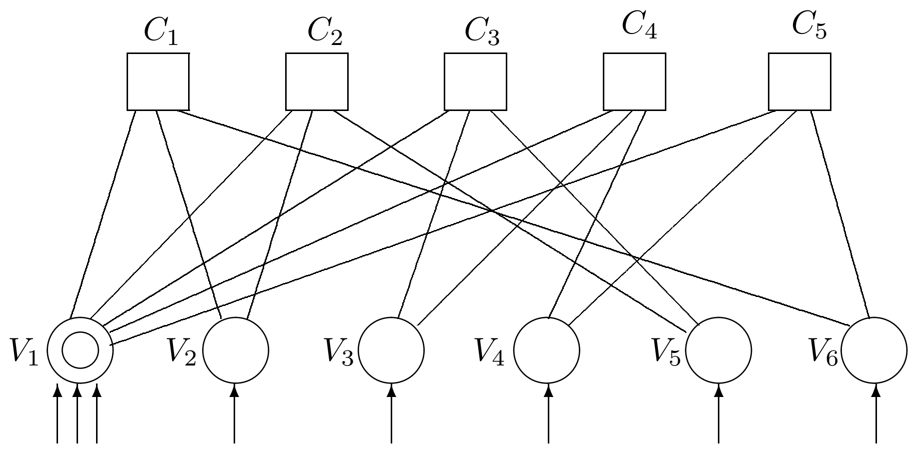


Figure 1.3: Bipartite graph for a $(8, 3)$ D-GLDPC code.

and

$$\mathbf{G_2} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \tag{1.2}$$

where $\mathbf{G_1}$ is a systematic representation for the generalized variable node, and $\mathbf{G_2}$ is a non-systematic representation.

Assuming the systematic representation $\mathbf{G_1}$ for the generalized variable node V_1, and applying the rules for developing the D-GLDPC code parity-check matrix described in Section 1.4, we obtain the following parity-check matrix for the overall D-GLDPC code:

$$\mathbf{H_{G_1}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{1.3}$$

On the other hand, assuming the non-systematic representation $\mathbf{G_2}$, we obtain the parity-check matrix

$$\mathbf{H_{G_2}} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{1.4}$$

The code book for the overall D-GLDPC code corresponding to $\mathbf{H_{G_1}}$ is

$$\begin{array}{rcccccccc}
\mathbf{x_1} = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x_2} = & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{x_3} = & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
\mathbf{x_4} = & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
\mathbf{x_5} = & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
\mathbf{x_6} = & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
\mathbf{x_7} = & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
\mathbf{x_8} = & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1\,,
\end{array}$$

while the code book corresponding to $\mathbf{H_{G_2}}$ is

$$\begin{array}{rcccccccc}
\mathbf{x_1} = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x_2} = & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{x_3} = & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
\mathbf{x_4} = & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
\mathbf{x_5} = & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
\mathbf{x_6} = & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
\mathbf{x_7} = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
\mathbf{x_8} = & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1\,.
\end{array}$$

It is clear that the choice of different representations for the generalized variable node V_1 lead to different D-GLDPC codes, with different properties. For instance, in this specific example, we obtain a D-GLDPC code with minimum distance $d_{\min} = 2$ by choosing $\mathbf{G_1}$ and a D-GLDPC code with minimum distance $d_{\min} = 3$ by choosing $\mathbf{G_2}$.

We also explicitly remark that this fact is not shared by the generalized check nodes. Assuming a different representation for the generalized check nodes does not modify the code book of the overall D-GLDPC code.

Chapter 2

Analysis of Generalized and Doubly-Generalized LDPC Codes on the Binary Erasure Channel

In this chapter, a technique for the asymptotic analysis of GLDPC and D-GLDPC codes on the BEC is presented. This technique is based on extrinsic information transfer (EXIT) charts [43, 10, 44, 45]. As it will be explained in the following, the success of EXIT chart analysis is bound to the knowledge of the EXIT function for each check and variable component code. By exploiting results from [44, Theorem 2] it is proved that, if the communication channel is a BEC, then the EXIT function of a check component code, under *maximum a posteriori* (MAP) decoding, can be related to the code *information functions* (a concept first introduced in [46]), and the EXIT function of a variable component code, under MAP decoding, to the code *split information functions* (see Section 2.1). This relationship between EXIT function and (split) information functions is very useful for the threshold analysis on the BEC of GLDPC and D-GLDPC codes built up with component codes whose (split) information functions are known. A major problem is that, for a wide range of linear block codes, including most binary double error-correcting and more powerful BCH codes, these parameters are still unknown. In fact, no closed-form expression is available as a function of the code dimension k and code length n, and a direct computation of these parameters is unfeasible, due to the huge computation time required, even for small codeword lengths. This is the case, for instance, of the split information functions for the $(31, 10)$ dual code of a narrowsense binary $(31, 21)$ BCH code.

In this work, a solution for the asymptotic analysis of GLDPC and D-GLDPC codes, which permits to overcome the impossibility to evaluate the EXIT function for the check or variable component codes also in situations where the information functions or split information functions remain unknown, is developed. The proposed method consists in considering *random* check and variable generalized component codes belonging to an expurgated ensemble, instead of specific check and variable generalized component codes (like Hamming codes, BCH codes, etc., or the dual codes). As it will be explained in Section 2.2 this expurgated ensemble is the ensemble of all the binary linear block codes with minimum distance $d_{\min}$ satisfying $d_{\min} \geq 2$. A technique is then developed in order to exactly evaluate the expected information functions for each check component code and the expected split information functions for each variable component code over the considered expurgated ensemble. This permits to evaluate the expected EXIT function for each check component code or variable component code, supposing transmission on the BEC, and the expected EXIT function for the overall check node decoder (CND) or variable node decoder (VND).

The developed analysis tool is then exploited in order to design capacity approaching GLDPC and D-GLDPC distributions. Simulation results obtained on random, long codes, reveal that capacity approaching D-GLDPC codes can exhibit better threshold *and* lower error floor than capacity approaching LDPC and GLDPC codes. Moreover, by imposing constraints on the fraction of edges towards the generalized check nodes, the error floor of D-GLDPC codes can be further lowered (at the cost of increased complexity), while preserving good or very good waterfall performance.

This chapter is organized as follows. In Section 2.1 the relationship between the EXIT function of check and variable component codes on the BEC, and (split) information functions, is recalled. In the same section, the EXIT function of a generalized check node on the BEC, assuming bounded distance decoding instead of MAP decoding, is investigated. Section 2.2 is devoted to the explanation of the random component codes hypothesis and to the definition of the expurgated ensemble of check and variable component codes, which guarantees a correct application of the EXIT chart analysis. In Sections 2.3 and 2.4 the technique for the evaluation of the expected information functions for a random check component code and, respectively, of the expected split information functions for a random variable component code, are developed. Numerical results involving threshold analysis, distribution optimization and finite-length performance analysis, for both GLDPC and D-GLDPC codes, are presented in Section 2.5. Finally, concluding remarks are presented in Section 4.6.

2.1 EXIT Functions for Generalized Variable and Check Nodes on the BEC

In [44, Fig. 3], a general decoding model is described, which can be effectively adopted in order to express the EXIT function for the generalized check nodes of a GLDPC or D-GLDPC code, and for the generalized variable nodes of D-GLDPC code. This general decoding model is depicted in Fig. 2.1. The encoder of a linear block code with dimension

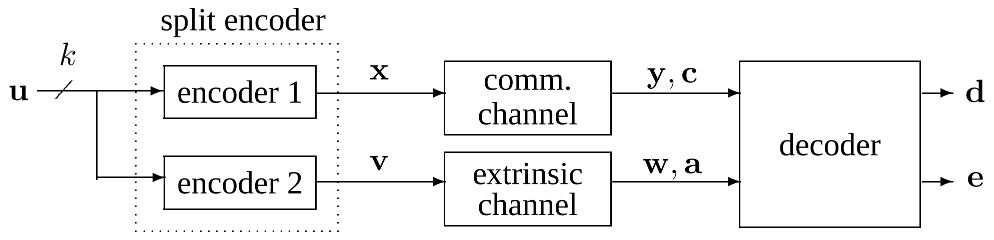


Figure 2.1: General decoding model for the variable and check nodes of a D-GLDPC code.

k is split into two linear encoders, namely *encoder 1* and *encoder 2*, with generator matrices $\mathbf{G}_1$ and $\mathbf{G}_2$ generating a codeword $(\mathbf{x}, \mathbf{v})$ (with $\mathbf{x} = \mathbf{u}\,\mathbf{G}_1$ and $\mathbf{v} = \mathbf{u}\,\mathbf{G}_2$), and codeword length $|\mathbf{x}| + |\mathbf{v}|$, where $|\cdot|$ denotes the vector size. For each information word $\mathbf{u}$, the encoded bits $\mathbf{x}$ are transmitted over a *communication channel*, resulting in $\mathbf{y}$, while the encoded bits $\mathbf{v}$ are transmitted over an *extrinsic channel*, resulting in $\mathbf{w}$. Both the likelihood ratios (LRs) $\mathbf{c}$ and $\mathbf{a}$, relative to $\mathbf{y}$ and $\mathbf{w}$ respectively, are exploited by the *a posteriori probability* (APP) decoder in order to compute the LRs $\mathbf{d}$ for the encoded bits, and the extrinsic LRs $\mathbf{e}$. In the following, capital letters are used to denote random variables, lower case letters to denote realizations of such random variables.

The EXIT function of the linear code in Fig. 2.1, assuming that the encoders 1 and 2 have no idle bits, that MAP decoding is performed, that communication channel is a BEC with erasure probability q and that the extrinsic channel is a BEC with erasure probability p, has been shown in [44, eq. 36] to be expressed by

$$I_E(p,q) = \frac{1}{|\mathbf{v}|}\sum_{i=1}^{|\mathbf{v}|} I(V_i; \mathbf{Y}, \mathbf{W}_{[\mathbf{i}]}) = 1 - \frac{1}{n}\sum_{h=0}^{k}(1-q)^h q^{k-h}\sum_{g=1}^{n}(1-p)^{g-1}p^{n-g} \cdot [g\,\tilde{e}_{g,h} - (n-g+1)\tilde{e}_{g-1,h}]. \quad (2.1)$$

In (2.1), V_i is the i-th bit of the encoded word $\mathbf{V}$ (whose elements are i.i.d. Bernoulli random variables with equiprobable values), $\mathbf{Y}$ is the random word outcoming from the communication channel, $\mathbf{W}_{[i]}$ is the random word outcoming from the extrinsic channel except the bit W_i, $I(\cdot)$ denotes the mutual information, and $\tilde{e}_{g,h}$ is the (g,h)-th *un-normalized split information function*. This parameter is defined as the summation of the dimensions of all the possible codes obtained considering just g positions among the encoded bits $\mathbf{v}$ and just h positions among the encoded bits $\mathbf{x}$. It can be computed by performing the summation of the ranks of the $\binom{n}{g}\binom{k}{h}$ submatrices obtained by selecting g columns in $\mathbf{G}_1$ and h columns in $\mathbf{G}_2$.

We refer to this decoding model in order to describe and analyze each generalized check node of a GLDPC or D-GLDPC code and each generalized variable node of a D-GLDPC code. In the contest of GLDPC and D-GLDPC codes, the communication channel is the channel over which the encoded bits of the overall code are transmitted; moreover, the extrinsic channel represents a model for the channel over which the messages are exchanged between variable and check nodes, during the iterative decoding process. If the communication channel is a BEC, then also the extrinsic channel is modelled as a BEC.

2.1.1 EXIT Function for the Variable Nodes on the BEC

The generic variable node (either a repetition code or a generalized one), representing an (n, k) linear block code, receives its k information bits from the communication channel, and interfaces to the extrinsic channel through its n encoded bits. For this reason, for a variable node, the encoder 1 is represented by the identity mapping $\mathbf{x} = \mathbf{u}$ (i.e. $\mathbf{G}_1 = \mathbf{I}_k$) and the encoder 2 performs a linear mapping $\mathbf{v} = \mathbf{G}\,\mathbf{u}$ (i.e. $\mathbf{G}_2 = \mathbf{G}$), where $\mathbf{G}$ is one of the several possible generator matrix representations for the (n, k) linear block code. In this case it results $|\mathbf{x}| = k$ and $|\mathbf{v}| = n$. Thus, the (n, k) variable node is indeed interpreted as a $(n + k, k)$ code whose generator matrix is in the form $[\mathbf{G}\,|\,\mathbf{I}_k]$, and its EXIT function on the BEC is the given by (2.1), with the encoder 1 being the identity mapping $\mathbf{x} = \mathbf{u}$ and the encoder 2 being the linear mapping $\mathbf{v} = \mathbf{G}\,\mathbf{u}$. This EXIT function can be equivalently expressed by:

$$I_E(p, q) = 1 - \frac{1}{n} \sum_{t=0}^{n-1} \sum_{z=0}^{k} p^t\,(1-p)^{n-t-1}\,q^z\,(1-q)^{k-z} \cdot \left[(n-t)\tilde{e}_{n-t,k-z} - (t+1)\tilde{e}_{n-t-1,k-z}\right], \tag{2.2}$$

which can be easily obtained from (2.1) by performing the substitutions $t = n - g$ and $z = k - h$. Expressions (2.1) and (2.2) are valid under the hypothesis of MAP erasure decoding. If applied to a $(n, 1)$ repetition code, (2.2) leads to $I_E(p, q) = 1 - q\,p^{n-1}$, i.e. to the well known expression of the EXIT function for a degree-n variable node of an LDPC code on the BEC.

It was shown in Section 1.5 that the properties of a D-GLDPC code depend on the chosen representation for its generalized variable nodes. Analogously, the split information functions are not unique for a given (n, k) generalized variable node and depend on the specific representation chosen for its generator matrix $\mathbf{G}$. This can be justified as follows. Different generator matrices correspond to different mappings of vectors $\mathbf{u}$'s to vectors $\mathbf{v}$'s. Hence, for a given information vector $\mathbf{u}$, a generator matrix $\mathbf{G}'$ will lead to a codeword $(\mathbf{v}', \mathbf{u})$ for the $(n + k, k)$ code with split encoder, while a generator matrix $\mathbf{G}''$ will lead to a different codeword $(\mathbf{v}'', \mathbf{u})$, thus generating a different code book. As a consequence, the EXIT function for a (n, k) linear block code, when used at a variable node within a D-GLDPC code, depends on the generator matrix representation chosen for the code. This fact does not hold for repetition codes (i.e. for the traditional variable node of LDPC and GLDPC codes), for which only one code representation is possible. Then, an important difference between variable nodes represented by repetition codes and generalized (n, k) variable nodes of a D-GLDPC code (with $k > 1$) is that, in the latter case, different representations of the generator matrix $\mathbf{G}$ are possible. These different code representations correspond to a *different performance* of the overall D-GLDPC code. The code representation for the generalized variable nodes becomes a degree of freedom for the code design.

2.1.2 EXIT Function for the Check Nodes on the BEC

For a check node (either a SPC code or a generalized one), no communication channel is present. Moreover, any check node representing an (n, k) linear block code interfaces to the extrinsic channel through its n encoded bits. Then, the encoder 1 is not present, while the encoder 2 performs a linear mapping $\mathbf{v} = \mathbf{G}\,\mathbf{u}$, where $\mathbf{G}$ is one of the several possible generator matrix representations for the (n, k) linear block code. It follows that $|\mathbf{v}| = n$. This model is the same proposed in [44, Sec. VII-A].

The EXIT function of a generic (n, k) check node of a D-GLDPC code on the BEC can be obtained by letting $q \to 1$ in (2.2) (no communication channel is present). The obtained expression, equivalent to [44, eq. 40], is

$$I_E(p) = \frac{1}{|\mathbf{v}|} \sum_{i=1}^{|\mathbf{v}|} I(V_i; \mathbf{W}_{[i]}) = 1 - \frac{1}{n} \sum_{t=0}^{n-1} p^t (1-p)^{n-t-1} \cdot [(n-t)\, \tilde{e}_{n-t} - (t+1) \tilde{e}_{n-t-1}], \tag{2.3}$$

where, for $g = 0, \dots, n$, $\tilde{e}_g$ is the g-th (un-normalized) information function. It is defined as the summation of the dimensions of all the possible codes obtained considering just g positions in the code block $\mathbf{v}$ of length n [46]. This parameter can be computed by performing the summation of the ranks of the $\binom{n}{g}$ submatrices obtained by selecting g columns in $\mathbf{G}$.

As for (2.2), also (2.3) assumes that erasures are corrected at the check node according to MAP decoding. If applied to a $(n, n-1)$ SPC code, (2.3) leads to the expression of the EXIT function on the BEC for a degree-n check node of an LDPC code, i.e. $I_E(p) = (1-p)^{n-1}$.

Since the code book of a linear block code is independent of the choice of the generator matrix, different code representations have the same information functions. Thus, different code representations for a generalized check node lead to the same EXIT function. This means that, differently from what happens for the generalized variable nodes, the performance of a GLDPC or D-GLDPC code is independent of the specific representation of its generalized check nodes.

2.1.3 Bounded-Distance EXIT Functions

Decoding algorithms that are less powerful than MAP decoding, but having a reduced complexity, may be used at the generalized variable and check nodes. In these cases, different expressions of the EXIT function must be considered. For example, let us consider a generalized (n,k) check node, and let us suppose that erasure recovery is performed according to the following bounded-distance decoding strategy, here referred to as d-bounded-distance decoding: "if the number of received erasures from the extrinsic channel is less than or equal to d, then execute MAP decoding, otherwise declare a decoding failure".

Theorem 2.1. *If the extrinsic channel is a BEC with erasure probability p, then the EXIT function for a generalized (n,k) check node without idle bits, under d-bounded-distance decoding, is given by:*

$$\begin{aligned} I_E(p) = 1 &- \frac{1}{n}\sum_{t=0}^{d-1}(1-p)^{n-t-1}p^t[(n-t)\tilde{e}_{n-t}-(t+1)\tilde{e}_{n-t-1}] \\ &- \sum_{t=d}^{n-1}(1-p)^{n-t-1}p^t\binom{n-1}{t}. \end{aligned} \tag{2.4}$$

Proof. The following equalities hold:

$$\begin{aligned} I_E(p) &= \frac{1}{|\mathbf{v}|}\sum_{i=1}^{|\mathbf{v}|} I(V_i;\mathbf{W}_{[i]}) \\ &= \frac{1}{n}\sum_{i=1}^{n} H(V_i) - \frac{1}{n}\sum_{i=1}^{n} H(V_i|\mathbf{W}_{[i]}) \\ &= 1-\frac{1}{n}\sum_{t=0}^{n-1}p^t(1-p)^{n-t-1}\sum_{i=1}^{n}\sum_{\mathbf{w}_{[i]}^{(t)}} H(V_i\,|\,\mathbf{W}_{[i]}=\mathbf{w}_{[i]}^{(t)}), \end{aligned}$$

where $H(\cdot)$ is the entropy function, $\mathbf{w}_{[i]}^{(t)}$ is a specific realization of $\mathbf{W}_{[i]}$ with t erasures, and $\sum_{i=1}^{n} H(V_i) = n$ since the code has no idle bits. The d-bounded-distance decoding strategy is equivalent to assume that is equal to 1 the entropy of each encoded bit V_i given any realization of $\mathbf{W}_{[i]}$ with $t \geq d$ erasures, thus leading to the term $\sum_{t=d}^{n-1}(1-p)^{n-t-1}p^t\binom{n-1}{t}$. On the contrary, if $t < d$, MAP decoding is performed thus leading to the same terms as in (2.3). □

Example 2.1. *In [34, Tab. 2], some thresholds on the BEC are presented for GLDPC codes with uniform check node structure, composed of narrow-sense binary BCH codes. These thresholds have been evaluated through a generalization of density evolution [47, 7,* *48, 49, 50], assuming d-bounded-distance decoding at the check nodes, with $d = d_{\min} - 1$. The same thresholds can also be obtained through an EXIT charts approach exploiting* (2.4) *for the BCH check nodes, with $d = d_{\min} - 1$.*

2.2 Random Code Hypothesis and Expurgated Ensemble Definition

Consider a D-GLDPC code with $\mathcal{I}_V$ different types of variable component codes and $\mathcal{I}_C$ different types of check component codes. The i-th variable component code type has EXIT function on the BEC $I_{E,V}^{(i)}(p,q)$, corresponding to a specific code representation, and is assumed without idle components. Analogously, the i-th check component code type has EXIT function $I_{E,C}^{(i)}(p)$, and is assumed without idle components. Variable and check nodes are supposed *randomly connected* through an edge interleaver. The fraction of edges towards the variable nodes of type i is denoted by λ_i, and the fraction of edges towards the check nodes of type i by ρ_i. Then, the total EXIT function for, respectively, the VND and for the CND is given by

$$I_{E,V}(p,q) = \sum_{i=1}^{\mathcal{I}_V} \lambda_i \, I_{E,V}^{(i)}(p,q) \tag{2.5}$$

$$I_{E,C}(p) = \sum_{i=1}^{\mathcal{I}_C} \rho_i \, I_{E,C}^{(i)}(p). \tag{2.6}$$

These relationships can be obtained by reasoning in the same way as [44] for the EXIT functions of the variable and check nodes decoders of an irregular LDPC code.

Next, we introduce the random code hypothesis. Each variable or check component code is assumed a random linear block code. More specifically, the generator matrix of each (n_i, k_i) variable component code of type i ($i = 1, \ldots, \mathcal{I}_V$) is assumed uniformly chosen at random from an expurgated ensemble denoted by $\mathcal{G}_*^{(n_i,k_i)}$. The generator matrix of each (n_i, k_i) check component code of type i ($i = 1, \ldots, \mathcal{I}_C$) is assumed uniformly chosen at random from the same expurgated ensemble.

Let $\mathbb{E}[I_{E,V}(p,q)]$ and $\mathbb{E}[I_{E,C}(p)]$ be, respectively, the expected variable and check EXIT function for such D-GLDPC code ensemble. Then, from (2.5) and (2.6) we have

$$\begin{aligned}\mathbb{E}[I_{E,V}(p,q)] &= \mathbb{E}\Big[\sum_{i=1}^{\mathcal{I}_V} \lambda_i \, I_{E,V}^{(i)}(p,q)\Big] \\ &= \sum_{i=1}^{\mathcal{I}_V} \lambda_i \, \mathbb{E}_{\mathcal{G}_*^{(n_i,k_i)}}[I_{E,V}^{(i)}(p,q)] \end{aligned} \tag{2.7}$$

$$\mathbb{E}[I_{E,C}(p)] = \mathbb{E}\Big[\sum_{i=1}^{\mathcal{I}_C} \rho_i \, I_{E,C}^{(i)}(p)\Big]$$
$$= \sum_{i=1}^{\mathcal{I}_C} \rho_i \, \mathbb{E}_{\mathcal{G}_*^{(n_i,k_i)}}[I_{E,C}^{(i)}(p)], \tag{2.8}$$

where $\mathbb{E}_{\mathcal{G}_*^{(n,k)}}[\cdot]$ denotes the expectation over $\mathcal{G}_*^{(n,k)}$. If the i-th variable code type is a repetition code, or the i-th check code type is a SPC code, no expectation is needed.

The reason for an expurgated ensemble of $(n \times k)$ generator matrices is needed, instead of the ensemble of all the possible $(k \times n)$ binary matrices with rank k, is to ensure a correct application of the EXIT charts analysis, as explained next.

The aim is to perform a threshold analysis of the D-GLDPC codes ensemble through an EXIT chart approach, using the expected EXIT functions expressed by (2.7) and (2.8). In order to correctly apply the analysis based on EXIT charts, the following conditions are required:

$$\lim_{p \to 0} \mathbb{E}[I_{E,V}(p,q)] = 1 \quad \forall\, q \in (0,1) \tag{2.9}$$
$$\lim_{p \to 0} \mathbb{E}[I_{E,C}(p)] = 1 \tag{2.10}$$
$$\lim_{p \to 1} \mathbb{E}[I_{E,C}(p)] = 0. \tag{2.11}$$

Conditions (2.9) and (2.10) guarantee that, for both the variable and the check node decoder, the average extrinsic information outcoming from the decoder is equal to 1 when the extrinsic channel in Fig. 2.1 is noiseless. Condition (2.11) guarantees that the average extrinsic information from the check node decoder is 0 when the extrinsic channel is the useless channel.

Since the conditions $\sum_{i=1}^{\mathcal{I}_V} \lambda_i = 1$ and $\sum_{i=1}^{\mathcal{I}_C} \rho_i = 1$ hold, it follows from (2.7) and (2.8) that (2.9) and (2.10) are satisfied if these relationships hold:

$$\lim_{p \to 0} \mathbb{E}_{\mathcal{G}_*^{(n_i,k_i)}}[I_{E,V}^{(i)}(p,q)] = 1 \quad \forall\, q \in (0,1),\, \forall\, i = 1, \dots, \mathcal{I}_V \tag{2.12}$$
$$\lim_{p \to 0} \mathbb{E}_{\mathcal{G}_*^{(n_i,k_i)}}[I_{E,C}^{(i)}(p)] = 1 \quad \forall\, i = 1, \dots, \mathcal{I}_C. \tag{2.13}$$

Analogously, (2.11) is fulfilled if

$$\lim_{p \to 1} \mathbb{E}_{\mathcal{G}_*^{(n_i,k_i)}}[I_{E,C}^{(i)}(p)] = 0 \quad \forall\, i = 1, \dots, \mathcal{I}_C. \tag{2.14}$$

In order to have (2.12), (2.13) and (2.14) satisfied it is sufficient that, for each (n,k), analogue relationships hold for the EXIT function of all the linear block codes represented by the generator matrices in $\mathcal{G}_*^{(n,k)}$. In other words, for each check component code represented by a generator matrix in $\mathcal{G}_*^{(n,k)}$, it must be $\lim_{p \to 0} I_E(p) = 1$ and $\lim_{p \to 1} I_E(p) = 0$; for each variable component code represented by a generator matrix in $\mathcal{G}_*^{(n,k)}$ it must be $\lim_{p \to 0} I_E(p,q) = 1$ for all $q \in (0,1)$. This is indeed the key for obtaining a definition of $\mathcal{G}_*^{(n,k)}$. In Subsections 2.2.1 and 2.2.2, two equivalent definitions of $\mathcal{G}_*^{(n,k)}$ are provided.

2.2.1 Operational Definition of $\mathcal{G}_*^{(n,k)}$

Definition 2.1 (independent set). *Given a $(k \times n)$ binary matrix of rank r, a set of t columns is said an* independent set *when removing these t columns from the matrix leads to a $(k \times (n-t))$ matrix with rank $r - \Delta r < r$, for some $0 < \Delta r \le t$. The number t is the size of the independent set.*

An independent set of size 1 will be also called *independent column*. An independent column is linearly independent of all the other columns of the matrix.

Theorem 2.2. $\mathcal{G}_*^{(n,k)}$ *is the ensemble of all the $(k \times n)$ binary matrices with rank k, without all-zero columns and without independent columns.*

Proof. Consider first a check component code. From (2.3) it follows

$$\lim_{p\to 0} I_E(p) = 1 - (\tilde{e}_n - \frac{\tilde{e}_{n-1}}{n}).$$

Then, the desired property $\lim_{p\to 0} I_E(p) = 1$ is guaranteed by the equality $n\,\tilde{e}_n = \tilde{e}_{n-1}$. This equality is satisfied when the generator matrix $\mathbf{G}$ is full-rank ($\text{rank}(\mathbf{G}) = k$), and when the $(k \times (n-1))$ matrix obtained by removing any column from $\mathbf{G}$ is full rank, i.e. $\mathbf{G}$ has no independent columns. In fact, in this case both sides of the previous equality are equal to $k\,n$. By reasoning in the same way, it is readily proved from (2.3) that

$$\lim_{p\to 1} I_E(p) = 1 - \frac{\tilde{e}_1}{n}.$$

Then, $\lim_{p\to 1} I_E(p) = 0$ when $\tilde{e}_1 = n$, i.e. when the generator matrix has no zero columns. This is equivalent to assume that the component code has no idle bits, an hypothesis already implicitly considered in (2.3). Then, (2.10) and (2.11) are satisfied if the generator matrix of any check component code is full rank, has no independent columns, and has no zero columns.

Consider a variable component code. From (2.2):

$$\lim_{p\to 0} I_E(p,q) = 1 - \sum_{z=0}^{k} q^z (1-q)^{k-z} (\tilde{e}_{n,k-z} - \frac{\tilde{e}_{n-1,k-z}}{n}).$$

If $n\,\tilde{e}_{n,h} = \tilde{e}_{n-1,h}$ for $h = 0, \ldots, k$, then $\lim_{p\to 0} I_E(p,q) = 1$ for any q. This is always true when the $(k \times n)$ generator matrix is full rank ($\text{rank}(\mathbf{G}) = k$) and has no independent columns. In fact, in this case $n\,\tilde{e}_{n,h} = \tilde{e}_{n-1,h} = \binom{k}{h} n\,k$. The constraint that the generator matrix has no zero columns must be also considered, since it is a key hypothesis for the validity of (2.1). Then, (2.9) is satisfied if the generator matrix of any variable component code is full rank, has no independent columns, and has no zero columns. □

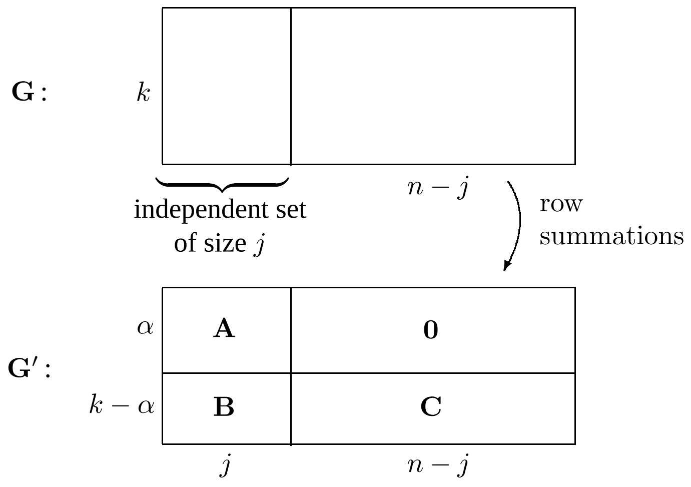


Figure 2.2: If rank($\mathbf{G}$) $= r$ and the first j columns are an independent set, then $\mathbf{G}$ can be transformed as shown by row summations only, with $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ non-null matrices.

This *operational definition* will be used in the following sections for developing a technique for the computation of the expected EXIT functions for the VND and CND of GLDPC and D-GLDPC codes. A more theoretical definition of $\mathcal{G}_*^{(n_i,k_i)}$ is given next.

2.2.2 Theoretical Definition of $\mathcal{G}_*^{(n,k)}$

If j columns are an independent set for a certain generator matrix representation of a linear block code, they are an independent set for any other representation. Moreover, removing them from any representation of the generator matrix leads to the same rank reduction Δr. This is because all the possible representations of the generator matrix can be obtained by row summations starting from any other representation.

Lemma 2.1. *If any representation of the generator matrix of a (n, k) linear block code has an independent set of size j, then the code minimum distance $d_{\min}$ satisfies $d_{\min} \leq j$.*

Proof. Let us suppose that j columns of a generator matrix $\mathbf{G}$ are an independent set of size j. Then, it must be possible to perform row summations on $\mathbf{G}$ in order to obtain a new generator matrix representation $\mathbf{G}'$, where a certain number α of rows have all their 1's in correspondence of only the columns of the independent set (see for example Fig. 2.2, where the first j columns are supposed an independent set, and where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are non-null matrices). Any of these α rows is a valid codeword. Then, $d_{\min} \leq j$. □

Lemma 2.2. *Let* $\mathbf{G}$ *be any representation of the generator matrix of an (n, k) linear block code. Then, the following statements are equivalent:*

a) the code has minimum distance t;

b) the minimum size of the independent sets of $\mathbf{G}$ *is t.*

Proof. [a $\Rightarrow$ b] If $d_{\min} = t$, then it is possible to construct a representation of $\mathbf{G}$ where there is at least one row with exactly t 1's. The columns of $\mathbf{G}$ corresponding to these t 1's are an independent set (of size t), because removing them from $\mathbf{G}$ leads to a reduction of the rank. This independent set must be of minimum size. In fact, if it existed an independent set of size $j < t$, then from Lemma 2.1 it would follow $d_{\min} \leq j < t$, thus violating the hypothesis $d_{\min} = t$.
[b $\Rightarrow$ a] Let us suppose that the minimum size of the independent sets of $\mathbf{G}$ is t, and let us consider an independent set of size t. From Lemma 2.1 it follows that $d_{min} \leq t$. The proof is completed by showing that it is not possible to have $d_{\min} < t$. In fact, if $d_{\min} = j < t$ then, by reasoning in the same way as for the [a $\Rightarrow$ b] proof, it would follow that the minimum size of the independent sets of $\mathbf{G}$ is $j < t$, which violates the hypothesis. □

By combining Theorem 2.2 and Lemma 2.2, it is immediate to obtain the following theorem.

Theorem 2.3. $\mathcal{G}_*^{(n,k)}$ *is the ensemble of all the $(k \times n)$ binary matrices that represent linear block codes without idle components and with minimum distance $d_{\min} \geq 2$.*

It follows from (2.2) and (2.3) that the problem of evaluating the expected EXIT function on $\mathcal{G}_*^{(n,k)}$, for each (n,k) variable (check) node, can be completely solved by evaluating the expected split information functions (information functions) on $\mathcal{G}_*^{(n,k)}$. In Section 2.3 and Section 2.4 a technique for solving this problem for the generalized check nodes and, respectively, for the generalized variable nodes is presented.

2.3 Expected Information Functions Computation

In this section we present an approach to compute the expected values of the information functions for any random (n,k) linear block code, where the expectation is over the expurgated ensemble $\mathcal{G}_*^{(n,k)}$ of all the (n,k) generator matrices representing codes with minimum distance at least 2. The method is based on some recursive formulas that permit to compute the exact number of binary matrices with specific properties.

Let $\mathbf{G}$ be a generator matrix randomly chosen in $\mathcal{G}_*^{(n,k)}$, and let $\mathcal{S}_g$ be a submatrix of $\mathbf{G}$ obtained selecting g columns. The expectation of $\tilde{e}_g$ can be developed as:

$$\begin{aligned}\mathbb{E}_{\mathcal{G}_*^{(n,k)}}[\tilde{e}_g] &= \mathbb{E}_{\mathcal{G}_*^{(n,k)}}\big[\sum_{\mathcal{S}_g}\operatorname{rank}(\mathcal{S}_g)\big]\\ &= \sum_{\mathcal{S}_g}\mathbb{E}_{\mathcal{G}_*^{(n,k)}}\left[\operatorname{rank}(\mathcal{S}_g)\right]\\ &= \binom{n}{g}\mathbb{E}_{\mathcal{G}_*^{(n,k)}}\left[\operatorname{rank}(\mathcal{S}_g)\right], \qquad (2.15)\end{aligned}$$

where the last equality is due to the fact that, for random matrices in $\mathcal{G}_*^{(n,k)}$, the expectation of the rank when selecting g columns is independent of the specific selected columns. In the following, we suppose that $\mathcal{S}_g$ in (2.15) is the submatrix composed of the first g columns of $\mathbf{G}$, without loss of generality. The expectation of $\operatorname{rank}(\mathcal{S}_g)$ in (2.15) can be further developed in the following way:

$$\begin{aligned}\mathbb{E}_{\mathcal{G}_*^{(n,k)}}\left[\operatorname{rank}(\mathcal{S}_g)\right] &= \sum_{u=1}^{\min\{k,g\}} u\cdot\Pr\{\operatorname{rank}(\mathcal{S}_g)=u\}\\ &= \sum_{u=1}^{\min\{k,g\}} u\cdot\frac{K(k,n,g,u,k)}{J(k,n,k)}, \qquad (2.16)\end{aligned}$$

where the summation in u is from $u=1$ and not from $u=0$ because $\mathcal{S}_g$ has no zero columns by hypothesis. In (2.16), we denote by $J(m,n,r)$ the number of rank-r $(m\times n)$ binary matrices without zero columns, and such that removing any column does not reduce the rank (i.e. with no independent columns). According to Theorem 2.2, it is $J(k,n,k)=|\mathcal{G}_*^{(n,k)}|$. The function $K(m,n,g,u,r)$ represents the number of rank-r $(m\times n)$ binary matrices without zero columns, without independent columns, and such that the first g columns have rank u. For any $1\le g\le n$, we have

$$\sum_{u=1}^{\min\{k,g\}} K(m,n,g,u,r) = J(m,n,r). \qquad (2.17)$$

Next we develop recursive formulas for computing $J(\cdot)$ and $K(\cdot)$. Even if $J(\cdot)$ can be expressed in terms of $K(\cdot)$ according to (2.17), an independent recursive formula for $J(\cdot)$ is presented. The proofs of lemmas and theorems claimed in this section are presented in Appendix 2.7. In this and in the next section, we will often invoke the following well known result.

Lemma 2.3. *The number of rank-r $(m\times n)$ binary matrices is given by*

$$\prod_{j=0}^{r-1}\frac{(2^m-2^j)(2^n-2^j)}{(2^r-2^j)}.$$

2.3.1 Computation of $J(m,n,r)$

The number $J(m,n,r)$ of rank-r $(m \times n)$ binary matrices without zero columns and without independent columns is computed as difference between the total number $F(m,n,r)$ of rank-r $(m \times n)$ binary matrices without zero columns and the number of such matrices with at least one independent column.

Lemma 2.4. *Let $F(m,n,r)$ be the number of rank-r $(m \times n)$ binary matrices without zero columns. Then*

$$F(m,n,r) = \prod_{j=0}^{r-1} \frac{(2^m - 2^j)(2^n - 2^j)}{(2^r - 2^j)} - \sum_{z=1}^{n-r} \binom{n}{z} F(m, n-z, r). \tag{2.18}$$

In (2.18) it must be imposed that $F(m,n,1) = 2^m - 1$, and $F(m,n,r) = 0$ when at least one of the following conditions is true: $m \leq 0$, $n \leq 0$, $r \leq 0$, $r > \min\{m,n\}$.

Theorem 2.4. *The function $J(\cdot)$ can be recursively evaluated according to*[1]

$$\begin{aligned} J(m,n,r) = F(m,n,r) - \sum_{j=1}^{r-1} \binom{n}{j} \Big[\prod_{i=0}^{j-1} (2^m - 2^i) \Big] \, 2^{j(r-j)} \\ \cdot J(m-j, n-j, r-j). \end{aligned} \tag{2.19}$$

As in (2.18), also in (2.19) it must be imposed that $J(m,n,1) = 2^m - 1$, and $J(m,n,r) = 0$ when at least one of the following conditions is true: $m \leq 0$, $n \leq 0$, $r \leq 0$, $r > \min\{m,n\}$.

2.3.2 Computation of $K(m,n,g,u,r)$

In order to develop a formula for computing the number $K(m,n,g,u,r)$ of rank-r binary $(m \times n)$ matrices without zero columns, without independent columns, and such that the first g columns have rank equal to u, we use a method analog to that one used for function $J(\cdot)$. Let $M(m,n,g,u,r)$ be the number of rank-r binary $(m \times n)$ matrices without zero columns and such that the first g columns have rank equal to u. Then $K(m,n,g,u,r)$ is equal to the difference between $M(m,n,g,u,r)$ and the number of such matrices with at least one independent column.

[1]The summation in j can be actually always stopped at $j = r-1$, i.e. $J(m,n,r) = F(m,n,r) - \sum_{j=1}^{r-1} J^{(j)}(m,n,r)$ instead of $J(m,n,r) = F(m,n,r) - \sum_{j=1}^{r} J^{(j)}(m,n,r)$, except for full-rank matrices for which $m \geq n$. Since for binary $(k \times n)$ generator matrices it is always $k < n$, for the purposes of expected information functions computation, the summation in j up to $r-1$ is sufficient. This observation is also exploited in (2.21) and (2.29).

Lemma 2.5. *Let $T(m,n)$ be the total number of $(m \times n)$ binary matrices without zero columns, i.e. $T(m,n) = \sum_{r=1}^{\min\{m,n\}} F(m,n,r)$, and let $T(m,0) = 1$ by definition. Then:*

$$M(m,n,g,u,r) = F(m,g,u) \cdot \sum_{z=0}^{(n-g)-(r-u)} \binom{n-g}{z} T(u,z) \cdot 2^{u(n-g-z)} F(m-u,n-g-z,r-u). \tag{2.20}$$

For a correct working of the recursion (2.20), $M(m,n,g,u,r)$ must be set to 0 if one of the following conditions is true: $m \leq 0$, $n \leq 0$, $u < 0$, $r \leq 0$, $g < 0$, $g > n$, $u > \min\{m,g\}$, $r > \min\{m,n\}$, $r-u > \min\{n-g, m-u\}$, $u > r$, $\{g > 0, u = 0\}$, $\{g = 0, u > 0\}$, $\{g = n, u \neq r\}$, $\{m = n, r = m, g \neq u\}$. Special cases are: $M(m,n,g,u,r) = F(m,n,r)$ if $\{g = 0, u = 0\}$ or $\{g = n, u = r\}$, $M(m,n,g,u,r) = F(m,g,m)\ cdotT(m,n-g)$ if $u = m$, $M(m,n,g,u,r) = F(m,g,r) \cdot (2^r - 1)^{n-g}$ if $\{u = r, n > g\}$.

Theorem 2.5. *The function $K(\cdot)$ can be recursively evaluated according to*

$$K(m,n,g,u,r) = M(m,n,g,u,r) - \sum_{j=1}^{r-1} \sum_{l=0}^{\min\{u,j\}} \binom{g}{l} \binom{n-g}{j-l} \Big[\prod_{i=0}^{j-1} (2^m - 2^i)\Big] 2^{j(r-j)} \cdot K(m-j, n-j, g-l, u-l, r-j). \tag{2.21}$$

The function $K(m,n,g,u,r)$ is set to 0 in the same cases where $M(m,n,g,u,r)$ is set to 0. Special cases are: $K(m,n,g,u,r) = J(m,n,r)$ if $\{g = 0, u = 0\}$ or $\{g = n, u = r\}$, $K(m,n,g,u,r) = 2^m - 1$ if $\{u = 1, r = 1, g > 0, n-g > 0\}$.

In summary, for some k and n, $E_{\mathcal{G}_*^{(n,k)}}[\tilde{e}_g]$ can be computed from (2.15) and (2.16), where $J(\cdot)$ is obtained recursively from (2.19), and $K(\cdot)$ is obtained recursively from (2.21).

Example 2.2. *In Fig. 2.3, a detail of the EXIT function on the BEC for three binary $(31,21)$ linear block codes used as generalized check nodes (solid lines) is shown, as a function of the* a priori *mutual information $I_A = 1 - p$, for I_A ranging between 0.45 and 0.65. The minimum distances for the three check nodes are 2, 3 and 5, where the $d_{\min} = 5$ code is the $(31,21)$ narrow-sense binary BCH code. For each of the three codes, the EXIT function has been evaluated by first computing the information functions $\tilde{e}_g$ (which is still feasible for $(31,21)$ codes, even if time consuming), and then applying (2.3). In the same figure, the dashed line is the expected EXIT function on $\mathcal{G}_*^{(31,21)}$, evaluated by first computing the expected information functions $\mathbb{E}_{\mathcal{G}_*^{(31,21)}}[\tilde{e}_g]$ according to (2.15), (2.16), (2.19) and (2.21), and then applying (2.3). The code details for the minimum distance 2 and for the minimum distance 3 codes, as well as the analytic expressions of the check node EXIT functions associated to these codes are provided in Appendix A. In the same appendix, the expected information functions over $\mathcal{G}_*^{(31,21)}$ are given.*

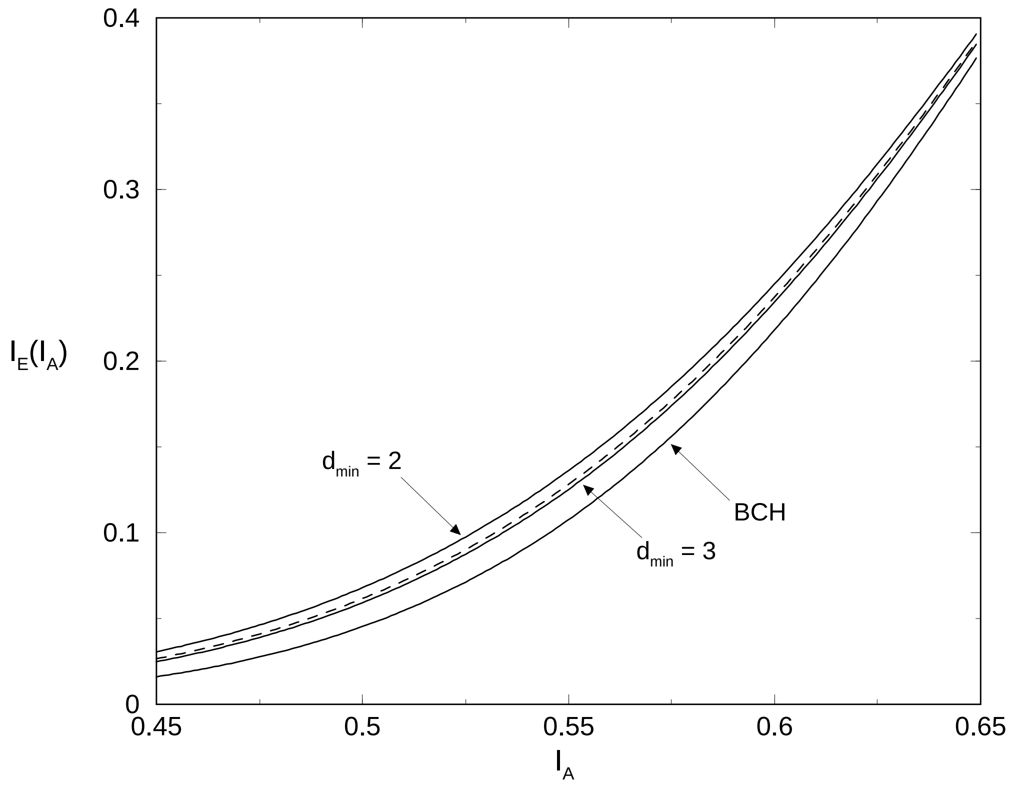


Figure 2.3: Plot detail of the EXIT functions of a $(31, 21)$ check node with $d_{\min} = 2$, a $(31, 21)$ check node with $d_{\min} = 3$ and the $(31, 21)$ BCH check node ($d_{\min} = 5$). Dashed line: $\mathcal{G}_*^{(31,21)}$ ensemble average.

The match between the solid curves and the dashed curve in Fig. 2.3 is quite good, despite the moderately short codeword length ($n = 31$). This fact indicates that the expected EXIT function can be confidently used, instead of the exact EXIT function, for longer component codes, for which the information functions remain unknown.

2.4 Expected Split Information Functions Computation

In this section, the technique for the evaluation of the expected information functions over $\mathcal{G}_*^{(n,k)}$, presented in the previous section, is extended to the variable node split information functions.

Let $\mathbf{G}$ be a $(k \times n)$ binary matrix from $\mathcal{G}_*^{(n,k)}$, and let $\mathcal{S}_{g,h}$ be a submatrix of $[\mathbf{G} \,|\, \mathbf{I}_k]$

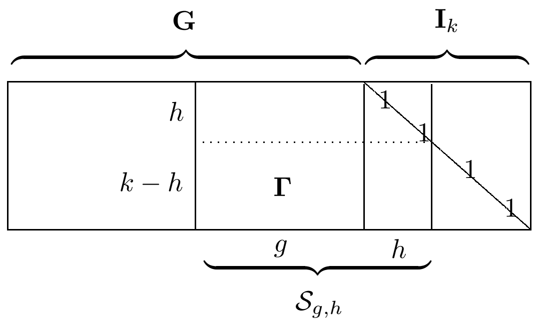


Figure 2.4: Scheme of matrix $[\mathbf{G}\,|\,\mathbf{I}_k]$ and definition of submatrix $\mathcal{S}_{g,h}$ for the evaluation of the expected split information functions.

obtained by selecting g columns in $\mathbf{G}$ and h columns in $\mathbf{I}_k$. Then:

$$\begin{aligned}\mathbb{E}_{\mathcal{G}_*^{(n,k)}}[\tilde{e}_{g,h}] &= \mathbb{E}_{\mathcal{G}_*^{(n,k)}}\Big[\sum_{\mathcal{S}_{g,h}} \operatorname{rank}(\mathcal{S}_{g,h})\Big] \\ &= \sum_{\mathcal{S}_{g,h}} \mathbb{E}_{\mathcal{G}_*^{(n,k)}}\left[\operatorname{rank}(\mathcal{S}_{g,h})\right] \\ &= \binom{n}{g}\binom{k}{h}\mathbb{E}_{\mathcal{G}_*^{(n,k)}}\left[\operatorname{rank}(\mathcal{S}_{g,h})\right]. \qquad (2.22)\end{aligned}$$

The last equality is due to the fact that, for matrices in $\mathcal{G}_*^{(n,k)}$, the expectation of the rank when selecting g columns in $\mathbf{G}$ and h columns in $\mathbf{I}_k$ is independent of the specific selected columns. Thus, $\mathcal{S}_{g,h}$ in (2.22) can be in principle any such submatrix.

Let us suppose that $\mathcal{S}_{g,h}$ in (2.22) is the submatrix composed of the last g columns of $\mathbf{G}$, and the first h columns of $\mathbf{I}_k$ (see Fig. 2.4). The probability $\Pr\{\operatorname{rank}(\mathcal{S}_{g,h}) = u\}$ that, for a randomly chosen matrix $\mathbf{G}$ in $\mathcal{G}_*^{(n,k)}$, the submatrix $\mathcal{S}_{g,h}$ has rank u, can be expressed as the number of matrices in $\mathcal{G}_*^{(n,k)}$ for which this property holds divided by the total number of matrices in $\mathcal{G}_*^{(n,k)}$. It is clear from Fig. 2.4 that the rank of $\mathcal{S}_{g,h}$ is at least h. In fact, the last h columns of this submatrix, i.e. the first h columns of $\mathbf{I}_k$, are linearly independent. Moreover, in order to have $\operatorname{rank}(\mathcal{S}_{g,h}) = u$, it is necessary and sufficient that the $((k-h)\times g)$ submatrix $\mathbf{\Gamma}$ in Fig. 2.4 has rank $u-h$. Hence, the binary matrices $\mathbf{G}\in\mathcal{G}_*^{(n,k)}$ satisfying $\operatorname{rank}(\mathcal{S}_{g,h}) = u$, are those ones for which $\operatorname{rank}(\mathbf{\Gamma}) = u-h$. Equivalently, they are those for which the submatrix intersection of the first g columns and the first $k-h$ rows has rank $u-h$ (since $\mathbf{\Gamma}$ can be any intersection of $k-h$ rows and g columns of $\mathbf{G}$).

The expectation of $\mathrm{rank}(\mathcal{S}_{g,h})$ in (2.22) can then be further developed as:

$$\begin{aligned}\mathbb{E}_{\mathcal{G}_*^{(n,k)}}\left[\mathrm{rank}(\mathcal{S}_{g,h})\right] &= \sum_{u=h}^{\min\{k,g+h\}} u \cdot \Pr\left\{\mathrm{rank}(\mathcal{S}_{g,h}) = u\right\} \\ &= \sum_{u=h}^{\min\{k,g+h\}} u \cdot \frac{\widetilde{K}(k,n,g,k-h,u-h,k)}{J(k,n,k)}, \qquad (2.23)\end{aligned}$$

where $\widetilde{K}(m,n,a,b,t,r)$ represents the number of rank-r $(m \times n)$ binary matrices without zero columns, without independent columns, and such that the submatrix intersection of the first a columns and first b rows has rank t. The function $\widetilde{K}(\cdot)$ is a generalization of the function $K(\cdot)$ investigated in the previous section. In fact, $K(m,n,g,u,r) = \widetilde{K}(m,n,g,m,u,r)$. In the following, a technique for the computation of $\widetilde{K}(\cdot)$ is derived.

2.4.1 Computation of $\widetilde{K}(m,n,a,b,t,r)$

Let $\widehat{K}(m,n,a,b,t,u,r)$ be the number of rank-r $(m \times n)$ binary matrices without zero columns, without independent columns, such that the submatrix intersection of the first a columns and b rows has rank t, and such that the submatrix composed of the first a columns has rank u. The function $\widetilde{K}(\cdot)$ can be expressed in terms of $\widehat{K}(\cdot)$, as

$$\widetilde{K}(m,n,a,b,t,r) = \sum_{u=1}^{\min\{m,a\}} \widehat{K}(m,n,a,b,t,u,r). \qquad (2.24)$$

The technique for the evaluation of $\widetilde{K}(\cdot)$ is based on a recursive formula developed for $\widehat{K}(\cdot)$, and presented in the next subsection. For any (m,n,a,b,t,r), $\widehat{K}(m,n,a,b,t,u,r)$ is first evaluated for $u = 1,\dots,\min\{m,a\}$, then $\widetilde{K}(m,n,a,b,t,r)$ is computed according to (2.24).

2.4.2 Computation of $\widehat{K}(m,n,a,b,t,r)$

In this section a recursion for the computation of $\widehat{K}(m,n,a,b,t,u,r)$ is developed. The proofs of lemmas and theorems claimed next are presented in Appendix 2.8.

Lemma 2.6. *Let $\widetilde{N}(m,n,p,t,r)$ be the number of rank-r $(m \times n)$ binary matrices, such that the rank of the first p rows is t. Then*

$$\widetilde{N}(m,n,p,t,r) = \prod_{j=0}^{t-1} \frac{(2^p - 2^j)(2^n - 2^j)}{2^t - 2^j} \cdot \prod_{j=0}^{r-t-1} \frac{(2^{m-p} - 2^j)(2^{n-t} - 2^j)}{2^{r-t} - 2^j} \cdot 2^{t(m-p)}. \qquad (2.25)$$

The function $\widetilde{N}(\cdot)$ is set to 0 if at least one of the following conditions is true: $m \leq 0$, $n \leq 0$, $p < 0$, $r < 0$, $t < 0$, $t > r$, $p > m$, $\{p = 0, t > 0\}$, $\{p = m, t \neq r\}$, $r > \min\{m, n\}$, $t > \min\{p, n\}$. Particular conditions are: $\widetilde{N}(m, n, p, 0, r) = \prod_{j=0}^{r-1}(2^{m-p} - 2^j)(2^n - 2^j)/(2^r - 2^j)$, $\widetilde{N}(m, n, m, r, r) = \prod_{j=0}^{r-1}(2^m - 2^j)(2^n - 2^j)/(2^r - 2^j)$.

Lemma 2.7. *Let $\widetilde{F}(m, n, p, t, r)$ be the number of rank-r $(m \times n)$ binary matrices, such that the rank of the first p rows is t, and without zero columns. Then*

$$\widetilde{F}(m, n, p, t, r) = \widetilde{N}(m, n, p, t, r) - \sum_{z=1}^{n-r} \binom{n}{z} \widetilde{F}(m, n - z, p, t, r). \tag{2.26}$$

The function $\widetilde{F}(\cdot)$ is set to 0 in the same cases as $\widetilde{N}(\cdot)$, or when $r = 0$. Special conditions are: $\widetilde{F}(m, n, p, t, r) = F(m - p, n, r)$ if $t = 0$, and $\widetilde{F}(m, n, p, t, r) = F(m, n, r)$ if $\{p = m, t = r\}$.

Lemma 2.8. *Let $\widehat{M}(m, n, a, b, t, u, r)$ be the number of rank-r $(m \times n)$ binary matrices without zero columns, such that the submatrix intersection of the first a columns and the first b rows has rank t, and such that the submatrix composed of the first a columns has rank u. Then*

$$\begin{aligned}\widehat{M}(m, n, a, b, t, u, r) &= \widetilde{F}(m, a, b, t, u) \\ &\cdot \sum_{z=0}^{(n-a)-(r-u)} \binom{n-a}{z} F(m - u, n - a - z, r - u) \cdot T(u, z) \cdot 2^{u(n-a-z)},\end{aligned} \tag{2.27}$$

where $T(m, n)$ is defined as in Lemma 2.5.

The function $\widehat{M}(\cdot)$ is set to 0 if at least one of the following conditions is true: $m \leq 0$, $n \leq 0$, $a < 0$, $b < 0$, $r \leq 0$, $t < 0$, $u < 0$, $t > r$, $t > u$, $u > r$, $a > n$, $b > m$, $r > \min\{m, n\}$, $t > \min\{a, b\}$, $u > \min\{m, a\}$, $\{a = 0, t > 0\}$, $\{b = 0, t > 0\}$, $\{a = 0, u > 0\}$, $\{a = n, u \neq r\}$, $\{b = m, t \neq u\}$, $\{a = n, b = m, t \neq r\}$. Special cases are: $\widehat{M}(m, n, a, b, t, u, r) = F(m, n, r)$ if $\{a = 0, t = 0, u = 0\}$ or $\{a = n, b = m, t = u = r\}$, $\widehat{M}(m, n, a, b, t, u, r) = \widetilde{F}(m, n, b, t, r)$ if $\{a = n, u = r, b < m\}$, $\widehat{M}(m, n, a, b, t, u, r) = M(m, n, a, u, r)$ if $\{b = m, t = u, a < n\}$, $\widehat{M}(m, n, a, b, t, u, r) = \widetilde{F}(m, a, b, t, u) \cdot T(u, n - a)$ if $\{u = r\}$.

Lemma 2.9. *Let $G(m, j, l, b, \gamma, \delta)$ be the number of rank-j $(m \times j)$ binary matrices (necessarily without zero columns), such that the submatrix intersection of the first l columns and the first b rows has has rank γ, and such that the submatrix composed of the first b rows has rank δ. Then*

$$G(m, j, l, b, \gamma, \delta) = \widetilde{F}(m, l, b, \gamma, l) \cdot \widetilde{F}(m - l, j - l, b - \gamma, \delta - \gamma, j - l) \cdot 2^{l(j-l)}. \tag{2.28}$$

The function $G(\cdot)$ is set to 0 when at least one of the following conditions is true: $m \leq 0$, $j \leq 0$, $l < 0$, $b < 0$, $m < j$, $\gamma > \min\{b, l\}$, $\delta > \min\{b, j\}$, $\{b = 0, \gamma > 0\}$, $\{b = 0, \delta > 0\}$, $\{l = j, b > 0, \delta \neq \gamma\}$, $\{l = 0, \gamma > 0\}$. Special conditions for $G(\cdot)$ are: $G(m, j, l, b, \gamma, \delta) = \widetilde{F}(m, j, b, \delta, j)$ if $\{l = 0, \gamma = 0, b > 0\}$ or $\{l = j, \delta = \gamma, b > 0\}$, and $G(m, j, l, b, \gamma, \delta) = \prod_{i=0}^{j-1}(2^m - 2^i)$ if $\{b = 0, \gamma = 0, \delta = 0\}$.

Theorem 2.6. *Let* $\gamma_{\max} = \min\{b, l\}$, $\delta_{\max} = \min\{b, j\}$ *and* $p_{\max} = \min\{b - \delta, a - l, t - \gamma\}$. *Then, the function* $\widehat{K}(\cdot)$ *can be recursively evaluated according to*

$$\begin{aligned}
\widehat{K}(m, n, a, b, t, u, r) = {} & \widehat{M}(m, n, a, b, t, u, r) \\
& - \sum_{j=1}^{r-1} \sum_{l=0}^{u} \binom{a}{l} \binom{n-a}{j-l} \sum_{\gamma=0}^{\gamma_{\max}} \sum_{\delta=\gamma}^{\delta_{\max}} G(m, j, l, b, \gamma, \delta) \cdot 2^{(r-j)(j-(\delta-\gamma))} \\
& \quad \cdot \sum_{p=0}^{p_{\max}} \widehat{K}(m-j, n-j, a-l, b-\delta, p, u-l, r-j) \\
& \quad\quad \cdot \prod_{i=0}^{t-\gamma-p-1} \frac{(2^{\delta-\gamma} - 2^i)(2^{u-l-p} - 2^i)}{2^{t-\gamma-p} - 2^i} \cdot 2^{((r-j)-(u-l-p))(\delta-\gamma)}. \qquad (2.29)
\end{aligned}$$

The function $\widehat{K}(\cdot)$ is set to 0 in the same cases as $\widehat{M}(\cdot)$. Special conditions are: $\widehat{K}(m, n, a, b, t, u, r) = J(m, n, r)$ if $\{a = 0, t = 0, u = 0\}$ or $\{a = n, b = m, t = u = r\}$, $\widehat{K}(m, n, a, b, t, u, r) = K(m, n, a, u, r)$ if $\{b = 0, t = 0\}$ or $\{b = m, t = u\}$.

Example 2.3. *In Fig. 2.5, a comparison is shown between the EXIT function on the BEC of a* $(16, 8)$ *variable node, with generator matrix randomly chosen from* $\mathcal{G}_*^{(16,8)}$, *and the expected EXIT function on* $\mathcal{G}_*^{(16,8)}$, *computed according to* (2.22), (2.23), (2.19), (2.24) *and* (2.29). *The EXIT functions are compared for* $q = 0, 0.2, 0.4, 0.6.0.8, 1$ *(the* $q = 1$ *case actually corresponds to the code used as a check node). Despite the short codeword length* ($n = 16$)*, the ensemble average confidently approximates the EXIT function of the specific code, for all the considered values of* q. *The expected EXIT function can be confidently used, instead of the exact EXIT function for those variable component codes for which the split information functions remain unknown.*

2.5 Numerical Results

In this section, some numerical results about GLDPC and D-GLDPC codes on the BEC are presented. These results are obtained by exploiting the technique for the evaluation of the expected CND and VND EXIT function, under the hypothesis of random component codes from the expurgated ensemble $\mathcal{G}_*^{(n,k)}$. Subsection 2.5.1 is focused on non-capacity-approaching GLDPC codes with a uniform check nodes structure, composed of

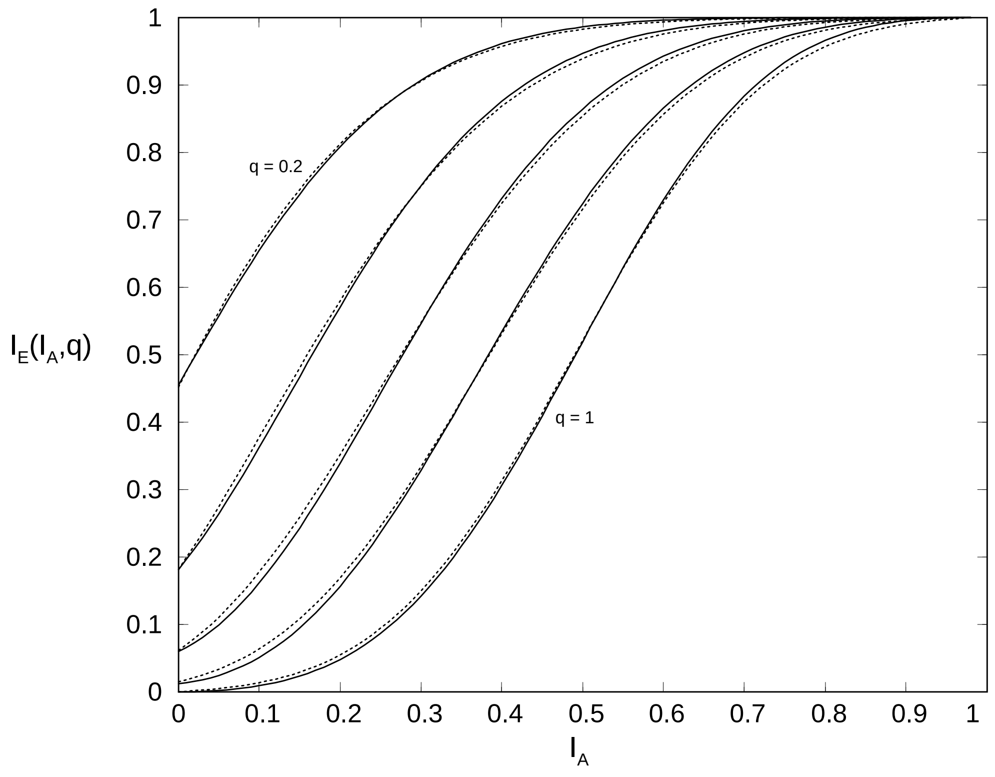


Figure 2.5: Comparison between the EXIT function of a $(16, 8)$ variable node, with generator matrix randomly chosen from $\mathcal{G}_*^{(16,8)}$ (solid), and the expected EXIT function over $\mathcal{G}_*^{(16,8)}$ (dotted), for of $q = 0.2, 0.4, 0.6, 0.8, 1.0$.

generalized check nodes, and a VND composed of length-2 repetition codes. It is shown that, for this class of codes, choosing check component codes with poor minimum distance provides a benefit with respect to good minimum distance codes. Subsection 2.5.2 moves to capacity-approaching GLDPC codes with hybrid check node structure and irregular VND. It is provided evidence that, in this case, generalized check component codes with good minimum distance properties are a better choice from a decoding threshold point of view. Finally, in Subsection 2.5.3, capacity approaching LDPC, GLDPC and D-GLDPC codes are compared, in terms of both asymptotic threshold and finite length performance of long random codes. These results reveal that random D-GLDPC codes can outperform standard LDPC codes and GLDPC codes in terms of both waterfall performance and error floor.

2.5.1 GLDPC Codes with Uniform Check Node Structure

Let us consider a GLDPC code, with $(31, 21)$ BCH codes as check nodes and length-2 repetition codes as variable nodes. The code rate is $R = 0.35484$, corresponding to a

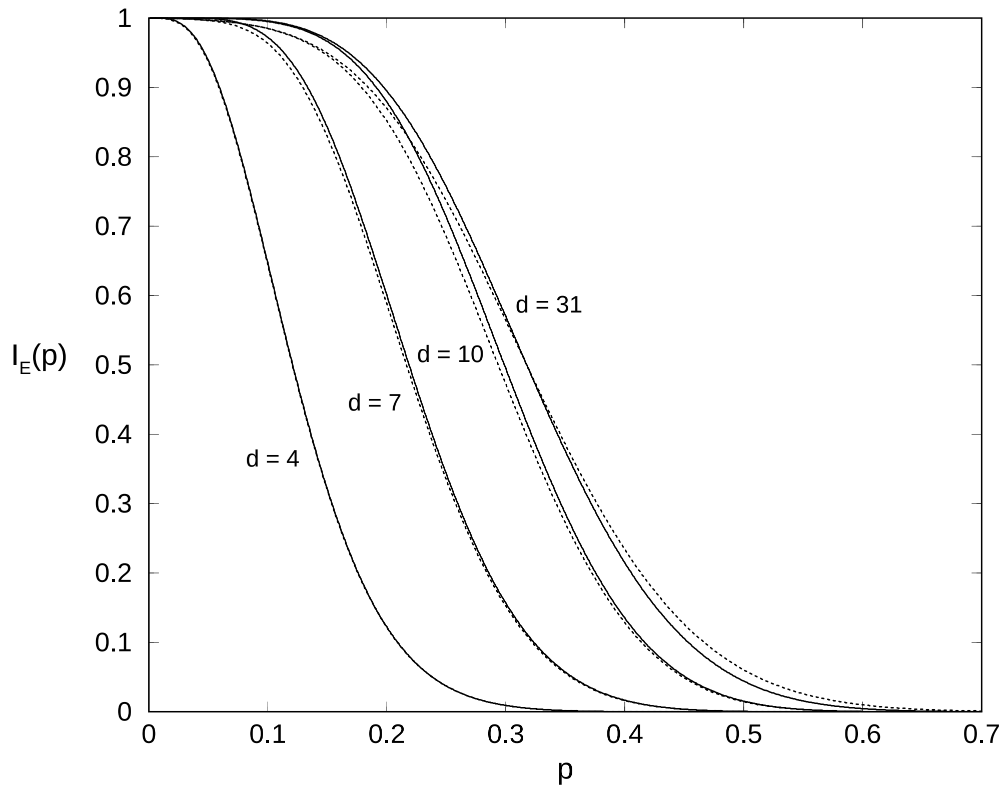


Figure 2.6: EXIT function of the $(31, 21)$ BCH code (solid) and expected EXIT function of the $(31, 21)$ expurgated ensemble (dotted) under d-bounded-distance decoding, for $d = 4, 7, 10, 31$.

Shannon limit on the BEC $q_{\text{Sh}} = 1 - R = 0.64516$. Let us assume d-bounded distance decoding (see Section 2.1.3) at the BCH check nodes. The GLDPC code threshold can be evaluated with the EXIT chart based on (2.4), by numerically evaluating the $(31, 21)$ BCH code information functions. The EXIT functions for $d = 4, 7, 10, 31$ are plotted in Fig. 2.6 (solid curves) as a function of the extrinsic channel erasure probability p, while the GLDPC thresholds are given in Table 2.1 (these threshold values are bounded away from $q_{\text{Sh}} = 0.64516$). Next, let us consider the same class of codes, under the hypothesis that the $(31, 21)$ check nodes are random linear block codes from $\mathcal{G}_*^{(31,21)}$. The corresponding expected EXIT functions under d-bounded-distance decoding are plotted in Fig. 2.6 (dotted curves), and the GLPDC thresholds are given in Table 2.1, for $d = 4, 7, 10, 31$.

The threshold values shown in Table 2.1 suggest the following consideration. From a threshold point of view, when the maximum number d of erasures faced by the decoder is bounded to a small value, it is convenient to use a component code with good minimum distance properties, like the BCH code. On the contrary, if this number is bounded to a higher value, or no bound is imposed at all ($d = 31$, which corresponds to MAP

d	BCH	expectation
4	0.21915	0.21879
7	0.35596	0.35407
10	0.46256	0.45929
31	0.50187	0.51426

Table 2.1: Thresholds on the BEC of GLPDC codes with $(31, 21)$ BCH check nodes, under d-bounded-distance decoding, and thresholds evaluated with the expected $(31, 21)$ EXIT function.

decoding), linear block codes must exist within $\mathcal{G}_*^{(n,k)}$ that guarantee a better GLDPC threshold than the BCH code. In fact, for sufficiently high d, the threshold computed assuming the expected CND EXIT function is better than the threshold guaranteed by the BCH check nodes. For this specific example, the crossover point between the ensemble average and the BCH code is at $d = 12$.

We actually found $(31, 21)$ linear block codes for which the GLDPC threshold is better than the ensemble average, under unconstrained MAP decoding. For instance, we generated a $(31, 21)$ code with $d_{\min} = 2$ for which the GLDPC threshold is 0.51920. We also generated a $(31, 21)$ linear code characterized by $d_{\min} = 3$, and we found a threshold equal to 0.51310 for the corresponding GLDPC code. This value is intermediate with respect to the thresholds corresponding to the BCH code and to the $d_{\min} = 2$ code. These codes are the ones whose EXIT functions have been already shown in Fig. 2.3 (the code details are available in Appendix A). Denoting by $T(\mathcal{C})$ the GLDPC code threshold corresponding to the choice of code $\mathcal{C}$ for the check nodes, we have $T(\mathcal{C}_{\text{BCH}}^{(31,21)}) < T(\mathcal{C}_{d_{\min}=3}^{(31,21)}) < T(\mathbb{E}[\mathcal{C}^{(31,21)}]) < T(\mathcal{C}_{d_{\min}=2}^{(31,21)})$. This reveals that using weak codes as check component codes for GLDPC codes with uniform check structure can be more convenient than using more powerful codes, like the BCH codes. This fact is confirmed by the simulation result shown in Fig. 2.7, where it is shown that the performance of a $(3999, 1419)$ GLDPC code using $(31, 21)$ BCH check nodes is worse than the performance of a similar code, having the same bipartite graph, but using the $d_{\min} = 2$ linear block code.

2.5.2 Capacity-Approaching GLDPC Codes with Hybrid Check Node Structure

Let us consider at first the optimization problem on the BEC for an LDPC distribution with the following constraints: variable nodes degree ranging from 2 up to 30, check nodes degree ranging from 3 up to 14, code rate $R = 1/2$. We solved the problem using the differential evolution (DE) algorithm [12, 51], for different starting distribution

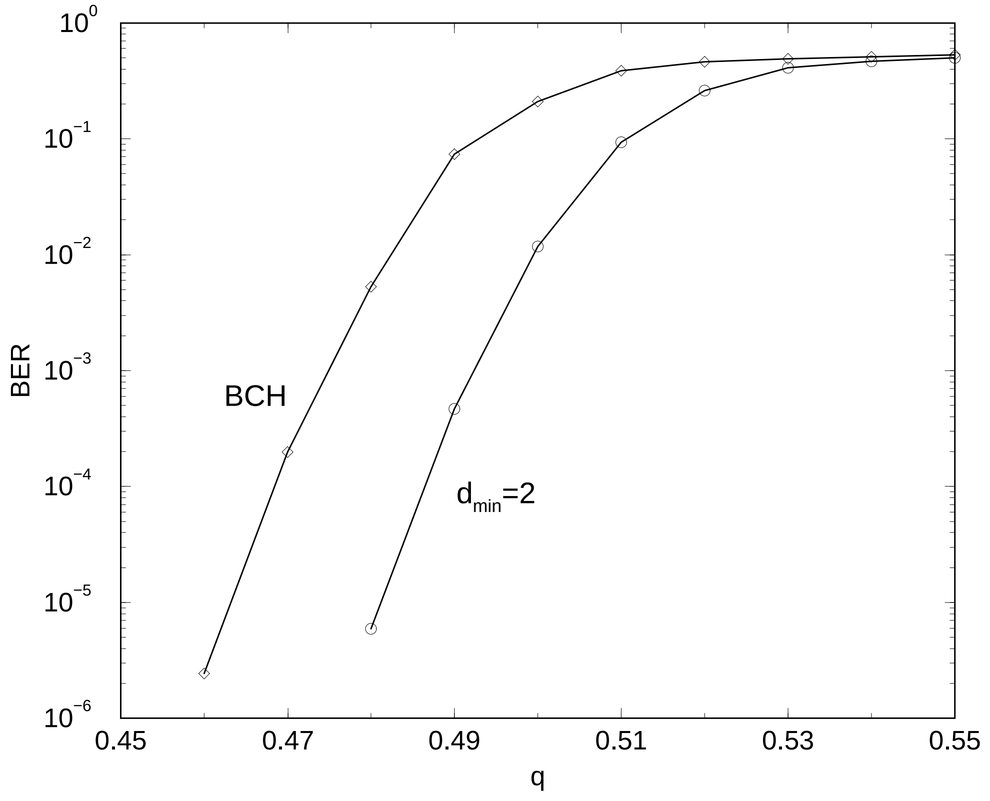


Figure 2.7: Comparison between the performance of a $(3999, 1419)$ GLDPC code with uniform check node structure composed of $(31, 21)$ binary BCH codes and a $(3999, 1419)$ GLDPC code with uniform check node structure composed of $(31, 21)$ linear block codes with $d_{\min} = 2$. The bipartite graph is the same for the two codes.

populations. The threshold of the best found distribution is $q^* = 0.49611$, quite close to the Shannon limit $1 - R = 0.5$, while the distribution is described in Table 2.2 (LDPC column).

Next, we solved the same optimization problem for a GLDPC code with a hybrid check node structure, composed of SPC codes and $(31, 21)$ linear block codes. We solved again the optimization problem through DE algorithm, assuming the same degree constraints for the variable nodes and for the SPC codes, and again $R = 1/2$. More specifically, we separately solved the problem in the cases where the $(31, 21)$ check nodes are represented by the binary BCH code with $d_{\min} = 5$, or by the $d_{\min} = i$ code ($i = 2, 3$) considered in previous subsection. We also solved the optimization problem using the expected EXIT function for the $(31, 21)$ expurgated ensemble. The optimal distribution corresponding to the choice of the $(31, 21)$ BCH code is described in Table 2.2 (GLDPC column), while the GLDPC thresholds are presented in Table 2.3.

For all the choices of the $(31, 21)$ generalized check nodes, the edges for the capacity-

Code type	LDPC	GLDPC	D-GLDPC$_1$	D-GLDPC$_2$
Variable Nodes				
Rep. 2	0.281884	0.270712	0.287410	0.275116
Rep. 3	0.123242	0.168858	0.161606	0.280377
Rep. 4	0.060701			
Rep. 5	0.106412	0.165958	0.293870	0.358294
Rep. 8		0.230227		
Rep. 9	0.084976			
Rep. 10	0.103547			
Rep. 29			0.217547	
Rep. 30	0.239238	0.164246		0.174163
$(31, 10)$			0.039568	0.030803
Check Nodes				
SPC 8	0.925027			
SPC 9		0.912838	0.871398	0.381799
SPC 10	0.074973			0.458201
$(31, 21)$-BCH		0.087162	0.128602	0.160000
Thresholds				
q^*	0.49611	0.49671	0.49759	0.49655

Table 2.2: Capacity-approaching rate-1/2 LDPC, GLPDC, D-GLDPC$_1$ and D-GLDPC$_2$ edge distributions.

LDPC	0.49611
GLDPC, $d_{\min} = 2$	0.49627
GLDPC, expectation	0.49639
GLDPC, $d_{\min} = 3$	0.49648
GLDPC, BCH	0.49671

Table 2.3: Decoding thresholds for capacity-approaching rate-1/2 LDPC and GLPDC distributions.

approaching distribution are connected to the SPC nodes with degree 9 and to the $(31, 21)$ nodes. Moreover, the optimized distributions (both variable and check) are very similar in all cases. The fraction of edges connected to the $(31, 21)$ codes ranges from about 5.73% for the $d_{\min} = 2$ code to about 8.72% for the BCH code. Some considerations about these results are presented next.

First, it is possible to improve the threshold of an LDPC code by letting unchanged the constraints for the degrees of the repetition and SPC nodes, introducing check component codes different from the SPC codes, and modifying accordingly the edge distribution. The presented example is even more meaningful, since the starting LDPC distribution is already capacity-approaching. This better GLDPC threshold is achieved with a relatively small fraction of generalized check nodes. In fact, the fraction of BCH check nodes is about 2.70%, which results in a small increase in terms of decoding complexity with respect to the LDPC code.

Second, when considering hybrid check node structures instead of uniform ones, using more powerful codes like the BCH codes leads to better thresholds. For the hybrid case it results $T(\mathcal{C}^{(31,21)}_{d_{\min}=2}) < T(\mathbb{E}[\mathcal{C}^{(31,21)}]) < T(\mathcal{C}^{(31,21)}_{d_{\min}=3}) < T(\mathcal{C}^{(31,21)}_{\mathrm{BCH}})$, which is the opposite of what found for a uniform check node structure. The reason is that the role of weak codes (necessary for obtaining good thresholds) is now played by the SPC codes.

Third, when combined with differential evolution, the developed technique for evaluating the expected EXIT function of generalized check nodes on $\mathcal{G}^{(n,k)}_*$ leads to an optimal distribution and threshold which closely matches that obtained for the choice of the BCH code. Hence, this technique can be confidently used not only for the threshold analysis, but also for the purposes of GLDPC distributions optimization.

2.5.3 Capacity Approaching D-GLDPC Codes

We solved the same optimization problem considered in the previous subsection for a $R = 1/2$ D-GLDPC coding scheme. The same check node types as for the GLDPC code were considered; in addition, hybrid variable nodes were allowed, composed by a mixture of repetition codes with the same degrees as for the LDPC code, plus $(31, 10)$ random linear block codes from $\mathcal{G}^{(31,10)}_*$. The choice of $(31, 10)$ codes as variable nodes was dictated by the heuristic guideline to use codes with same length and dual dimension at different sides of the bipartite graph[2]. The random codes approach was followed since the direct computation of the split information functions for a specific $(31, 10)$ linear block code, e.g. the dual of the $(31, 21)$ BCH code, is not feasible in terms of computation time. The expected EXIT function for the generalized variable nodes over $\mathcal{G}^{(31,10)}_*$ was evaluated according to the method presented in Section 2.3. The expected split information functions over $\mathcal{G}^{(31,10)}_*$ are listed in Appendix B. The capacity-approaching D-GLDPC distribution returned by differential evolution is shown in Table 2.2, together with its threshold, and called D-GLDPC$_1$ distribution. Some considerations are provided next.

First, the D-GLDPC$_1$ distribution has the best threshold. Hence, under the described constraints, using generalized variable nodes together with generalized check nodes permits to increase the threshold with respect to GLDPC codes, even in a case study where the GLDPC threshold is already very close to capacity. This better D-GLDPC threshold is achieved with a small increase of the fraction of generalized check nodes, and with a small fraction of generalized variable nodes. In fact, the fraction of BCH check nodes and

[2]No proof has actually been developed so far that this is an optimal choice.

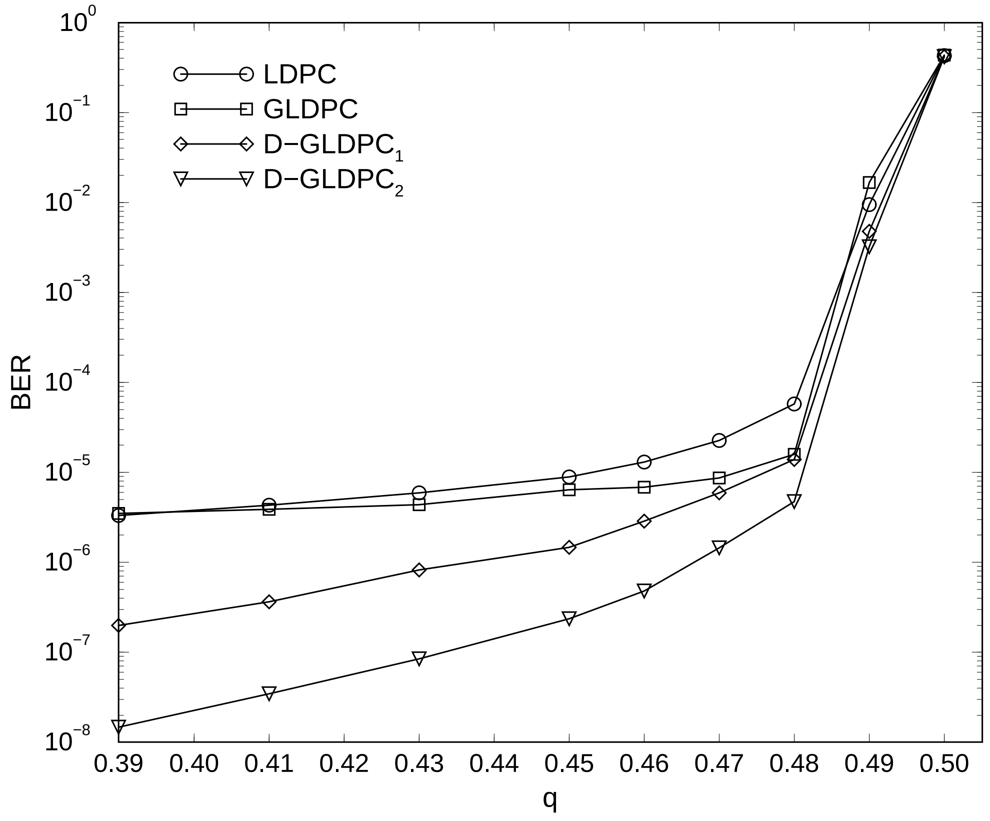


Figure 2.8: Performance of $R = 1/2$ LDPC, GLDPC and D-GLDPC codes of length $N = 128000$ on the BEC.

$(31, 10)$ variable nodes for the D-GLDPC code are about 4.11% and 0.48% respectively, which results a small increase in terms of decoding complexity with respect to the GLDPC code.

Second, the larger fraction of BCH check nodes in the D-GLDPC distribution than in the GLDPC one (4.11% vs 2.70%) suggests the following. The original idea behind GLDPC codes was to strengthen the check nodes, by introducing powerful generalized check nodes [20]. This approach can provide good minimum distance properties, but the drawback is a lowering of the overall code rate, which reveals unacceptable in many cases [34][37]. The key idea behind D-GLDPC codes is to overcome this rate loss by introducing weaker component codes at the variable nodes. In the case study under analysis, the introduction of $(31, 10)$ generalized variable nodes is able to partly compensate the rate loss due to the $(31, 21)$ BCH check nodes; then, it is possible to use a larger number of powerful erasure correcting codes among the check nodes, without loosing performance in terms of threshold.

In order to support these asymptotic performance results, we simulated long and randomly constructed codes, designed according to the distributions presented in Table

2.2. Random codes were simulated, because random connections between the variable and check nodes were assumed to develop the expected EXIT functions for the VND and CND. For the D-GLDPC coding scheme, the dual of the $(31, 21)$ BCH code (in systematic representation) was used at the generalized variable nodes. In Fig. 2.8, the performance in terms of post-decoding bit erasure rate (BER) is shown for codes of length $N = 128000$. As expected, the LDPC code exhibits bad error floor performance, due to the poor minimum distance of capacity approaching distributions (for this distribution it is $\lambda'(0)\rho'(1) = 2.015$). This high error floor is not improved when considering the GLDPC code construction. However, the D-GLDPC code exhibits both a slightly better waterfall performance (according to the slightly better threshold) and an error floor which is about one order of magnitude lower than that of LDPC and GLDPC codes. This result suggests that capacity approaching D-GLDPC codes can be constructed, characterized by better properties in terms of both waterfall and error floor performance than LDPC and GLDPC codes, and with limited increase of decoding complexity.

Using generalized variable nodes enables to use a larger number of generalized (powerful) check nodes than for GLDPC codes, providing better minimum distance properties, while keeping a better threshold. In order to construct D-GLDPC codes with better minimum distance properties than the D-GLDPC$_1$ code, and still a good threshold, we tried the following approach. We ran the DE algorithm again for the D-GLDPC distribution, with the additional constraint of a lower bound on the fraction of edges towards the generalized check nodes. More specifically, we ran the DE optimization with the further constraint $\rho_{\text{BCH}} \geq \rho_{\text{BCH}}^{\min}$. The obtained distribution for $\rho_{\text{BCH}}^{\min} = 0.16$ and the corresponding threshold are presented in Table 2.2, in the D-GLDPC$_2$ column. The threshold is still better than that of the LDPC distribution.

The performance curve on the BEC obtained for a random $N = 128000$ code, designed according to the D-GLDPC$_2$ distribution, is shown in Fig. 2.8. We observe an improvement in the error floor region, about one order of magnitude with respect to the D-GLDPC$_1$ code, and about two orders of magnitude with respect to the GLDPC and LDPC codes, with no loss in terms of waterfall performance.

2.6 Conclusion

In this chapter, a technique for the analysis of D-GLDPC codes on the BEC has been proposed. This technique assumes that the variable and check component codes are random codes with minimum distance at least 2. It permits to evaluate the expected EXIT function for the variable and check node decoders, thus enabling for the EXIT chart analysis. The core of this method is the computation of the expected (split) information functions over the expurgated ensemble of (n, k) linear block codes with minimum distance

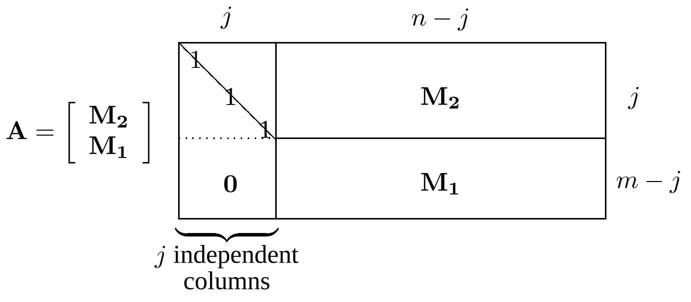


Figure 2.9: Specific matrix for the computation of function $J(\cdot)$.

at least 2, where the expurgation guarantees a correct application of the EXIT charts analysis. The expected (split) information function computation exploits some formulas for obtaining the exact number of binary matrices with specific properties. The proposed analysis method has been combined with the differential evolution algorithm, in order to develop optimal D-GLDPC distributions. Focusing on random, capacity approaching codes, it has been shown that D-GLDPC codes can be constructed, outperforming LDPC and GLDPC codes in terms of *both* waterfall and error floor. Moreover, by lower bounding the fraction of edges towards the generalized check nodes, D-GLDPC codes have been designed, which gain two orders of magnitude with respect to the LDPC and GLDPC codes in terms of error floor, with no sacrifice in terms of waterfall performance.

2.7 Proofs of Lemmas and Theorems of Section 2.3

Proof of Lemma 2.4: $F(m,n,r)$ is equal to the total number of rank-r $(m \times n)$ binary matrices minus the number of rank-r $(m \times n)$ binary matrices with at least one zero column. The total number of rank-r $(m \times n)$ binary matrices is given by $\prod_{j=0}^{r-1}(2^m - 2^j)(2^n - 2^j)/(2^r - 2^j)$. The number of rank-$r$ $(m \times n)$ binary matrices with exactly z zero columns $(z \leq n-r)$ is expressed by $\binom{n}{z} F(m, n-z, r)$. □

Proof of Theorem 2.4: $J(m,n,r)$ can be computed as $F(m,n,r) - \sum_{j=1}^{r} J^{(j)}(m,n,r)$, where $J^{(j)}(m,n,r)$ is the number of rank-r binary $(m \times n)$ matrices without zero columns and with j independent columns.

There are $\binom{n}{j}$ possible positions for the j independent columns, and the number of choices of the j independent columns is $\prod_{i=0}^{j-1}(2^m - 2^i)$. Hence we have

$$J^{(j)}(m,n,r) = \binom{n}{j} \left[\prod_{i=0}^{j-1} (2^m - 2^i) \right] \cdot D^{(j)}(m, n-j),$$

where $D^{(j)}(m, n-j)$ is the residual number of $(m \times (n-j))$ binary matrices (that must have no zero columns and no independent columns). We prove next that $D^{(j)}(m, n-j) =$

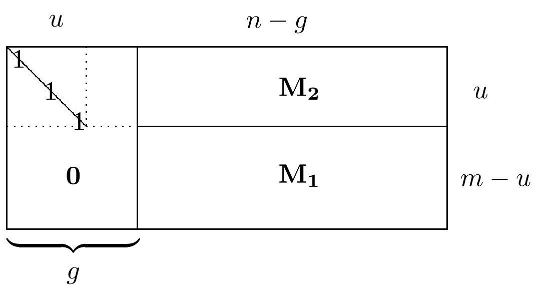


Figure 2.10: Specific matrix for the computation of function $M(\cdot)$.

$J(m-j, n-j, r-j) \cdot 2^{j(r-j)}$, thus leading to the recursion (2.19).

Since $D^{(j)}(m, n-j)$ is independent of the position and choice of the j independent columns, we can reason on the specific matrix shown in Fig. 2.9, where the matrix $\mathbf{A}$ is defined. With respect to this choice, $D^{(j)}(m, n-j)$ is the number of choices of the last $n-j$ columns.

The rank of such a matrix is equal to $j + \text{rank}(\mathbf{M_1})$. Thus $\mathbf{M_1}$ must have rank $r-j$, and it must have no independent columns (in fact, since the total rank is $j+\text{rank}(\mathbf{M_1})$, an independent column for $\mathbf{M_1}$ would be independent for the whole matrix). Moreover, since each of the last $n-j$ columns must be independent of each of the first j columns, each column of $\mathbf{M_1}$ must have at least one 1. Hence, the number of $\mathbf{M_1}$ matrices is equal to $J(m-j, n-j, r-j)$. Since the first j columns are independent, removing them from the matrix must lead to a rank $r-j$. Then, $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{M_1}) = r-j$. Then, any row in $\mathbf{M}_2$ must be a linear combination of rows in $\mathbf{M}_1$. The total number of such combinations is $2^{j(r-j)}$. Hence it results $D^{(j)}(m, n-j) = J(m-j, n-j, r-j) \cdot 2^{j(r-j)}$, □

Proof of Lemma 2.5: The function $M(m, n, g, u, r)$ can be expressed as $F(m, g, u)$ (number of choices of the first g columns) times the number of $(m \times (n-g))$ binary matrices without zero columns and such that the overall rank is r. Since this number is independent of the specific choice of the first g columns, we can reason on the specific matrix of Fig. 2.10. In order to have an overall rank r, we must have $\text{rank}(\mathbf{M_1}) = r-u$. Denoting by z the number of zero columns in $\mathbf{M_1}$, the number of $\mathbf{M_1}$ matrices can be expressed as $\binom{n-g}{z} F(m-u, n-g-z, r-u)$. Since $\text{rank}(\mathbf{M_1}) = r-u$, the number z of zero columns in $\mathbf{M_1}$ cannot exceed $s = (n-g)-(r-u)$. The only constraint on $\mathbf{M_2}$ is that at least one 1 must be present in each column of $\mathbf{M_2}$ corresponding to a zero column of $\mathbf{M_1}$. Thus the number of $\mathbf{M_2}$ matrices corresponding to a $\mathbf{M_1}$ matrix with z zero columns is $T(u, z)\, 2^{u(n-g-z)}$, where $T(\cdot)$ is defined in the statement of the lemma, and where $T(u, 0)$ must be set to 1 (no zero columns in $\mathbf{M_1}$). Then we obtain (2.20). □

Proof of Theorem 2.5: In a similar way as for function $J(\cdot)$, we have $K(m, n, g, u, r) = M(m, n, g, u, r) - \sum_{i=1}^{r-1} K^{(j)}(m, n, g, u, r)$, where $K^{(j)}(m, n, g, u, r)$ is the number of rank-r

$$\mathbf{A} = \begin{bmatrix} \mathbf{M_3} \\ \mathbf{M_1} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{M_5} & \mathbf{M_6} \\ \mathbf{M_3} & \mathbf{M_4} \\ \mathbf{M_1} & \mathbf{M_2} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{M_5} & \mathbf{M_6} \\ \mathbf{M_3} & \mathbf{M_4} \end{bmatrix}$$

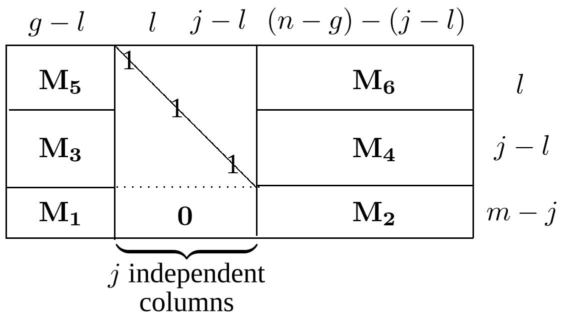


Figure 2.11: Matrix for the computation of function $K(\cdot)$.

$(m \times n)$ binary matrices without zero columns, with the first g columns of rank u, and with j independent columns.

Let the number of independent columns among the first g columns be $l \le \min\{u, j\}$, and the number of independent columns among the last $n-g$ columns be $j-l$. The number of possible positions of the independent columns is $\binom{g}{l}\binom{n-g}{j-l}$, while the number of choices of the j independent columns is $\prod_{i=0}^{j-1}(2^m - 2^j)$. We can reason on a specific position and choice of the independent columns. The specific choice is depicted in Fig. 2.11, where matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are defined. We have $K^{(j)}(m,n,g,u,r) = \sum_{l=0}^{\min\{u,j\}} \binom{g}{l}\binom{n-g}{j-l}\left[\prod_{i=0}^{j-1}(2^m - 2^i)\right] \cdot N_B$, with N_B number of $\mathbf{B}$ matrices for each choice of the j independent columns. We prove next that the number of $[\mathbf{M_1}|\mathbf{M_2}]$ possible matrices is $K(m-j, n-j, g-l, u-l, r-j)$ and, for each choice of $[\mathbf{M_1}|\mathbf{M_2}]$, the number of $\mathbf{C}$ matrices is $2^{j(r-j)}$, so that

$$N_B = 2^{j(r-j)} \cdot K(m-j, n-j, g-l, u-l, r-j),$$

from which it follows the recursion (2.21).

The rank of the overall matrix is equal to j+rank($[\mathbf{M_1}|\mathbf{M_2}]$). Consequently, rank($[\mathbf{M_1}|\mathbf{M_2}]$) $= r-j$. Furthermore, $[\mathbf{M_1}|\mathbf{M_2}]$ must also have no independent columns (since the overall rank is $j+$ rank($[\mathbf{M_1}|\mathbf{M_2}]$), any such column would be independent also for the overall matrix). $[\mathbf{M_1}|\mathbf{M_2}]$ must also have at least one 1 for each column due to the linear independence between the j independent columns and all the columns of $\mathbf{B}$. Finally, it must be rank($\mathbf{M_1}$) $= u-l$. This condition can be obtained in the following way. Since rank($\mathbf{B}$) $=$ rank($[\mathbf{M_1}|\mathbf{M_2}]$) $= r-j$, each row in $\mathbf{C}$ must be a linear combination of rows in $[\mathbf{M_1}|\mathbf{M_2}]$. This implies in particular that each row in $\mathbf{M_3}$ must be a linear combination of rows in $\mathbf{M_1}$, i.e. rank($\mathbf{A}$) $=$ rank($\mathbf{M_1}$). The rank of the first g columns is equal to $l+$ rank($\mathbf{A}$). Since this rank must be equal to u, it follows rank($\mathbf{A}$) $= u-l$, i.e. rank($\mathbf{M_1}$) $= u-l$. Then, the number of $[\mathbf{M_1}|\mathbf{M_2}]$ matrices is $K(m-j, n-j, g-l, u-l, r-j)$.

Since each row of $\mathbf{C}$ is a linear combination of rows of $[\mathbf{M_1}|\mathbf{M_2}]$, and since rank($[\mathbf{M_1}|\mathbf{M_2}]$) $=$

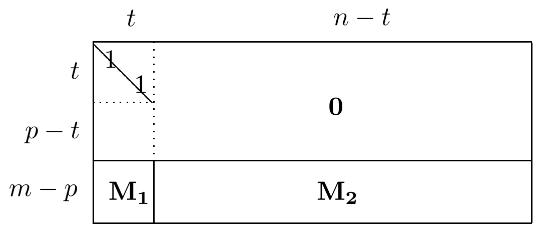


Figure 2.12: Matrix for the computation of function $\widetilde{N}(\cdot)$.

$r-j$, then for each choice of $[\mathbf{M_1}|\mathbf{M_2}]$ there are $2^{j(r-j)}$ matrices $\mathbf{C}$ allowed. □

2.8 Proofs of Lemmas and Theorems of Section 2.4

Proof of Lemma 2.6: The number of choices of the first p rows is $\prod_{j=0}^{t-1}(2^p-2^j)(2^n-2^j)/(2^t-2^j)$. Since the number of choices of the last $m-p$ rows does not depend on the structure of the first p rows, the specific matrix depicted in Fig. 2.12 can be considered. The rank of the $(m \times n)$ matrix is given by $t+\text{rank}(\mathbf{M_2})$: then, $\text{rank}(\mathbf{M_2}) = r-t$, and the number of $\mathbf{M_2}$ matrices is $\prod_{j=0}^{r-t-1}(2^{m-p}-2^j)(2^{n-t}-2^j)/(2^{r-t}-2^j)$. Since there are no constraints on the choice of $\mathbf{M_1}$, the number of $\mathbf{M_1}$ matrices for each choice of the first p columns and for each choice of $\mathbf{M_2}$ is $2^{t(m-p)}$. □

Proof of Lemma 2.7: $\widetilde{F}(m,n,p,t,r)$ is equal to the total number of rank-r binary matrices such that the rank of the first p rows is t, $\widetilde{N}(m,n,p,t,r)$, minus the number of such matrices with z zero columns, for $z=1,\dots,n-r$. The number of rank-r $(m \times n)$ binary matrices such that the rank of the first p rows is t and with exactly z zero columns $(z \le n-r)$ is expressed by $\binom{n}{z}\widetilde{F}(m,n-z,p,t,r)$. □

Proof of Lemma 2.8: Let $\mathbf{M}_1$ be the submatrix composed of the first a columns, and $\mathbf{M}_2$ be the submatrix composed of the last $n-a$ columns. The number of $\mathbf{M}_1$ matrices is $\widetilde{F}(m,a,b,t,u)$, expressed by Lemma 2.7. The number of $\mathbf{M}_2$ submatrices is independent of the specific choice of $\mathbf{M}_1$. A convenient choice of $\mathbf{M}_1$ is depicted in Fig. 2.13, where $\mathbf{M}_2$ is partitioned into the three submatrices $\mathbf{M}_2^{(1)}$, $\mathbf{M}_2^{(2)}$, $\mathbf{M}_2^{(3)}$, and the matrix $\mathbf{A}$ is defined. In order to have a total rank r, it must be $\text{rank}(\mathbf{A}) = r-u$. Denoting by z the number of zero columns in $\mathbf{A}$, the number of $\mathbf{A}$ matrices is $\sum_{z=0}^{z_{\max}}\binom{n-a}{z}F(m-u,n-a-z,r-u)$, where $z_{\max}=(n-a)-(r-u)$ (because we need at least $r-u$ non-zero columns for $\mathbf{A}$). Since the overall $(m \times n)$ matrix must have no zero columns, the total number of choices for the z columns of $\mathbf{M}_2^{(2)}$, corresponding to the z zero columns of some choice of $\mathbf{A}$, is $T(u,z)$. Moreover, no constraint exists on the choice of the $n-a-z$ columns of $\mathbf{M}_2^{(2)}$ corresponding to the non-zero columns of $\mathbf{A}$. Then, this number is $2^{u(n-a-z)}$. □

Proof of Lemma 2.9: Since the rank of the $(m \times j)$ matrix is equal to j, all the columns

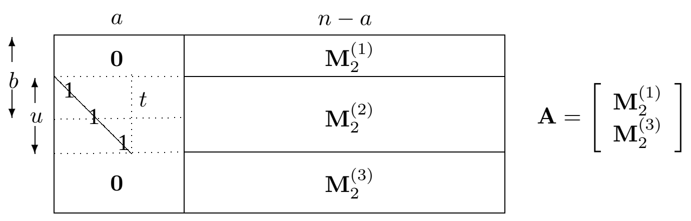

a
n − a
b
u
t
0
1
1
1
0
$\mathbf{M}_2^{(1)}$
$\mathbf{M}_2^{(2)}$
$\mathbf{M}_2^{(3)}$
$\mathbf{A} = \begin{bmatrix} \mathbf{M}_2^{(1)} \\ \mathbf{M}_2^{(3)} \end{bmatrix}$

Figure 2.13: Specific choice of the submatrix $\mathbf{M_1}$ for the computation of $\widehat{M}(m,n,a,b,t,u,r)$.

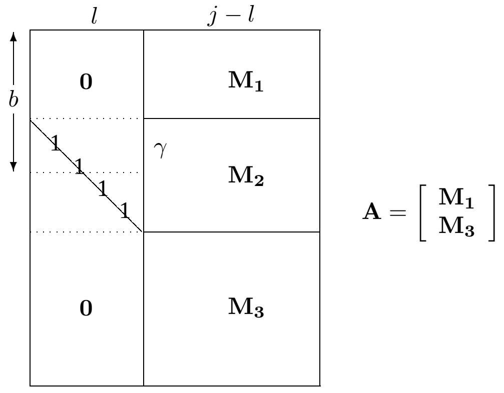

l
j − l
b
γ
0
1
1
1
1
0
$\mathbf{M_1}$
$\mathbf{M_2}$
$\mathbf{M_3}$
$\mathbf{A} = \begin{bmatrix} \mathbf{M_1} \\ \mathbf{M_3} \end{bmatrix}$

Figure 2.14: Specific choice of the $(m \times j)$ matrix for the computation of $G(m,j,l,b,\gamma,\delta)$.

must be linearly independent. Hence, the number of possible choices for the first l columns is $\widetilde{F}(m,l,b,\gamma,l)$, with $\widetilde{F}(\cdot)$ defined in Lemma 2.7. The number of possible choices for the last $j-l$ columns is independent of the specific choice of the first l columns. A convenient choice is depicted in Fig. 2.14, where the last $n-l$ columns are decomposed into three the submatrices $\mathbf{M_1}$, $\mathbf{M_2}$ and $\mathbf{M_3}$, and where $((m-l)\times(j-l))$ matrix $\mathbf{A}$ is defined. We prove next that, for each choice of the first l columns, the number of $\mathbf{A}$ matrices is $\widetilde{F}(m-l,j-l,b-\gamma,\delta-\gamma,j-l)$ and, for each choice of the first l columns and $\mathbf{A}$ matrix, the number of $\mathbf{M_2}$ matrices is $2^{l(j-l)}$.

The total rank of the $(m \times j)$ matrix is equal to $l + \text{rank}(\mathbf{A})$: then, it must be $\text{rank}(\mathbf{A}) = j-l$. Moreover, in order to have a rank δ for the first b rows, it must be $\text{rank}(\mathbf{M_1}) = \delta-\gamma$. Since all the j columns must be independent, $\mathbf{A}$ must have no zero columns. Then, the number of $\mathbf{A}$ matrices is $\widetilde{F}(m-l,j-l,b-\gamma,\delta-\gamma,j-l)$. Since any choice is allowed for $\mathbf{M_2}$, the number of such submatrices is $2^{l(j-l)}$. □

Proof of Theorem 2.6: The searched number of binary matrices can be obtained as

$$\widehat{M}(m,n,a,b,t,u,r) - \sum_{j=1}^{r-1} \widehat{K}^{(j)}(m,n,a,b,t,u,r),$$

where $\widehat{K}^{(j)}(m,n,a,b,t,u,r)$ is the number of rank-r $(m \times n)$ binary matrices without

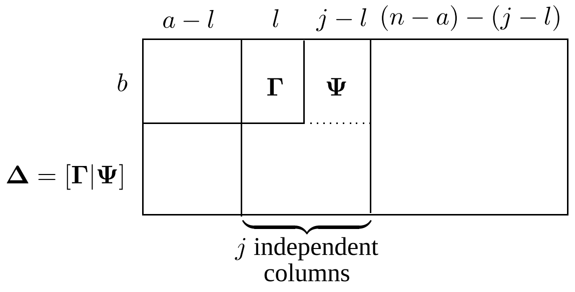
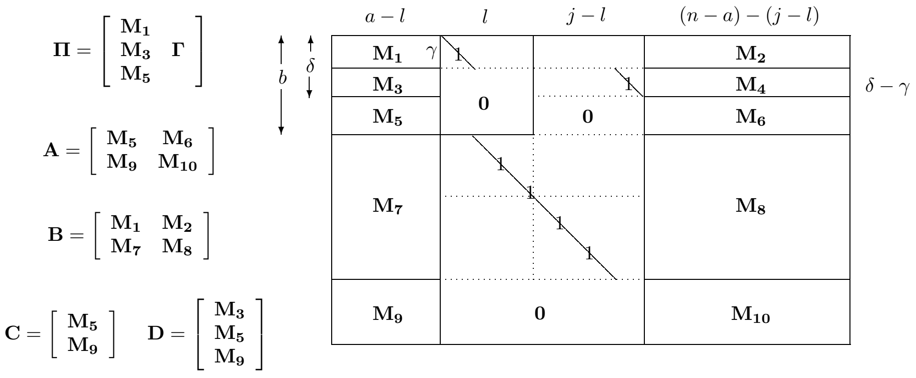


Figure 2.15: Specific position of the j independent columns, with $\text{rank}(\mathbf{\Gamma}) = \gamma$ and $\text{rank}(\mathbf{\Delta}) = \delta$.



$\mathbf{\Pi} = \begin{bmatrix} \mathbf{M_1} & \\ \mathbf{M_3} & \mathbf{\Gamma} \\ \mathbf{M_5} & \end{bmatrix}$
$\mathbf{A} = \begin{bmatrix} \mathbf{M_5} & \mathbf{M_6} \\ \mathbf{M_9} & \mathbf{M_{10}} \end{bmatrix}$
$\mathbf{B} = \begin{bmatrix} \mathbf{M_1} & \mathbf{M_2} \\ \mathbf{M_7} & \mathbf{M_8} \end{bmatrix}$
$\mathbf{C} = \begin{bmatrix} \mathbf{M_5} \\ \mathbf{M_9} \end{bmatrix}$
$\mathbf{D} = \begin{bmatrix} \mathbf{M_3} \\ \mathbf{M_5} \\ \mathbf{M_9} \end{bmatrix}$
$a-l$
l
$j-l$
$(n-a)-(j-l)$
b
δ
γ
$\delta-\gamma$
$\mathbf{M_1}$
$\mathbf{M_2}$
$\mathbf{M_3}$
$\mathbf{M_4}$
$\mathbf{M_5}$
$\mathbf{M_6}$
$\mathbf{M_7}$
$\mathbf{M_8}$
$\mathbf{M_9}$
$\mathbf{M_{10}}$
$\mathbf{0}$
1

Figure 2.16: Specific position of the j independent columns, with $\text{rank}(\mathbf{\Gamma}) = \gamma$ and $\text{rank}(\mathbf{\Delta}) = \delta$.

zero columns, such that the rank of the submatrix intersection of the first a columns and the first b rows is t, such that the rank of the first a columns is u and with exactly j independent columns. For each j, let l be the number of independent columns among the first a columns, and $j-l$ the number of independent columns among the last $n-a$ columns. Since the rank of the first a columns must be u, it is $l \in \{0, \dots, u\}$. For l, there are $\binom{a}{l}\binom{n-a}{j-l}$ possible positions for the j independent columns.

Let us assume that the j independent columns are the last l out of the first a columns, and the first $j-l$ out of the last $n-a$ columns, as shown in Fig. 2.15, where the matrices $\mathbf{\Gamma}$, $\mathbf{\Psi}$ and $\mathbf{\Delta}$ are defined. Denoting by γ the rank of the $(b \times l)$ matrix $\mathbf{\Gamma}$ and by δ the rank of the $(b \times j)$ matrix $\mathbf{\Delta}$, it is $\gamma \in \{0, \dots, \min\{b, l\}\}$ and $\delta \in \{\gamma, \dots, \min\{b, j\}\}$.

For fixed γ and δ, the number choices for the j independent columns is $G(m, j, l, b, \gamma, \delta)$ defined in Lemma 2.9. We can reason on the specific choice of the j independent columns depicted in Fig. 2.16, where the matrices $\mathbf{\Pi}$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are defined. Let $\text{rank}(\mathbf{M_5}) = p$. Since $\mathbf{M_5}$ is a $((b-\delta) \times (a-l))$ matrix, and since $\text{rank}(\mathbf{\Pi}) = t$, it is $p \in \{0, \dots, \min\{b-\delta, a-l, t-\gamma\}\}$.

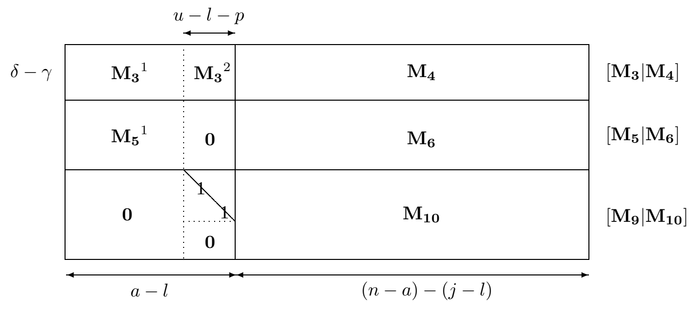


Figure 2.17: Specific choice of $\mathbf{A}$ for the computation of the number of $[\mathbf{M_3}|\mathbf{M_4}]$ matrices.

For each value of p, the number of $\mathbf{A}$ matrices is equal to $\widehat{K}(m-j, n-j, a-l, b-\delta, p, u-l, r-j)$, as proved next. The $((m-j)\times(n-j))$ matrix $\mathbf{A}$ in Fig. 2.16 must have rank $r-j$, because the overall rank r is given by $j+\mathrm{rank}(\mathbf{A})$. It must have no zero columns due to the linear independence between the j independent columns and all the columns. It must have no independent columns because, being the total rank equal to $j+\mathrm{rank}(\mathbf{A})$, such columns would be independent for the overall matrix. The rank of the intersection between its first $a-l$ columns and $b-\delta$ rows is p. Finally, the rank of its first $a-l$ columns (i.e. $\mathrm{rank}(\mathbf{C})$) must be equal to $u-l$. The latter property can be proved as follows. By removing the j independent columns, we obtain a matrix with rank $r-j$, which is also the rank of $\mathbf{A}$. Then, each row of the matrix obtained by removing the j independent columns is a linear combination of the rows of $\mathbf{A}$. In particular, each row of $\mathbf{M_3}$ is a linear combination of the rows of $\mathbf{C}$, from which we obtain $\mathrm{rank}(\mathbf{D}) = \mathrm{rank}(\mathbf{C})$. Since the rank of the first a columns of the overall matrix is u, it follows from Fig. 2.16 that $\mathrm{rank}(\mathbf{D}) = u-l$ i.e. $\mathrm{rank}(\mathbf{C}) = u-l$.

The number of rows of $\mathbf{B}$ is $j-(\delta-\gamma)$, and the only condition on this matrix is that all its rows are linear combinations of the rows of $\mathbf{A}$, whose rank is $r-j$. Then, the number of choices of $\mathbf{B}$ is $2^{(r-j)(j-(\delta-\gamma))}$, that is independent of p, so can be moved outside the summation over this parameter.

The proof is completed by computing the number of $[\mathbf{M_3}|\mathbf{M_4}]$ matrices. It must be $\mathrm{rank}(\mathbf{\Pi}) = t$, $\mathrm{rank}(\mathbf{M_5}) = p$, $\mathrm{rank}(\mathbf{C}) = u-l$ and any row of $[\mathbf{M_3}|\mathbf{M_4}]$ must be a linear combination of the rows of $\mathbf{A}$. Let us consider the specific choice of of $\mathbf{A}$ depicted in Fig. 2.17, where $\mathrm{rank}(\mathbf{M_5^1}) = p$. The condition $\mathrm{rank}(\mathbf{\Pi}) = t$ is satisfied if and only if $\mathrm{rank}(\mathbf{M_3^2}) = t-\gamma-p$. Each row of $\mathbf{M_3^2}$ must be a linear combination of the $u-l-p$ rows of $\mathbf{A}$ corresponding to the $\mathbf{I}_{u-l-p}$ matrix. All the possible $((\delta-\gamma)\times(u-l-p))$ $\mathbf{M_3^2}$ matrices can be generated with these vectors. Then, the number of $\mathbf{M_3^2}$ matrices is

$$\prod_{i=0}^{t-\gamma-p-1} \frac{(2^{\delta-\gamma}-2^i)(2^{u-l-p}-2^i)}{2^{t-\gamma-p}-2^i}\,.$$

Let us fix a $\mathbf{M_3^2}$ matrix. Each row of $\mathbf{M_3^2}$ selects a specific linear combination of the $u-l-p$ rows of $\mathbf{A}$ that correspond to $\mathbf{I}_{u-l-p}$. The other rows have a rank $(r-j)-(u-l-p)$, so there are $2^{[(r-j)-(u-l-p)]}$ possible choices for each row of $[\mathbf{M_3^1}|\mathbf{M_4}]$. Since the total number of such rows is $\delta-\gamma$, the number of $[\mathbf{M_3^1}|\mathbf{M_4}]$ matrices is $2^{[(r-j)-(u-l-p)](\delta-\gamma)}$. $\square$

Chapter 3

Generalized Stability Condition for Generalized and Doubly-Generalized LDPC Codes

3.1 Introduction

As remarked in Chapter 1, a traditional LDPC code of length N and dimension K is graphically represented through a bipartite graph with N variable nodes and $M \geq N - K$ check nodes [20]. A degree-n check node of an LDPC code can be interpreted as a length-n single parity-check (SPC) code, i.e. as a $(n, n-1)$ linear block code, while a degree-n variable node can be interpreted as a length-n repetition code, i.e. as a $(n, 1)$ linear block code.

For standard LDPC ensembles, an important role is played by an inequality known as *stability condition* [7][52]. For transmission on a BEC with erasure probability q, and assuming minimum variable node degree 2, the stability condition establishes the following upper bound to the asymptotic threshold q^* for the LDPC ensemble:

$$q^* \leq [\lambda'(0)\, \rho'(1)]^{-1}. \tag{3.1}$$

In (3.1), $\lambda'(0) = \lambda_2$ is the fraction of edges towards the length-2 repetition variable nodes, while $\rho'(1)$ is the derivative (computed in $x = 1$) of the function $\rho(x) = \sum_{j \geq 2} \rho_j x^{j-1}$, where ρ_j is the fraction of edges connected to SPC check nodes of length j. The bound (3.1) was first developed in [7] from density evolution. It is possible, however, to interpret it in a simple graphical way, by exploiting EXIT charts. More specifically, the stability condition is equivalent to the following statement: a necessary condition for (asymptotic) LDPC successful decoding, is that the derivative of the EXIT function $I_{E,V}(I_A, q)$ for the variable node decoder (VND), with respect to I_A[1] and evaluated in $I_A = 1$, is smaller than

[1] I_A denotes the average *a priori* mutual information in input to the VND or to the CND. Since the extrinsic channel is modelled as a BEC with erasure probability p, it is $I_A = 1 - p$.

the derivative of the inverse EXIT function $I_{E,C}^{-1}(I_A)$ for the check node decoder (CND), evaluated in $I_A = 1$, i.e. (3.1) is equivalent to

$$\left.\frac{\partial I_{E,V}(I_A, q^*)}{\partial I_A}\right|_{I_A=1} \leq \left.\frac{\mathrm{d}I_{E,C}^{-1}(I_A)}{\mathrm{d}I_A}\right|_{I_A=1}. \tag{3.2}$$

The LDPC stability condition (3.1) is a tight upper bound, since there exist LDPC distributions whose threshold achieves it with equality, assuming the closed form $q^* = [\lambda'(0)\rho'(1)]^{-1}$. For achieving (3.1) with equality, it is sufficient that the first occurrence of a tangency point between the EXIT function $I_{E,V}(I_A, q)$ of the VND and the inverse EXIT function $I_{E,C}^{-1}(I_A)$ of the CND appears in $I_A = 1$, i.e.

$$\begin{cases} I_{E,V}(1, q^*) = I_{E,C}^{-1}(1) = 1 \\ \left.\frac{\partial I_{E,V}(I_A, q^*)}{\partial I_A}\right|_{I_A=1} = \left.\frac{\mathrm{d}I_{E,C}^{-1}(I_A)}{\mathrm{d}I_A}\right|_{I_A=1}. \end{cases} \tag{3.3}$$

For LDPC codes, the first equality is always true. As proved in Section 2.2, it is always satisfied also for GLDPC and D-GLDPC codes, provided that all the variable and check component codes have $d_{\min} \geq 2$, which is assumed true throughout this chapter. Then, only the second equality will be considered in the sequel, and referred to as *derivative matching condition.*

In this chapter, the stability condition (3.2), and the derivative matching condition (3.3) are extended to GLDPC and D-GLDPC codes [53]. Two main results are obtained. The first is that only the component codes with $d_{\min} = 2$, including length-2 repetition codes and SPC codes, appear in the stability condition. The second is that, for GLDPC codes satisfying the derivative matching condition, it is always possible to develop a closed-form expression of the threshold; the same expression holds also for D-GLDPC codes satisfying the derivative matching condition, if all the generalized variable nodes have $d_{\min} \geq 3$.

3.2 Definitions and Basic Notation

The transmission channel is a BEC with erasure probability q. Assuming a bipartite graph with random connections, the *extrinsic channel*, over which the messages are exchanged between the variable and check nodes, during the iterative decoding process, is modelled as a second BEC with erasure probability p (see Section 2.1). It is readily proved that $I_A = 1 - p$: since the EXIT functions will be expressed as functions of p (and q for the VND), the derivatives of the EXIT function for the VND and of the inverse EXIT function for the CND will be evaluated in $p = 0$ ($I_A = 1$).

Under the hypothesis of random bipartite graph, the VND and CND EXIT functions are expressed, respectively, by (2.5) and (2.6), here recalled for convenience:

$$I_{E,V}(p,q) = \sum_{i=1}^{\mathcal{I}_V} \lambda_i \, I_{E,V}^{(i)}(p,q) \tag{3.4}$$

$$I_{E,C}(p) = \sum_{i=1}^{\mathcal{I}_C} \rho_i \, I_{E,C}^{(i)}(p), \tag{3.5}$$

where $\mathcal{I}_V$ and $\mathcal{I}_C$ are the number of variable and check node types, $I_{E,V}^{(i)}(p,q)$ and $I_{E,C}^{(i)}(p)$ are the EXIT function for the i-th variable node type and i-th check node type, λ_i and ρ_i are the fraction of edges towards the variable nodes of type i and the check nodes of type i.

For the scope of this chapter, it is useful to isolate, in (3.4), the contribution of the repetition codes and, in (3.5), the contribution of the SPC codes. Since the EXIT function on the BEC for a length-j repetition variable node is $I_E(p,q) = 1 - q\,p^{j-1}$, and since the EXIT function on the BEC for a length-j SPC check node is $I_E(p) = (1-p)^{j-1}$, then we obtain:

$$\begin{aligned} I_{E,V}(p,q) &= \sum_{j \geq 2}^{\text{(rep)}} \lambda_j^{(\text{r})} \cdot (1 - q\,p^{j-1}) + \sum_{i}^{\text{(gen)}} \lambda_i \, I_{E,V}^{(i)}(p,q) \\ &= \sum_{j \geq 2}^{\text{(rep)}} \lambda_j^{(\text{r})} - q\,\lambda_{\text{r}}(p) + \sum_{i}^{\text{(gen)}} \lambda_i \, I_{E,V}^{(i)}(p,q) \end{aligned} \tag{3.6}$$

$$\begin{aligned} I_{E,C}(p) &= \sum_{j \geq 2}^{\text{(SPC)}} \rho_j^{(\text{SPC})} \cdot (1-p)^{j-1} + \sum_{i}^{\text{(gen)}} \rho_i \, I_{E,C}^{(i)}(p) \\ &= \rho_{SPC}(1-p) + \sum_{i}^{\text{(gen)}} \rho_i \, I_{E,C}^{(i)}(p). \end{aligned} \tag{3.7}$$

In (3.6), j is the length of the generic repetition variable node, $\lambda_j^{(\text{r})}$ is the fraction of edges towards the repetition variable nodes of length j, $\lambda_{\text{r}}(x)$ is defined as $\sum_{j \geq 2} \lambda_j^{(\text{r})} x^{j-1}$. The summation in i is over all the generalized variable node types. Analogously, in (3.7), j is the length of the generic SPC node, $\rho_j^{(\text{SPC})}$ is the fraction of edges towards the SPC nodes of length j, $\rho_{\text{SPC}}(x)$ is defined as $\sum_{j \geq 2} \rho_j^{(\text{SPC})} x^{j-1}$.

The EXIT function for an (n,k) generalized variable node of a D-GLDPC code on the BEC can be expressed by (2.2), here recalled for convenience:

$$I_E(p,q) = 1 - \frac{1}{n} \sum_{t=0}^{n-1} \sum_{z=0}^{k} a_{t,z}\, p^t\,(1-p)^{n-t-1}\, q^z\,(1-q)^{k-z}, \tag{3.8}$$

where $a_{t,z} = [(n-t)\widetilde{e}_{n-t,k-z} - (t+1)\widetilde{e}_{n-t-1,k-z}]$, and where the parameter $\tilde{e}_{g,h}$ (with $g = 0, \dots, n$ and $h = 0, \dots, k$) is the (g,h)-th un-normalized split information function. Considering a representation of the generator matrix $\mathbf{G}$ for the (n,k) variable node, and appending to it the $(k \times k)$ identity matrix $\mathbf{I}_k$, $\tilde{e}_{g,h}$ is equal to the summation of the ranks over all the possible submatrices obtained selecting g columns in $\mathbf{G}$ and h columns in $\mathbf{I}_k$. The split information functions for a generalized variable node, and then its EXIT function, heavily depend on the code representation, i.e. on the chosen generator matrix [41]. Then, the performance of the overall D-GLDPC code depends on the code representation used at the generalized variable nodes.

The EXIT function for a generalized (n,k) check node of a GLDPC or D-GLDPC code on the BEC can be obtained by letting $q \to 1$ in (3.8) (no communication channel is present). The obtained expression is given in (2.3), here recalled for convenience:

$$I_E(p) = 1 - \frac{1}{n} \sum_{t=0}^{n-1} a_t p^t (1-p)^{n-t-1}, \tag{3.9}$$

where $a_t = (n-t)\,\tilde{e}_{n-t} - (t+1)\tilde{e}_{n-t-1}$ and where for $g = 0, \dots, n$, $\tilde{e}_g$ is the g-th un-normalized information function of the (n,k) code. It is defined as the summation of the ranks over all the possible submatrices obtained selecting g columns from the generator matrix $\mathbf{G}$. As opposed to the split information functions, the information functions are independent of the code representation. Thus, different code representations lead to the same EXIT function for the generalized check node. The performance of the overall D-GLDPC code is then independent of the specific representation of the generalized check nodes.

Equations (3.8) and (3.9) assume that MAP erasure correction is performed at the variable and check node.

3.3 Independent Sets and Minimum Distance

The development of a generalized stability condition for GLDPC and D-GLDPC codes is mostly based on the concept of *independent set* of columns introduce in Section 2.2.1 and on Lemma 2.2 proved on page 28. This lemma establishes a necessary and sufficient condition for a $(k \times t)$ binary matrix, obtained by selecting t columns in the generator matrix $\mathbf{G}$ (any representation) of a (n,k) linear block code, to be full rank. The concept of independent set and Lemma 2.2 are recalled next.

Definition 3.1 (independent set). *Given a $(k \times n)$ binary matrix of rank r, a set of t columns is said an* independent set *when removing these t columns from the matrix leads to a $(k \times n - t)$ matrix with rank $r - \Delta r < r$, for some $0 < \Delta r \leq t$. The number t is the size of the independent set.*

Lemma 3.1. *Let* $\mathbf{G}$ *be any representation of the generator matrix of an* (n, k) *linear block code. Then, the following statements are equivalent:*

a) the code has minimum distance t*;*

b) the minimum size of the independent sets of $\mathbf{G}$ *is* t*.*

The present section is concluded by the following lemma.

Lemma 3.2. *Let* $k < n$*, and* t *be the minimum size of the independent sets of a* $(k \times n)$ *binary matrix with rank* r*. Then, removing any independent set of size* t*, leads to a* $(k \times (n - t))$ *matrix with rank* $r - 1$*.*

Proof. Since t is the minimum size of the independent sets of the matrix, then removing any set of $j < t$ columns does not affect the rank. For an independent set of size t, one can remove any subset of $t - 1$ columns without reducing the rank; when removing the t-th column, the rank can only decrease by 1. □

3.4 Stability Condition and Derivative Matching for GLDPC Codes

In GLDPC codes, all the variable nodes are repetition codes. Recalling (3.6), and observing that $\sum_{j\geq 2}^{(\mathrm{rep})} \lambda_j^{(\mathrm{r})} = 1$, the EXIT function on the BEC for the VND is given by $I_{E,V}(p, q) = 1 - q\lambda^{(\mathrm{r})}(x)$. Hence:

$$\frac{\partial I_{E,V}(p, q)}{\partial p}\Big|_{p=0} = -q\,\lambda_2^{(\mathrm{r})}. \tag{3.10}$$

Recalling (3.7), the derivative of $I_{E,C}(p)$ in $p = 0$ is

$$\frac{\mathrm{d}I_{E,C}(p)}{\mathrm{d}p}\Big|_{p=0} = -\rho'^{(\mathrm{SPC})}(1) + \sum_{i}^{(\mathrm{gen})} \rho_i \frac{\mathrm{d}I_{E,C}^{(i)}(p)}{\mathrm{d}p}\Big|_{p=0}. \tag{3.11}$$

In order to develop the previous expression, it is necessary to express the derivative of the EXIT function for the generalized check nodes. This task can be performed by exploiting Lemma 3.1, as explained next.

Consider an (n, k) generalized check node, with EXIT function $I_E(p)$ in the form (3.9). It is readily shown that the derivative of $I_E(p)$, computed in $p = 0$, is

$$\frac{\mathrm{d}I_E(p)}{\mathrm{d}p}\,|_{p=0} = \frac{(n-1)a_0 - a_1}{n}.$$

According to Lemma 3.1, $a_0 = 0$ if and only if the generalized check node has minimum distance $d_{\min} \geq 2$. In fact, the generator matrix of the check node is full rank (rank $= k$)

by definition, so $\tilde{e}_n = k$. Furthermore, from Lemma 3.1, removing any single column from the generator matrix does not reduce the rank if and only if $d_{\min} \geq 2$, thus leading to $\tilde{e}_{n-1} = n\,k$. Then, $a_0 = n\,\tilde{e}_n - \tilde{e}_{n-1} = n\,k - n\,k = 0$. As recalled in Section 1.1, the hypothesis $d_{\min} \geq 2$ is always assumed in this work. Then, it will be always assumed $a_0 = 0$.

It follows that, assuming $d_{\min} \geq 2$ for the check node,

$$\frac{\mathrm{d}I_E(p,q)}{\mathrm{d}p}\,|_{p=0} = -\frac{a_1}{n},$$

where $a_1 = (n-1)\tilde{e}_{n-1} - 2\,\tilde{e}_{n-2} = k\,n\,(n-1) - 2\,\tilde{e}_{n-2}$. By applying again Lemma 3.1, we obtain

$$a_1 \left\{ \begin{array}{ll} = 0 & \text{if } d_{\min} \geq 3 \\ \neq 0 & \text{if } d_{\min} = 2\,. \end{array} \right. \tag{3.12}$$

In fact, if $d_{\min} \geq 3$, removing any pair of columns from the generator matrix does not affect the rank. In this case $2\,\tilde{e}_{n-2} = 2\,k\,\binom{n}{2} = k\,n\,(n-1)$, hence $a_1 = 0$.

According to these results, the only generalized check nodes that give some contribution to the summation in the second member of (3.11) are those characterized by $d_{\min} = 2$. By recalling that all the SPC codes have minimum distance 2, we conclude that $\frac{\mathrm{d}I_{E,C}(p)}{\mathrm{d}p}\,|_{p=0}$ only depends on the check nodes with $d_{\min} = 2$. The derivative in $p = 0$ of the EXIT function for the CND can be then expressed as

$$\begin{aligned} &\frac{\mathrm{d}I_{E,C}(p)}{\mathrm{d}p}\Big|_{p=0} \\ &= -\rho'^{(\mathrm{SPC})}(1) - \sum_{i}^{d_{\min}=2} \rho_i\,\frac{k_i n_i(n_i-1) - 2\,\tilde{e}_{n_i-2}}{n_i} \\ &= -\rho'^{(\mathrm{SPC})}(1) - \sum_{i}^{d_{\min}=2} \frac{2\rho_i}{n_i}\,\Delta_{n-2}^{(i)}, \end{aligned} \tag{3.13}$$

where the summation is over the generalized check nodes with minimum distance 2. In the last equality, $\Delta_{n-2}^{(i)} > 0$ is defined as follows. Let $\mathcal{S}_{n_i-2}$ be the generic $(k_i \times (n_i-2))$ matrix obtained by selecting $n_i - 2$ columns in the generator matrix. Then,

$$\Delta_{n-2}^{(i)} = \sum_{\mathcal{S}_{n_i-2}} (k_i - \mathrm{rank}(\mathcal{S}_{n_i-2})),$$

where the summation is over all the possible matrices $\mathcal{S}_{n_i-2}$. The parameter $\Delta_{n-2}^{(i)}$ does not depend on the chosen representation for the i-th generalized check node.

The derivative in $p = 0$ for the inverse EXIT function $I_{E,C}^{-1}(p)$ of the CND is simply given by the inverse of (3.13). Then, from (3.10) and (3.13), the stability condition

$\partial I_{E,V}(I_A, q^*)/\partial I_A \,|_{I_A=1} \leq \mathrm{d}I_{E,C}^{-1}/\mathrm{d}I_A \,|_{I_A=1}$ for GLDPC codes leads to [2]

$$q^* \leq \left[\lambda_2^{(\mathrm{r})} \left(\rho'^{(\mathrm{SPC})}(1) + \sum_i^{d_{\min}=2} \frac{2\rho_i}{n_i} \Delta_{n-2}^{(i)}\right)\right]^{-1}, \tag{3.14}$$

an upper bound on the threshold q^* which represents a necessary condition for successful GLDPC (asymptotic) decoding.

For GLDPC codes satisfying the derivative matching condition (3.3) (the first occurrence of a tangency point between $I_{E,V}(p,q)$ and $I_{E,C}^{-1}(p)$ appears in $p=0$), the threshold assumes the following simple closed form:

$$q^* = \left[\lambda_2^{(\mathrm{r})} \left(\rho'^{(\mathrm{SPC})}(1) + \sum_i^{d_{\min}=2} \frac{2\rho_i}{n_i} \Delta_{n-2}^{(i)}\right)\right]^{-1}. \tag{3.15}$$

If only generalized check nodes with $d_{\min} \geq 3$ are used, then the stability condition (3.14) and the threshold expression (3.15) become, respectively,

$$q^* \leq \left[\lambda_2^{(\mathrm{r})}\, \rho'^{(\mathrm{SPC})}(1)\right]^{-1} \tag{3.16}$$

$$q^* = \left[\lambda_2^{(\mathrm{r})}\, \rho'^{(\mathrm{SPC})}(1)\right]^{-1}. \tag{3.17}$$

3.5 Stability Condition and Derivative Matching for D-GLDPC Codes

The derivative in $p=0$ of the EXIT function for the CND of D-GLDPC codes is the same as for GLDPC codes, and is expressed by (3.13). The derivative with respect to p, in $p=0$, of the EXIT function for the VND is developed next.

By computing the partial derivative respect to p of $I_{E,V}(p,q)$ from (3.6), we obtain:

$$\frac{\partial I_{E,V}(p,q)}{\partial p}\Big|_{p=0} = -q\,\lambda_2^{(\mathrm{r})} + \sum_i^{(\mathrm{gen})} \lambda_i \frac{\partial I_{E,V}^{(i)}(p,q)}{\partial p}\Big|_{p=0}. \tag{3.18}$$

In order to develop the summation on the generalized variable node types in the second member, the derivative in $p=0$ for the EXIT function of each generalized variable node must be computed, based on (3.8). By defining

$$f(p) = \sum_{t=0}^{n-1} a_{t,z}\, p^t\, (1-p)^{n-1-t},$$

[2] If the derivatives are computed with respect to $p = 1 - I_A$, the stability condition is $\partial I_{E,V}(p,q^*)/\partial p \,|_{p=0} \geq \mathrm{d}I_{E,C}^{-1}/\mathrm{d}p \,|_{p=0}$.

it results

$$\frac{\partial I_E(p,q)}{\partial p}\Big|_{p=0} = -\frac{1}{n}\sum_{z=0}^{k} q^z\,(1-q)^{k-z}\frac{\mathrm{d}f(p)}{\mathrm{d}p}\Big|_{p=0}$$
$$= \sum_{z=0}^{k} q^z\,(1-q)^{k-z}\,\frac{(n-1)a_{0,z}-a_{1,z}}{n}, \tag{3.19}$$

where the fact that

$$\frac{\mathrm{d}f(p)}{\mathrm{d}p}\Big|_{p=0} = -(n-1)a_{0,z}+a_{1,z}$$

has been exploited. The previous relationship can be further developed by exploiting Lemma 3.1. Since, by hypothesis, any variable node has minimum distance at least 2, removing any single column from the generator matrix $\mathbf{G}$ of the code associated to the variable node does not affect the rank of $\mathbf{G}$. It follows $a_{0,z} = n\,\widetilde{e}_{n,k-z} - \widetilde{e}_{n-1,k-z} = k\,n\binom{k}{k-z} - k\,n\binom{k}{k-z} = 0$, thus leading to

$$\frac{\partial I_E(p,q)}{\partial p}|_{p=0} = -\sum_{z=0}^{k} q^z\,(1-q)^{k-z}\,\frac{a_{1,z}}{n}.$$

Lemma 3.1 can be invoked again in order to show that

$$a_{1,z}\begin{cases} =0 & \forall\, z \quad \text{if } d_{\min}\geq 3\\ \neq 0 & \forall\, z \quad \text{if } d_{\min}=2,\end{cases} \tag{3.20}$$

where $d_{\min}$ is the minimum distance of the (n,k) code associated to the variable node under analysis. In fact, under the hypothesis $d_{\min}\geq 3$, no independent sets of size 1 and 2 are present in (any representation of) the generator matrix $\mathbf{G}$. Then

$$a_{1,z} = (n-1)\,\tilde{e}_{n-1,k-z} - 2\,\tilde{e}_{n-2,k-z} = k\,n\,(n-1)\binom{k}{k-z} - 2\,k\binom{n}{n-2}\binom{k}{k-z} = 0.$$

It follows that the contribution of the generalized variable nodes to $\frac{\partial I_{E,V}(p,q)}{\partial p}|_{p=0}$ is 0 if they all have minimum distance greater than or equal to 3. The only non-null contribution comes from the generalized variable nodes with minimum distance $d_{\min}=2$, which is coherent with the fact that, among the repetition codes, only those with $d_{\min}=2$ (i.e. the length-2 repetition codes) give a non-null contribution.

Then, relationship (3.18) can be developed as

$$\frac{\partial I_{E,V}(p,q)}{\partial p}\Big|_{p=0} = -q\,\lambda_2^{(\mathrm{r})} - \sum_{i}^{d_{\min}=2}\sum_{z=0}^{k_i} q^z\,(1-q)^{k_i-z}$$
$$\cdot\,\lambda_i\frac{k_i\,n_i(n_i-1)\binom{k_i}{k_i-z} - 2\,\tilde{e}_{n_i-2,k_i-z}}{n_i}$$
$$= -q\,\lambda_2^{(\mathrm{r})} - \sum_{i}^{d_{\min}=2}\sum_{z=0}^{k_i} q^z\,(1-q)^{k_i-z}\frac{2\lambda_i}{n_i}\Delta^{(i)}_{n-2,k-z}. \tag{3.21}$$

In the previous expression, $\Delta^{(i)}_{n-2,k-z}$ is defined in a similar way as $\Delta^{(i)}_{n-2}$ for GLDPC codes. Let $\mathcal{S}_{n_i-2,k_i-z}$ be the generic $(k_i \times (n_i - 2 + k_i - z))$ matrix obtained by selecting $n_i - 2$ columns in the generator matrix, and $k_i - z$ columns in the $(k \times k)$ identity matrix. Then,

$$\Delta^{(i)}_{n-2,k-z} = \sum_{\mathcal{S}_{n_i-2,k_i-z}} (k_i - \mathrm{rank}(\mathcal{S}_{n_i-2,k_i-z})),$$

where the summation is over all the possible matrices $\mathcal{S}_{n_i-2,k_i-z}$. Differently from $\Delta^{(i)}_{n-2}$, the parameter $\Delta^{(i)}_{n-2,k-z}$ depends on the code representation, i.e. on the chosen generator matrix for the variable node.

By combining (3.13) and (3.21) the stability condition $\left.\frac{\partial I_{E,V}(p,q^*)}{\partial p}\right|_{p=0} \geq \mathrm{d}I^{-1}_{E,C}(p)/\mathrm{d}p\,|_{p=0}$ becomes:

$$q^*\,\lambda_2^{(\mathrm{r})} + \sum_{i}^{d_{\min}=2} \sum_{z=0}^{k_i} (q^*)^z\,(1-q^*)^{k_i-z} \frac{2\lambda_i\,\Delta^{(i)}_{n-2,k-z}}{n_i}$$
$$\leq \left[\rho'^{(\mathrm{SPC})}(1) + \sum_{i}^{d_{\min}=2} \frac{2\rho_i}{n_i}\,\Delta^{(i)}_{n-2}\right]^{-1}. \qquad (3.22)$$

Inequality (3.22) is a necessary condition for (asymptotic) successful D-GLDPC decoding on the BEC. Differently from the GLDPC case, it is not possible to express this inequality as an explicit upper bound on q^*, because of the impossibility to factor q^* from the summation in z. For D-GLDPC codes satisfying the derivative matching condition, (3.22) holds with equality. For the same reason, it does not lead to an explicit closed form expression of the threshold. However, if $d_{\min} \geq 3$ for all the generalized variable nodes, then any term of the summation over the generalized variable nodes is null. In this case, the same upper bound to q^* as in (3.15) holds:

$$q^* \leq \left[\lambda_2^{(\mathrm{r})}\left(\rho'^{(\mathrm{SPC})}(1) + \sum_{i}^{d_{\min}=2} \frac{2\rho_i}{n_i}\,\Delta^{(i)}_{n-2}\right)\right]^{-1},$$

and this inequality is achieved with equality by D-GLDPC codes satisfying the derivative matching condition. Moreover, if $d_{\min} \geq 3$ also for all the generalized check nodes, then the upper bound on q^* assumes the simple form as in (3.16):

$$q^* \leq \left[\lambda_2^{(\mathrm{r})}\,\rho'^{(\mathrm{SPC})}(1)\right]^{-1}.$$

This inequality is achieved with equality by D-GLDPC codes satisfying the derivative matching condition.

3.6 Final Remarks

In this chapter, a stability condition on the BEC has been derived for GLPDC and D-GLDPC codes, generalizing the inequality $q^* \leq [\lambda'(0)\rho'(1)]^{-1}$, valid for standard LDPC codes. A derivative matching condition has been also defined, sufficient for a GLDPC and D-GLDPC codes to achieve the stability condition with equality.

It has been shown that, as for LDPC codes, only the variable and check nodes with minimum distance 2 are involved in the stability condition. Then, it has been shown that the stability condition for GLDPC codes can be always explicitly expressed as an upper bound on the decoding threshold. D-GLDPC codes do not share this property in general; however, if all the generalized variable nodes have minimum distance at least 3, the stability condition becomes the same as for GLDPC codes. As a consequence, for GLDPC codes and for D-GLDPC codes with variable component codes of minimum distance at least 3, the decoding threshold assumes a simple closed form.

3.7 Appendix: Some Hints for Further Investigation

It was proved in [15] (also exploiting results from [54]) that the parameter $\lambda'(0)\rho'(1)$ plays a fundamental role with respect to the minimum distance properties of standard irregular LDPC codes. More specifically, it was shown that the minimum distance of an irregular, length-n, LDPC code $\mathcal{G}$ randomly chosen in the standard ensemble $\mathcal{C}(\lambda, \rho, n)$ (i.e. in the ensemble of all LDPC codes with length n and degree distribution (λ, ρ)) satisfies $\lim_{n\to\infty} \Pr(d_{\min}(\mathcal{G}) > \lfloor \delta n \rfloor) = \sqrt{1 - \lambda'(0)\rho'(1)}$ for strictly positive δ, if $\lambda'(0)\rho'(1) < 1$. It was also shown in [15] that this result can be even improved to $\lim_{n\to\infty} \Pr(d_{\min}(\mathcal{G}) > \lfloor \delta n \rfloor) = 1$ by properly expurgating the standard LDPC ensemble. On the contrary, if $\lambda'(0)\rho'(1) > 1$, for sufficiently small $l \in \mathbb{N}$, the number $N(\mathcal{G}, l)$ of codewords of Hamming weight l for a randomly chosen code $\mathcal{G}$ in $\mathcal{C}(\lambda, \rho, n)$ satisfies $\Pr(N(\mathcal{G}, l) > 0) = 1 - e^{-\frac{(\lambda'(0)\rho'(1))^l}{2l}} + O(1/\sqrt[3]{n})$.

The parameter $\lambda'(0)\rho'(1)$ is the inverse of the second member in the stability condition, which has been generalized in this chapter to GLDPC and D-GLDPC codes. Thus, a natural question is whether the mentioned result for LDPC code ensembles can be extended to GLDPC and D-GLDPC codes. In other words, writing the stability condition as $q^* \leq q^*_{\max}$, with $q^*_{\max}$ depending on the edge distribution and on the structure of the variable and check component codes, the question is whether there exists some threshold value for $[q^*_{\max}]^{-1}$ which discriminates between a linear and a sublinear minimum distance behavior of sufficiently long GLDPC or D-GLDPC codes randomly chosen in their ensemble. In the most general case, the parameter $[q^*_{\max}]^{-1}$ does not admit an explicit close form, which indeed exists for GLDPC codes and for D-GLDPC codes without generalized component codes having minimum distance 2.

Such investigation would help to better understand, from a theoretical point of view, the potentials of these generalized classes of iteratively decoded codes, as well as their possible advantages with respect to standard LDPC codes.

Chapter 4

A Class of LDPC Erasure Distributions with Closed-Form Threshold Expression

4.1 Introduction

The unavailability of a closed form expression for the decoding threshold of low-density parity-check (LDPC) codes still represents an open problem. So far, no general closed form threshold expression is known for any transmission channel. Some results in this sense have been developed for the BEC, for which some analytic threshold expressions have been proposed. In [55], an analytic threshold expression has been presented for regular LDPC ensembles. A more general analytic expression, valid for check regular ensembles, has been proposed in [56]. The most general analytic expression currently available is that one developed in [57], which can be applied to fully irregular ensembles. For regular or check regular ensembles, the formula proposed in [57] coincides with that one from [56]. The problem with these expressions is that none of them can be really considered a *closed form* one (except for the case of regular LDPC ensembles with degree-2 variable nodes presented in [55]). In fact, in all these formulas, the threshold is a function of some parameter which in general does not admit a closed form expression. This parameter, which depends on the degree distribution, can be a root of a real polynomial, a fixed point of a real function, or the abscissa of the tangent point between two EXIT curves [58].

An exception is represented by the degree distributions for which the first occurrence of a tangency point between the EXIT curves in the EXIT chart appears for a value of the *a priori* mutual information equal to 1 (which is known as *derivative matching condition*, as explained in Section 3.1). Denoting the decoding threshold by q^*, these distributions achieve with equality the stability condition $q^* \leq [\lambda'(0)\rho'(1)]^{-1}$ [7] (which holds for any distribution). This chapter presents a family of check regular LDPC distributions, called

p-positive distributions [59], which are a subset of the distributions achieving the derivative matching condition. The starting point for obtaining this family is the afore mentioned analytic formula developed in [56].

The name "p-positive" is chosen because these distributions are defined as those ones for which a certain polynomial $p(\cdot)$, whose coefficients depend on the edge-oriented variable distribution $\lambda(x)$ and on the degree d_c of the check nodes, is non-negative between 0 and 1. The theory of polynomials can be applied to obtain conditions for a check regular distribution to be p-positive. A useful theorem in this sense is the Fourier-Budan theorem [60, p. 27], which states an upper bound to the number of real roots in a given interval, for a polynomial with real coefficients. The application of the Fourier-Budan theorem to the polynomial $p(\cdot)$ leads to a set of inequalities involving the coefficients λ_i's and d_c, which represent a necessary condition for a distribution to be p-positive.

Besides the closed form threshold expression, the p-positive distributions are shown to exhibit some additional good properties. First, the threshold of the optimal p-positive distribution under a set of constraints, represented by the given code rate, check degree and maximum variable degree, is in some cases extremely close to that of the best known check regular degree distributions. Second, within the p-positive class, it is in some cases possible to optimize the degree distribution with analytic methods only, i.e. without using numerical optimization tools. In fact, despite being only necessary, the condition obtained from the Fourier-Budan theorem is shown to be sufficient, in some cases, to find the p-positive distribution with optimum threshold under the given set of constraints. Third, it is proved that capacity achieving sequences exist within the p-positive class. More specifically, it is recognized that any binomial degree distribution [27] is p-positive.

4.2 Threshold for Check Regular Distributions

Let $\mathcal{C}^n(\lambda, \rho)$ be the ensemble of all the length-n LDPC codes with edge-oriented degree distribution (λ, ρ) [7]. Let q be the BEC erasure probability. Then, the (bit oriented) threshold q^* for the ensemble $\mathcal{C}^n(\lambda, \rho)$ on the BEC is defined as the maximum q for which the residual bit erasure probability can be made arbitrarily small by increasing the number of decoding iterations, in the limit where the codeword length n tends to infinity.

Density evolution for the BEC [7], can be expressed as follows. If x_ℓ is the probability that a message from a variable node to a check node during the ℓ-th decoding iteration is an erasure message and the bipartite graph is cycle-free (infinite codeword length), then $x_{\ell+1} = q\,\lambda(1-\rho(1-x_\ell))$. Hence, the threshold q^* is the maximal q for which $x_\ell \to 0$ in the limit where $\ell \to \infty$. The first appearance of a non-zero fixed point for x_ℓ is a necessary and sufficient condition for the corresponding q to be the threshold.

From this observation, q^* can be equivalently expressed as the maximal q for which

$q\,\lambda(1-\rho(1-x)) < x \;\forall\, x \in (0,q]$ [61], or the maximal q for which:

$$\rho(1-q\,\lambda(x)) > 1-x \qquad \forall\, x \in (0,1]\,. \tag{4.1}$$

For check regular codes, with variable distribution $\lambda(x) = \sum_{i\geq 2} \lambda_i x^{i-1}$ and check distribution $\rho(x) = x^{d_c-1}$ ($d_c \geq 3$), the inequality (4.1) becomes:

$$(1-q\lambda(x))^{d_c-1} > 1-x \qquad \forall\, x \in (0,1]. \tag{4.2}$$

Let us define $f(x,q) = (1-q\lambda(x))^{d_c-1}$ and suppose at first $\lambda_2 = 0$. For any given $x \in (0,1]$, $f(x,q)$ is continuous and monotonically decreasing with respect to q, varying from $f(x,0)=1$ to $f(x,1) = (1-\lambda(x))^{d_c-1} > 0$. Moreover, it is everywhere derivable for x in $(0,1)$, and the following relationships hold:

$$\begin{aligned}
\partial f/\partial x\,(0,q) &= 0 \quad \forall\, q \in (0,1)\\
f(0,q) &= 1 \quad \forall\, q \in (0,1)\\
f(1,q) &= (1-q)^{d_c-1} > 0\\
f(x,0) &= 1 \quad \forall\, x \in (0,1).
\end{aligned}$$

It follows from these properties that the maximal value of q for which (4.2) holds is the minimum value of q for which the graph of $f(x,q)$ (considered as a function of x with q playing the role of parameter) is tangent to the graph of $g(x) = 1-x$, for some $x=\gamma$. The tangency condition in $x=\gamma$ is:

$$\begin{cases} (1-q\lambda(\gamma))^{d_c-1} = 1-\gamma \\ q(d_c-1)\lambda'(\gamma)(1-q\lambda(\gamma))^{d_c-2} = 1. \end{cases} \tag{4.3}$$

The relationships (4.3) are not able to unambiguously determine the threshold. In fact, (4.3) can admit several discrete solutions (q,γ), with $\gamma \in (0,1)$ and $q \in (0,1)$. From a geometrical point of view, several discrete values of $q \in (0,1)$ can exist for which the graph of $f(x,q)$ (for fixed q) is tangent to the graph of $g(x) = 1-x$ in some $x=\gamma$. The threshold q^* for the check regular distribution is the minimum among these discrete values of q.

The second equation of (4.3) can be written as

$$(1-q\lambda(\gamma))^{d_c-2} = [q(d_c-1)\lambda'(\gamma)]^{-1}\,,$$

that can be substituted in the first equation in order to obtain

$$q = [\lambda(\gamma) + (d_c-1)(1-\gamma)\lambda'(\gamma)]^{-1}\,. \tag{4.4}$$

In the following, $h(\cdot)$ will denote the following key function:

$$h(x) = [\lambda(x) + (d_c-1)(1-x)\lambda'(x)]^{-1}. \tag{4.5}$$

By substituting (4.4) into the first equation of (4.3), we obtain

$$\frac{\lambda(\gamma)+(d_c-1)(1-\gamma)\lambda'(\gamma)}{(d_c-1)(1-\gamma)\lambda'(\gamma)}=(1-\gamma)^{-\frac{1}{d_c-1}},$$

and developing this expression leads to:

$$\gamma = 1-\Big(\frac{\lambda(\gamma)}{(d_c-1)\lambda'(\gamma)}+1-\gamma\Big)^{\frac{d_c-1}{d_c-2}}. \tag{4.6}$$

Summarizing, for a given check regular ensemble with $\lambda_2=0$, the threshold on the BEC is equal to the minimum $h(\gamma)$ under the constraint that γ is one of the (usually) several fixed points in $(0,1)$ of

$$\phi(x) = 1-\Big(\frac{\lambda(x)}{(d_c-1)\lambda'(x)}+1-x\Big)^{\frac{d_c-1}{d_c-2}}. \tag{4.7}$$

Note that $\gamma=0$ is always a fixed point for $\phi(\cdot)$. However, for $\lambda_2=0$ the threshold is never achieved in correspondence of $\gamma=0$, because in this case $\lim_{x\to 0+} h(x)=+\infty$.

The only differences when removing the hypothesis $\lambda_2=0$ are that $\partial f/\partial x(0,q)<0\ \forall\, q\in(0,1)$, and that $h(0)$ is finite and equal to $[\lambda_2(d_c-1)]^{-1}=[\lambda'(0)\rho'(1)]^{-1}$. Hence, the threshold could be achieved in correspondence of the fixed point $\gamma=0$. Thus, the threshold for a check regular ensemble with $\lambda_2>0$ is equal to the minimum $h(\gamma)$, with γ fixed point in $[0,1)$ of $\phi(\cdot)$. An analysis of the fixed points of $\phi(\cdot)$ is presented in appendix to this chapter.

In the special case of regular LDPC codes, with variable distribution $\lambda(x)=x^{d_v-1}$, $d_v\geq 3$, and check distribution $\rho(x)=x^{d_c-1}$, $d_c\geq 3$, the threshold is given by

$$q^* = \frac{[(d_c-1)(d_v-1)]^{-1}}{\gamma^{d_v-2}-c\,\gamma^{d_v-1}}, \tag{4.8}$$

where γ is the unique fixed point in $(0,1)$ of

$$\psi(x)=1-(1-c\,x)^{\frac{d_c-1}{d_c-2}}. \tag{4.9}$$

with $c=[(d_c-1)(d_v-1)-1]/[(d_c-1)(d_v-1)]<1$.

4.3 The New Class of Degree Distributions

In this section, the class of p-positive degree distributions is introduced.

Definition 4.1. *A degree distribution with code rate R and threshold $q^*=(1-\epsilon)(1-R)$ is called* $(1-\epsilon)$ capacity achieving of rate R.

Consider a check regular distribution for which $h(x) \geq h(0)\ \forall x \in (0,1]$, where $h(\cdot)$ is defined in (4.5). This can be equivalently written as $[h(x)]^{-1} \leq [h(0)]^{-1}$, i.e.

$$\sum_{i=2}^{L} \lambda_i x^{i-1} + (d_c - 1)(1-x)\sum_{i=2}^{L}(i-1)\lambda_i x^{i-2} \leq \lambda_2(d_c - 1),$$

where L denotes the maximum variable degree. After some algebraic manipulation, the previous inequality can be put in the form $p(x) \geq 0$, where $p\,(\cdot)$ is the real polynomial

$$p\,(x) = \omega_L\,\lambda_L\,x^{L-2} + \sum_{i=2}^{L-1}[\omega_i\lambda_i - (\omega_{i+1}+1)\,\lambda_{i+1}]\,x^{i-2}, \tag{4.10}$$

where $\omega_i = (d_c - 1)(i-1) - 1$. The polynomial $p\,(\cdot)$ will be expressed in the form $p\,(x) = \sum_{i=0}^{L-2} p_i\,x^i$. The condition $h(x) \geq h(0)$ for all $x \in (0,1]$ is equivalent to the condition that the real polynomial $p\,(\cdot)$ is positive or null in $(0,1]$. The class of degree distribution pairs studied in this paper is introduced next.

Definition 4.2. *A* p-positive distribution *is any check regular degree distribution with* $\lambda_2 > 0$*, such that* $p(x) \geq 0\ \forall\ x \in (0,1]$.

The following theorem individuates a simple closed form for the threshold of the p-positive distributions, and states the necessary and sufficient condition for any p-positive distribution to be $(1-\epsilon)$ capacity achieving of rate R.

Theorem 4.1. *The threshold of any p-positive degree distribution is equal to* $[\lambda'(0)\rho'(1)]^{-1}$*. The degree distribution is* $(1-\epsilon)$ *capacity achieving of rate R if and only if*

$$\lambda_2 = [(1-\epsilon)(1-R)(d_c-1)]^{-1}. \tag{4.11}$$

Proof. For a p-positive distribution, $h(x) \geq h(0)$ for all $x \in (0,1]$. From Section 4.2 it is known that the threshold for a check regular ensemble with $\lambda_2 > 0$ is the minimum among discrete values $h(\gamma)$, with γ fixed point in $[0,1)$ of $\phi(\cdot)$ defined in (4.7). Moreover, $\gamma = 0$ is always a fixed point of $\phi(\cdot)$ under the condition $\lambda_2 > 0$. If $h(x) \geq h(0)$ for all $x \in (0,1]$, then the minimum is always achieved in correspondence of the fixed point $\gamma = 0$, independently of the number and the positions of all the other fixed points of $\phi(\cdot)$. Hence, $q^* = h(0) = [\lambda'(0)\rho'(1)]^{-1} = [\lambda_2(d_c-1)]^{-1}$. If and only if equality (4.11) holds, it is $q^* = (1-\epsilon)(1-R)$. □

For a given $\epsilon > 0$, the search for p-positive $(1-\epsilon)$ capacity achieving of rate R distributions, with check degree d_c and maximum variable degree L, can be performed by letting λ_2 be expressed by (4.11), and looking for λ_i, $i = 3, \dots, L$, such that $p(x) \geq 0$ for all $x \in (0, 1]$. In general, for some R, d_c and L, this problem admits solutions for $\epsilon \geq \epsilon_{\mathrm{opt}} > 0$, with ϵ_{opt} depending on R, d_c and L. The value ϵ_{opt} is associated to the optimal p-positive distribution, under the imposed set of constraints (i.e. code rate, check nodes degree and active variable degrees).

It is readily shown that $p(1) = (d_c - 1)\lambda_2 - 1$, which is positive (from (4.11)). Then, the condition that $p(x) \geq 0$ for x between 0 and 1, is equivalent to the condition that $p(\cdot)$ has only roots with even multiplicity between 0 and 1. Several well known properties of the real roots of polynomials can be then exploited, in order to develop conditions for a check regular distribution to be p-positive. In particular the following theorem, known as the Fourier-Budan theorem, permits to obtain a simple necessary (but not sufficient) condition. It provides an upper bound to the number of zeroes of a real polynomial between two values a and b, with $a < b$.

Theorem 4.2 (Fourier-Budan Theorem [60, p. 27]). *Let $p(\cdot)$ be a real polynomial of degree d, and let a and b be two real values such that $a < b$, and $p(a) \cdot p(b) \neq 0$. Then, the number of real roots of $p(\cdot)$ between a and b (each one counted with its multiplicity) is* not greater *than $A - B$, where A and B are, respectively, the number of sign changes in sequences:*

$$p(a),\, p'(a),\, p''(a),\, \dots,\, p^{(d)}(a)$$
$$p(b),\, p'(b),\, p''(b),\, \dots,\, p^{(d)}(b).$$

Moreover, if the number of real roots of $p(\cdot)$ between a and b is smaller than $A - B$, then it differs from $A - B$ by an even number.

This theorem can be directly applied to the polynomial $p(\cdot)$ defined by (4.10), where $a = 0$ and $b = 1$, leading to the following corollary.

Corollary 4.1. *The number of real roots between 0 and 1 of $p(x) = \sum_{i=0}^{L-2} p_i\, x^i$ defined by* (4.10) *is not greater than the number A of sign changes in the sequence $p_0, p_1, \dots, p_{L-2}$. If the number of roots is smaller than A, then it differs from A by an* even *number.*

Proof. The i-th derivative of $p(\cdot)$ in 0 and 1 is

$$p^{(i)}(0) = i!\, p_i$$
$$p^{(i)}(1) = \sum_{j=i}^{L-2} \frac{j!}{(j-i)!}\, p_j.$$

From the structure of the coefficients of $p(\cdot)$ it is not difficult to recognize that all the values in the sequence $p(1),\, p'(1),\, \dots,\, p^{(L-2)}(1)$ are positive for any L and $d_c \geq 3$. Hence,

$B = 0$ and the number of roots of $p(\cdot)$ between 0 and 1 is not greater than the number A of sign changes in the sequence $p_0, 1!p_1, \ldots, (L-2)!\,p_{L-2}$. This is equal to the number of sign changes in $p_0, p_1, \ldots, p_{L-2}$. By directly applying Theorem 4.2, we obtain the statement. We also observe that $p_{L-2} = [(L-1)d_c - L]\,\lambda_L$ is always positive. □

Recall that, for the p-positive distributions, $p(\cdot)$ has only roots, between 0 and 1, with even multiplicity. Then, the following necessary condition for a check regular distribution to be a p-positive distribution is obtained from Corollary 4.1.

Corollary 4.2. *Necessary condition for a check regular distribution to be a p-positive distribution is that the number of sign changes in the sequence $p_0, p_1, \ldots, p_{L-2}$ is even.*

In the next section, some examples of threshold optimizations for p-positive distributions are provided. These results reveal that, for some values of the code rate R, check degree d_c and maximum variable degree L, the p-positive distributions are characterized by very good thresholds. In some cases, the threshold is very close to that one of the best known distributions, under the same set of constraints.

4.4 Optimization of p-positive Degree Distributions

4.4.1 Optimization with Constrained Differential Evolution

The threshold optimization problem for the p-positive distributions can be in principle performed numerically, using a constrained version of the differential evolution (DE) algorithm. Specifically, a number $N_D + 1$ of values $x_i = (1/N_D) \cdot i$, for $i = 0, \ldots, N_D$, are first chosen, for sufficiently large N_D, and then the DE optimization tool is run for given code rate, given check degree and given maximum variable degree, and with the additional set of constraints $p(x_i) \geq 0$ for all i.

We performed this constrained DE optimization for $R = 1/2$, $L = 20$ and $d_c = \{6, 7, 8\}$. Under the same set of constraints, we also performed the DE optimization, without imposing the p-positive bound. The results of this search are shown in Table 4.1 for $d_c = 6$ and $d_c = 7$. In both cases, the best p-positive distribution exhibits a threshold that is only slightly worse than that one of the best check regular distribution obtained removing the p-positive constraint, with the advantage represented by the closed form threshold expression.

This conclusion is no longer valid for $d_c = 8$. In fact, for this check node degree, the best found p-positive threshold was $q^* = 0.469592$, while the best found check regular threshold was $q^* = 0.491988$. This means that the p-positive constraint is not "compatible" with all the possible sets of bounds on the code rate, check degree and maximum

	$d_c = 6$		$d_c = 7$	
	p-pos.	not p-pos.	p-pos.	not p-pos.
λ_2	0.415884	0.415273	0.339162	0.338843
λ_3	0.165968	0.160268	0.138401	0.140058
λ_4	0.095028	0.142202	0.104711	0.104198
λ_5	0.106071		0.033138	
λ_6		0.034597		0.087264
λ_7			0.166166	0.104669
λ_8	0.070638	0.247661		
λ_9	0.146412			
λ_{14}			0.104300	
λ_{16}				0.224968
λ_{19}			0.114122	
Thresholds				
q^*	0.480904	0.481524	0.491407	0.491740

Table 4.1: Optimal p-positive and not p-positive distributions and thresholds for $R = 1/2$, $L = 20$ and $d_c = \{6, 7\}$.

variable degree. When this compatibility holds, the p-positive distributions exhibit very good thresholds. In the other cases, they exhibit a threshold loss with respect to the best known distributions.

4.4.2 Optimization Based on the Fourier-Budan Theorem

In Section 4.4.1, the p-positive distribution optimization has been performed according to a constrained version of the DE algorithm. Some examples of p-positive threshold optimization, based on a totally different technique, are presented next. More specifically, it is shown that the necessary condition, developed in the previous section (Corollary 4.2), can be exploited in order to perform the optimization process without using numerical tools. This is possible for particular structures of the variable nodes degree distribution.

As a case study, let us consider check regular distributions with variable distribution in the form $\lambda(x) = \lambda_2\, x + \lambda_3\, x^2 + \lambda_K\, x^{K-1} + \lambda_L\, x^{L-1}$ with $L \geq 7$ and $4 < K < L - 1$. For any choice of R, d_c, K and L, ϵ_{opt} will denote the minimum ϵ such that a corresponding p-positive $(1-\epsilon)$ capacity achieving distribution of rate R exists, for the given parameters.

For such $\lambda(\cdot)$, the coefficients of $p\,(\cdot)$ are:

$$\begin{aligned}
p_{\,0} &= (d_c - 2)\lambda_2 - (2d_c - 2)\lambda_3 \\
p_{\,1} &= (2d_c - 3)\lambda_3 \\
p_{\,K-3} &= -[(K-1)d_c - (K-1)]\lambda_K \\
p_{\,K-2} &= [(K-1)d_c - K]\lambda_K \\
p_{\,L-3} &= -[(L-1)d_c - (L-1)]\lambda_L \\
p_{\,L-2} &= [(L-1)d_c - L]\lambda_L.
\end{aligned}$$

According to Corollary 4.2, necessary (but not sufficient) condition for the degree distribution to be p-positive is that the number A of sign changes in the sequence $p_{\,0}, p_{\,1}, p_{\,K-3}$, $p_{\,K-2}, p_{\,L-3}, p_{\,L-2}$ is even. In the specific case under analysis, the following inequalities are always true: $p_{L-2} > 0$, $p_{L-3} < 0$, $p_{K-2} > 0$, $p_{K-3} < 0$, $p_1 > 0$. Then, if and only if $p_{\,0} > 0$, the condition that A is even is satisfied (specifically, it results $A = 4$).

Suppose to fix the code rate R, the check nodes degree d_c and the active variable degrees (K and L). From the constraints $\sum_{i=2}^{L} \lambda_i = 1$ and $R = 1 - d_c/(\sum_{i=2}^{L} \lambda_i/i)$, and from (4.11), expressions of λ_3 and λ_K as functions of ϵ and λ_L, namely $\lambda_3 = \lambda_3(\epsilon, \lambda_L)$ and $\lambda_K = \lambda_K(\epsilon, \lambda_L)$, can be found. Then, the necessary condition for the distribution to be p-positive is given by the following set of inequalities:

$$\begin{aligned}
&0 < \lambda_2(\epsilon) < 1 \qquad 0 < \lambda_3(\epsilon, \lambda_L) < 1 \\
&0 < \lambda_K(\epsilon, \lambda_L) < 1 \qquad 0 < \lambda_L < 1 \\
&p_0(\epsilon, \lambda_L) > 0.
\end{aligned}$$

The solution of this set of inequalities identifies a region on the plane $\epsilon - \lambda_L$. This region, denoted by $\mathcal{M}$, will be called the *permitted region* (since no p-positive distribution can exist outside this set). Some examples of permitted regions, for different values of d_c, K and L, all corresponding to $R = 1/2$, are depicted in Fig. 4.1. If $(\epsilon, \lambda_L) \in \mathcal{M}$, then the polynomial $p\,(\cdot)$ could have in principle 0, 2 or 4 real roots between 0 and 1 ($A = 4$).

Let $\epsilon_{\min}^{\mathcal{M}}$ be the minimum value of ϵ allowed for points within $\mathcal{M}$. If, for $\epsilon = \epsilon_{\min}$, at least a λ_L exists such that $(\epsilon_{\min}, \lambda_L) \in \mathcal{M}$, and such that $p(\cdot)$ is not negative between 0 and 1, then it follows $\epsilon_{\text{opt}} = \epsilon_{\min}$. On the contrary, if $p(x)$ is negative for some x between 0 and 1, for $\epsilon = \epsilon_{\min}$ and for any admitted λ_L, it results $\epsilon_{\text{opt}} > \epsilon_{\min}$. This is always the case when $\epsilon_{\min} = 0$. An approach to evaluate ϵ_{opt} in this case is described next.

The proposed approach is based on this observation: If $(\epsilon_{\text{opt}}, \hat{\lambda}_L)$ is a solution of the optimization problem (for some $\hat{\lambda}_L$), then at least one $\overline{x} \in (0,1)$ must exist such that

$$p\,(\overline{x}; \epsilon_{\text{opt}}, \hat{\lambda}_L) = 0 \tag{4.12}$$

$$p'(\overline{x}; \epsilon_{\text{opt}}, \hat{\lambda}_L) = 0, \tag{4.13}$$

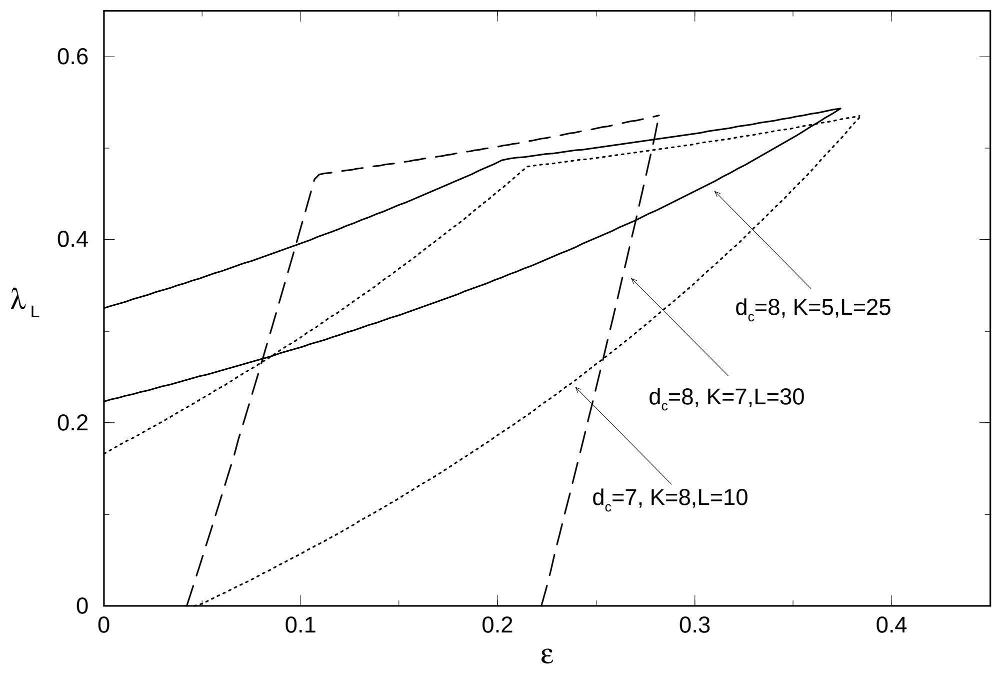


Figure 4.1: Permitted region for some values of (d_c, K, L).

where the dependence of the coefficients of $p(\cdot)$ on ϵ and λ_L have been explicitly indicated. Then, the search for the optimal distribution can be restricted to the points $(\epsilon, \lambda_L) \in \mathcal{M}$ for which $p(\overline{x}; \epsilon, \lambda_L) = 0$ and $p'(\overline{x}; \epsilon, \lambda_L) = 0$.

From these relationships, it is possible to obtain ϵ and λ_L as rational functions of $\overline{x}$, namely $\epsilon = \epsilon(\overline{x})$ and $\lambda_L = \lambda_L(\overline{x})$. Then, the technique consists in plotting the trajectory of the point $(\epsilon(\overline{x}), \lambda_L(\overline{x}))$ on the plane $\epsilon - \lambda_L$, as well as the permitted region $\mathcal{M}$. From this diagram it is usually possible to univocally determine ϵ_{opt}, as shown next with an example.

For instance, suppose $R = 1/2$, $d_c = 7$, $K = 5$ and $L = 17$. The rational functions $\epsilon = \epsilon(\overline{x})$ and $\lambda_{17} = \lambda_{17}(\overline{x})$ are:

$$\epsilon(\overline{x}) = \frac{4620 - 42880\,\overline{x} + 99260\,\overline{x}^2 - 72105\,\overline{x}^3 - 944384\,\overline{x}^{13} + 1351364\,\overline{x}^{14} + 4718775\,\overline{x}^{15} - 9931740\,\overline{x}^{16} + 4828850\,\overline{x}^{17}}{\overline{x}\left(-233280 + 442260\,\overline{x} - 204930\,\overline{x}^2 + 2467584\,\overline{x}^{12} - 4716684\,\overline{x}^{13} + 4588470\,\overline{x}^{14} - 4583880\,\overline{x}^{15} + 2228700\,\overline{x}^{16}\right)}$$
$$\lambda_L(\overline{x}) = \frac{85\left(-33 + 944\,\overline{x} - 1753\,\overline{x}^2 + 759\,\overline{x}^3\right)}{7\left(660 + 27200\,\overline{x} - 49000\,\overline{x}^2 + 18975\,\overline{x}^3 - 487424\,\overline{x}^{13} + 866864\,\overline{x}^{14} + 18615\,\overline{x}^{15} - 763980\,\overline{x}^{16} + 371450\,\overline{x}^{17}\right)}$$

The trajectory of $(\epsilon(\overline{x}), \lambda_L(\overline{x}))$ is depicted in Fig. 4.2, together with a detail of the permitted region. The trajectory of $(\epsilon(\overline{x}), \lambda_L(\overline{x}))$ crosses the permitted region for values of $\overline{x}$ between $\overline{x}_1 = 0.527434$, and $\overline{x}_2 = 0.735514$. The minimum-$\epsilon$ point on the trajectory segment $\overline{x}_1 \rightarrow \overline{x}_2$ is the intersection point between the trajectory and the upper boundary of $\mathcal{M}$. For this point, it results $\epsilon = \epsilon(\overline{x}_1) = 0.032242$. The corresponding polynomial

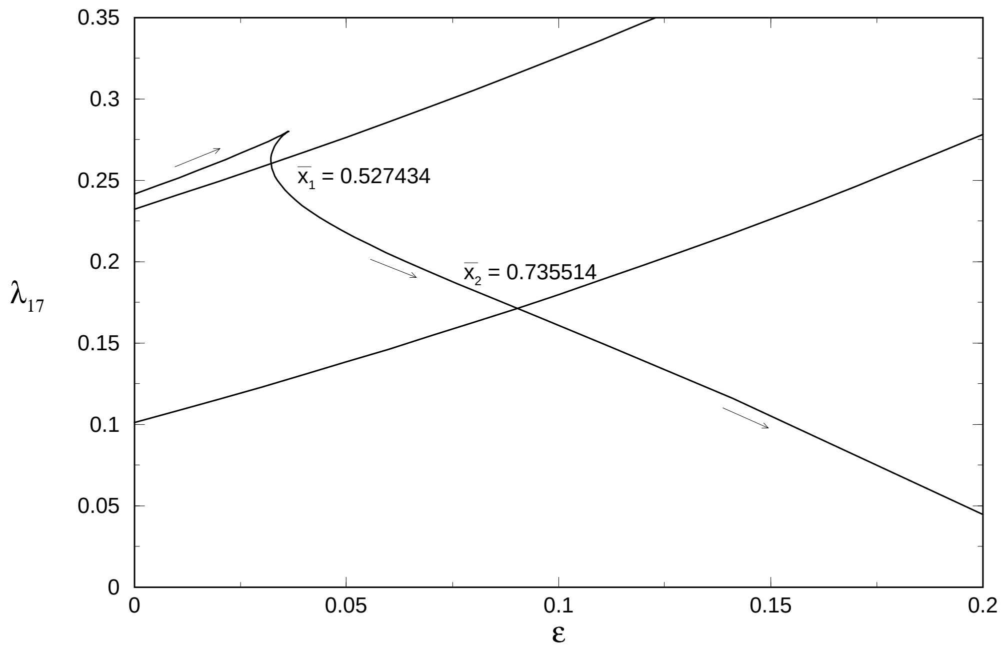


Figure 4.2: Diagram showing the permitted region and the trajectory of the point $(\epsilon(\overline{x}), \lambda_{17}(\overline{x}))$, for $R = 1/2$, $d_c = 7$, $K = 5$ and $L = 17$.

	$d_c = 6$, $L = 10$		$d_c = 7$, $L = 15$	
	p-pos.	DE	p-pos.	DE
λ_2	0.418913	0.415774	0.341501	0.339505
λ_3	0.167565	0.180916	0.142292	0.140214
λ_5	0.266696	0.248100	0.248395	0.259036
λ_L	0.146826	0.155210	0.267812	0.261244
Thresholds				
q^*	0.477426	0.480325	0.488041	0.490947

Table 4.2: Optimal $R = 1/2$ p-positive and not p-positive distributions for $d_c = \{6, 7\}$ with $\lambda(x) = \lambda_2\, x + \lambda_3\, x^2 + \lambda_5\, x^4 + \lambda_L\, x^{L-1}$.

$p\,(\cdot)$, whose graph is depicted in Fig. 4.3, does not assume negative values between 0 and 1. Hence, the minimum-ϵ point on the segment $\overline{x}_1 \rightarrow \overline{x}_2$ corresponds to a p-positive distribution: Then, $\epsilon_{\text{opt}} = 0.032242$. The threshold of the corresponding degree distribution is $q^* = [\lambda'(0)\rho'(1)] = (1 - \epsilon_{\text{opt}})(1 - R) = 0.483879$, the value of λ_{17} can be obtained from the function $\lambda_{17} = \lambda_{17}(\overline{x})$ and the values of λ_3 and λ_5 can be obtained from the functions $\lambda_3 = \lambda_3(\epsilon, \lambda_{17})$ and $\lambda_5 = \lambda_5(\epsilon, \lambda_{17})$.

By adopting a similar approach, we fixed $R = 1/2$, $K = 5$, and looked for the best p-positive distribution for $d_c = 6$ and $d_c = 7$. The optimal distributions were obtained for

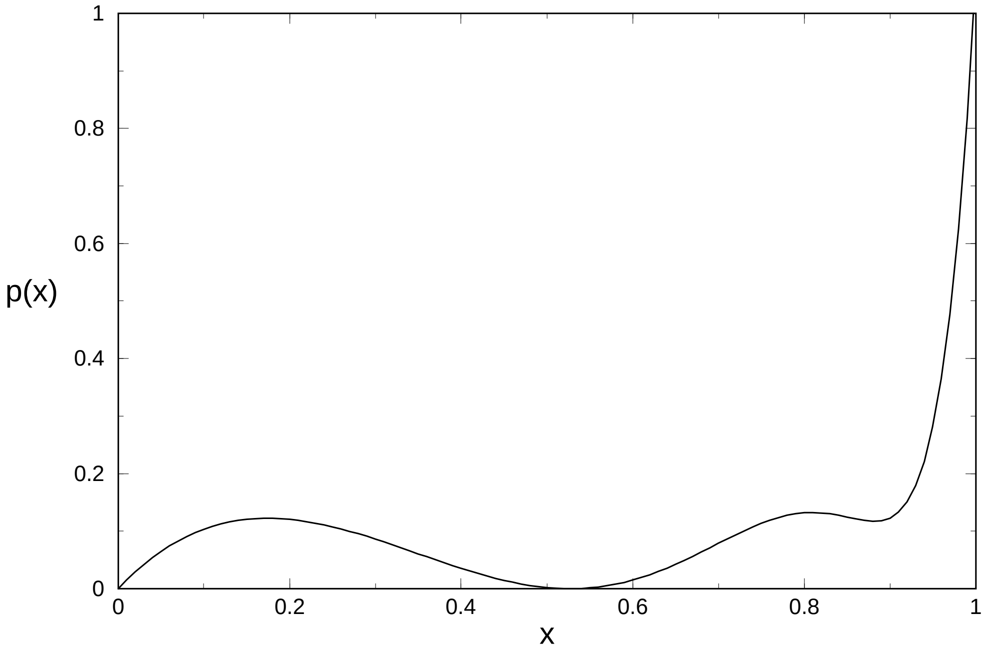


Figure 4.3: Plot of $p(\cdot)$ for the optimal $R = 1/2$ p-positive distribution corresponding to $d_c = 7$, $K = 5$ and $L = 17$.

$L = 10$ and $L = 15$, respectively. They are shown in Table 4.2, as well as the optimal check regular distributions obtained by running the DE tool, under the same set of constraints (R, d_c and active variable degrees), but without the p-positive constraint. The p-positive thresholds are only slightly lower than the not p-positive counterparts, and the degree distributions are quite similar. Again, the p-positive distributions have the advantage represented by the closed-form threshold expression $[\lambda'(0)\rho'(1)]^{-1}$.

4.5 The Binomial Distributions are p-positive

In [27], it was shown that capacity achieving sequences of degree distributions for the BEC can be constructed according to the binomial degree distribution. This is a check regular distribution, whose variable distribution is in the form $\lambda(x) = \sum_{i=2}^{L} \frac{\alpha\binom{\alpha}{i-1}(-1)^i}{\alpha - L\binom{\alpha}{L}(-1)^{L+1}} x^{i-1}$, where $\binom{\alpha}{N} = \alpha(\alpha-1)\dots(\alpha-N+1)/(N!)$, and $\alpha = (d_c-1)^{-1}$. Recognizing the binomial degree distributions as part of the p-positive family, the following theorem states that this family can achieve the BEC capacity.

Theorem 4.3. *Any binomial degree distribution is p-positive.*

Proof. Recall that, for $i = 2, \dots, L-1$, the $(i-2)$-th coefficient of $p(\cdot)$ is $p_{i-2} = \omega_i\lambda_i -$

$(\omega_{i+1}+1)\,\lambda_{i+1}$. For the binomial distribution, it is

$$\omega_i\,\lambda_i = [(d_c-1)(i-1)-1]\,\frac{\alpha\binom{\alpha}{i-1}(-1)^i}{\alpha - L\binom{\alpha}{L}(-1)^{L+1}}.$$

Furthermore, it is

$$\begin{aligned}(\omega_{i+1}+1)\,\lambda_{i+1} &= (d_c-1)\,i\,\frac{\alpha\binom{\alpha}{i}(-1)^{i+1}}{\alpha - L\binom{\alpha}{L}(-1)^{L+1}}\\ &= (d_c-1)\,\frac{(i-1-\alpha)\,\alpha\binom{\alpha}{i-1}(-1)^i}{\alpha - L\binom{\alpha}{L}(-1)^{L+1}}\\ &= [(d_c-1)\,(i-1)-\alpha\,(d_c-1)]\,\frac{\alpha\binom{\alpha}{i-1}(-1)^i}{\alpha - L\binom{\alpha}{L}(-1)^{L+1}}\\ &= [(d_c-1)(i-1)-1]\,\frac{\alpha\binom{\alpha}{i-1}(-1)^i}{\alpha - L\binom{\alpha}{L}(-1)^{L+1}},\end{aligned}$$

where the second equality is due to the fact that $\binom{\alpha}{i} = \binom{\alpha}{i-1}\frac{\alpha-i+1}{i}$, and the last equality to the fact that $\alpha = (d_c-1)^{-1}$. Thus, for $i = 0, \dots, L-1$, $\omega_i\,\lambda_i = (\omega_{i+1}+1)\,\lambda_{i+1}$, i.e. $p_{\,i-2} = 0$. It follows $p\,(x) = \omega_L\,\lambda_L x^{L-2}$, which is always positive for $x \in (0,1]$. □

4.6 Final Remarks

In this chapter, a special family of LDPC degree distributions has been presented. The main feature of these distributions is the possibility to express in closed form their decoding threshold on the BEC, under iterative decoding. More specifically, the threshold admits the simple closed form $[\lambda'(0)\rho'(1)]^{-1}$. This family is a subset of the class of check-regular distributions, and the distributions within this family have been called p-positive distributions. A simple necessary condition for a check regular distribution to belong to this class has been obtained, by invoking some known results about the real roots of polynomials.

Three additional properties of the proposed distributions have been highlighted. The first one is their very good threshold under some set of constraints. This property is not general, depending on the specific imposed set of constraints. The second one is the possibility, for particular structures of the variable distribution, to optimize the distribution without necessarily using the numerical optimization tools which are usually exploited. The third one is the possibility to achieve the BEC capacity with sequences belonging to the proposed family.

4.7 Appendix: Fixed Points Analysis of $\phi(\cdot)$ Function

The fixed points of a real function $f : \mathbb{R} \to \mathbb{R}$ can be classified as attractors and repellers. The attractors are those fixed points x_{fp} towards which the *fixed point iteration* $x_{n+1} = f(x_n)$ converges, for appropriate starting value x_0. On the other hand, the repellers are those fixed points x_{fp} which cannot be computed through the fixed point iteration, since there is no convergence towards them for any $x_0 \neq x_{\text{fp}}$. The attractors and repellers are further classified in strong/weak attractors and strong/weak repellers, depending on the convergency or divergency rapidity of the fixed point iteration. If x_{fp} is a fixed point for $f(\cdot)$, then it is a strong attractor if $0 < f'(x_{\text{fp}}) < 1$, a strong repeller if $f'(x_{\text{fp}}) > 1$, a weak attractor if $-1 < f'(x_{\text{fp}}) < 0$ and a weak repeller if $f'(x_{\text{fp}}) < -1$.

Lemma 4.1. *For any edge-oriented variable distribution $\lambda(x) = \sum_{i\geq 2} \lambda_i x^{i-1}$ of a standard LDPC code:*

$$\lim_{x\to 0+} \frac{\lambda(x)\lambda''(x)}{(\lambda'(x))^2} = \frac{m-2}{m-1},$$

where $m = \min\{i \,|\, \lambda_i \neq 0\}$.

Proof. Suppose first $\lambda_2 \neq 0$ ($m = 2$), and define $p = \min\{i > 2 \,|\, \lambda_i \neq 0\}$. It follows $\lambda(x) = \lambda_2 x + \lambda_p x^{p-1} + o(x^{p-1})$, $\lambda'(x) = \lambda_2 + (p-1)\lambda_p x^{p-2} + o(x^{p-2})$ and $\lambda''(x) = (p-1)(p-2)\lambda_p x^{p-3} + o(x^{p-3})$. Since $\lambda'(x)$ converges to $\lambda_2 > 0$ and $\lambda(x)$ to 0 as x tends to 0,

$$\lim_{x\to 0+} (\lambda(x)\lambda''(x))/(\lambda'(x))^2 = 0$$

for every $p \geq 3$, that proves the lemma for $m = 2$.

Suppose now $m \geq 3$ ($\lambda_2 = 0$). We find $\lambda(x) = \lambda_m x^{m-1} + o(x^{m-1})$, $\lambda'(x) = (m-1)\lambda_m x^{m-2} + o(x^{m-2})$ and $\lambda''(x) = (m-1)(m-2)\lambda_m x^{m-3} + o(x^{m-3})$, from which

$$\begin{aligned}\lim_{x\to 0+} \frac{\lambda(x)\lambda''(x)}{(\lambda'(x))^2} &= \lim_{x\to 0+} \frac{(m-1)(m-2){\lambda_m}^2\, x^{2m-4}}{(m-1)^2{\lambda_m}^2\, x^{2m-4}} \\ &= \frac{m-2}{m-1}\end{aligned}$$

for every choice of $m \geq 3$. □

Property 4.1. *The function $\phi(\cdot)$ has no weak attractors/repellers in the interval* $(0, 1)$.

Proof. The derivative of $\phi(\cdot)$ can be put in the form:

$$\begin{aligned}\phi'(x) = \frac{d_c-1}{d_c-2} &\cdot \Big(\frac{\lambda(x)}{(d_c-1)\lambda'(x)} + 1 - x\Big)^{\frac{1}{d_c-2}} \\ &\cdot \Big(\frac{d_c-2}{d_c-1} + \frac{\lambda(x)\lambda''(x)}{(d_c-1)(\lambda'(x))^2}\Big),\end{aligned}$$

which is a strictly positive function for $x \in (0,1)$; hence, $\phi(\cdot)$ is monotonically increasing in $(0,1)$, and its fixed points can be just strong attractors and strong repellers. □

Property 4.2. *Let $\lambda_2 = 0$. Then there is some positive and odd integer N such that $\phi(\cdot)$ has exactly $\frac{N+1}{2}$ strong attractors and $\frac{N-1}{2}$ strong repellers in $(0,1)$.*

Proof. It is simple to show that, for every choice of $\lambda(x) = \sum_{i\geq 2} \lambda_i x^{i-1}$, it results

$$\lim_{x\to 0+} \lambda(x)/\lambda'(x) = 0. \tag{4.14}$$

From this result, and from Lemma 4.1, it follows:

$$\lim_{x\to 0+} \phi(x) = 0$$

$$\begin{aligned}\lim_{x\to 0+} \phi'(x) &= \frac{d_c-1}{d_c-2}\cdot\left(\frac{d_c-2}{d_c-1} + \frac{1}{d_c-1}\cdot\frac{m-2}{m-1}\right)\\ &= 1 + \frac{1}{d_c-2}\cdot\frac{m-2}{m-1} > 1\,.\end{aligned}$$

Thus, if $\lambda_2 = 0$, for sufficiently small and positive values of x, $\phi(x) > x$. Since $\phi(\cdot)$ is a continuous function and

$$\phi(1) = 1 - \left[(d_c-1)\lambda'(1)\right]^{-\frac{d_c-1}{d_c-2}} < 1, \tag{4.15}$$

an odd number N of fixed points must exist in $(0,1)$ and the smallest of them is an attractor. Since attractors and repellers must be necessarily alternated, the total number of strong attractors and repellers is respectively $\frac{N+1}{2}$ and $\frac{N-1}{2}$. □

Property 4.3. *Let $\lambda_2 > 0$. Then:*

1. *if $\frac{\lambda_3}{\lambda_2} > \frac{1}{2}\cdot\frac{d_c-2}{d_c-1}$, then there is some positive and odd integer N such that $\phi(\cdot)$ has $\frac{N+1}{2}$ strong attractors and $\frac{N-1}{2}$ strong repellers in $(0,1)$;*

2. *if $\frac{\lambda_3}{\lambda_2} < \frac{1}{2}\cdot\frac{d_c-2}{d_c-1}$, then there is some positive and even integer N such that $\phi(\cdot)$ has $\frac{N}{2}$ strong attractors and $\frac{N}{2}$ strong repellers in $(0,1)$.*

Proof. From Lemma 4.1 and (4.14) it follows that $\lim_{x\to 0+} \phi'(x) = 1$ if $\lambda_2 > 0$. Furthermore:

$$\lim_{x\to 0+} \phi''(x) = \frac{2(d_c-1)\lambda_3 - (d_c-2)\lambda_2}{(d_c-1)(d_c-2)\lambda_2}\,.$$

Hence, $\lim_{x\rightarrow 0+} \phi''(x) > 0$ if and only if $\frac{\lambda_3}{\lambda_2} > \frac{1}{2} \cdot \frac{d_c-2}{d_c-1}$. Under this condition, $\phi(x) > x$ for small and positive x. It follows that $\phi(\cdot)$ must have $\frac{N+1}{2}$ attractors and $\frac{N-1}{2}$ repellers in $(0,1)$ for odd N, with the smallest fixed point being a strong attractor. On the contrary, if $\frac{\lambda_3}{\lambda_2} < \frac{1}{2} \cdot \frac{d_c-2}{d_c-1}$, then $\phi(x) < x$ for small and positive values of x. In this case, the smallest fixed point of $\phi(\cdot)$ in $(0,1)$ must be a strong repeller; since (4.15) is still valid, the number N of fixed points of $\phi(\cdot)$ in $(0,1)$ must be necessarily even, with an equal number $\frac{N}{2}$ of strong attractors and repellers. $\square$

Chapter 5

Construction of Near-Optimum Burst Erasure Correcting Low-Density Parity-Check Codes

5.1 Introduction

Recently, the problem of designing low-complexity codes for transmission on burst erasure channels, especially low-density parity-check (LDPC) codes [3], has gained a certain interest (e.g. [62, 63, 23, 64, 65, 66, 67, 68, 69, 70]). This demand for burst erasure correcting codes can be explained by the fact that such codes are quite interesting for several (vey heterogeneous) applications, including magnetic storage, wireless communications on correlated fading channels, and even space communications. For instance, LDPC codes with good properties in terms of correction of single bursts of erasures have been shown in [62] and [63] to represent a promising solution, respectively, for the error control system in magnetic recording applications, and for communication on correlated Rayleigh fading channels. Furthermore, it is currently under investigation the possibility to implement LDPC-like codes at the upper layers of the communication stack, for correcting bursts of packet erasures in space and satellite communication links [23, 64], in order to heavily reduce the use of automatic repeat request (ARQ) protocols.

In a recent paper, it has been shown that practically any (N, K) linear block code can be used to correct any single burst of $N - K$ or less erasures, thus achieving the optimal correction capability of single bursts of erasures [71]. More specifically, it has been proved that, under very mild assumptions, it is possible to obtain a (redundant) representation of the parity-check matrix for the code (called parity-check matrix in burst correction form) that permits to recover from any pattern of $N-K$ or less contiguous erasures, by applying a decoding algorithm whose computational complexity is quadratic in the codeword length N. The same code can be then used in a communication system for erasure recovery both in scenarios with independent bit erasures and in scenarios where the erasures occur in bursts. The decoding is performed according to a two-step process: the received sequence is first processed by the burst correcting algorithm operating on the parity-check matrix in burst correction form, and then by the decoding algorithm for independent erasures correction, operating on a different parity-check matrix representation.

This very general technique can be in principle applied also to LDPC codes. In this case, in the first step of the decoding process, the burst erasure correcting decoder is applied, with quadratic complexity, to the parity-check matrix in burst correction form; in the second step of the decoding process, the iterative decoding [26] is performed on a low-density representation of the parity-check matrix.

In this chapter, the possibility to construct LDPC codes, capable to perform recovery of both independent erasures and bursts of erasures by exploiting *only* the iterative decoder, is investigated. More specifically, the approach consists in starting from an LDPC code parity-check matrix, usually designed in order to achieve good performance on the (memory-less) BEC[1], and then properly modifying it in order to make it suitable also for the *iterative* correction of single bursts of erasures.

The performance of this single-step LDPC iterative decoding process is in general suboptimal, in burst scenarios, with respect to the two-step technique proposed in [71], because of the suboptimal iterative burst correction. However, avoiding the quadratic complexity burst correction step (which becomes an issue for long LDPC codes) it only requires linear in N decoding complexity [26]. Furthermore, the proposed algorithm used for making the parity-check matrix suitable to iterative correction of bursts of erasures, besides being extremely simple, turns out to be also extremely effective, generating finite length LDPC codes whose erasure burst correction capability is very close to its maximum possible value. Moreover, if the input parity-check matrix is designed for achieving good performance on the BEC, the optimized code can be used for transmission in both burst and independent erasure scenarios.

This chapter is strictly related to a number of recent works, i.e. [62, 63, 65, 68, 69]. In [62], a key parameter is proposed as a measure of the robustness of an LDPC code to single bursts of erasures, namely the *maximum guaranteed resolvable erasure burst length*, denoted by $L_{\max}$. A general definition of the parameter $L_{\max}$, valid for any code, can be given as follows.

Definition 5.1. *For a given code, a given parity-check matrix representation, and a given decoding algorithm, the maximum guaranteed resolvable erasure burst length ($L_{\max}$) is the maximum length of an erasure burst which is correctable independently of its position within the codeword.*

[1]Throughout the chapter the acronym BEC will be used to denote the standard memory-less binary erasure channel.

As explicitly remarked in this definition, $L_{\max}$ is not unique for a given code, heavily depending on both the decoding algorithm and the parity-check matrix representation. For instance, for a given parity-check matrix representation of an LDPC code, the value of $L_{\max}$ is not the same with respect to the standard iterative decoding algorithm or to the improved decoding algorithm proposed in [72]. In the sequel, the standard iterative decoder for LDPC codes will be always considered. In [62], an algorithm for the efficient computation of $L_{\max}$ for LDPC codes under iterative decoding is developed.

An estimate of the optimal value of $L_{\max}$ for LDPC codes, under standard iterative decoding, has been proposed in [63]. Consider an LDPC code, and let q^* denote the associated asymptotic decoding threshold [7] on the BEC. Then, for sufficiently large codeword length N, there exists some proper permutation of the parity-check matrix columns such that $L_{\max}/N \simeq q^*$. Then, $\lfloor q^* N \rfloor$ can be used as an estimate of the maximum value of $L_{\max}$ that can be obtained for the length-N LDPC code, by permuting the parity-check matrix columns.

The maximum guaranteed resolvable erasure burst length for an LDPC code, under iterative decoding, has a strong dependence on the *stopping sets* present in the bipartite graph. The concept of stopping set has been first introduced in [30], and further investigated in [54]. By definition, a stopping set is any set of variable nodes such that any check node connected to this set is connected to it at least twice. In [30], it is also proved that the union of stopping sets is a stopping set, so that it is possible to define a maximal stopping set included in a subset of the variable nodes, as the union of all the stopping sets included in the subset. The residual erased variable nodes (after iterative decoding) constitute the maximal stopping set included in the original erasure pattern. Hence, a decoding failure takes place whenever the erasure pattern due to the channel contains a stopping set.

The relation between stopping sets and $L_{\max}$ has been addressed in [65]. Let $\mathcal{G}$ be the LDPC code bipartite graph, and let $\mathcal{V} = \{V_0, V_1, \ldots, V_{n-1}\}$ be the set of the variable nodes. If $\mathcal{S} = \{V_{i_1}, V_{i_2}, \ldots, V_{i_t}\}$, with $i_1 < i_2 < \cdots < i_t$, is a stopping set, then the span of $\mathcal{S}$ is defined as $1 + |i_t - i_1|$. Denoting by $\mu(\mathcal{G})$ the minimum span of stopping sets (i.e. the minimum among the spans of all the stopping sets of $\mathcal{G}$), it follows that $L_{\max} = 1 + \mu(\mathcal{G})$. The span of the stopping sets, a concept of no interest on the BEC, heavily affects the code performance when the erasures occur in bursts.

The concept of span of stopping sets is considered in [67], where a lower bound is found for the minimum span of stopping sets for any regular LDPC code. More specifically, it is proved that the minimum span of stopping sets satisfies $\mu(\mathcal{G}) \geq \delta$, where δ is the minimum *zero span* in the parity-check matrix. A zero-span is defined as a sequence of consecutive zeroes in a parity-check matrix row; in terms of $L_{\max}$ the bound is $L_{\max} \geq \delta + 1$. In [67], a technique for constructing regular LDPC codes with good $L_{\max}$ is also presented.

Finally, some results about protograph based LDPC codes on burst erasure channels are presented in [68].

The main contribution of this work is the development of a greedy algorithm, which is able to modify the parity-check matrix of an LDPC code, designed for erasure correction on the BEC, in order to make it suitable also for the iterative correction of single erasure bursts. The present work extends and improves some results from [69], where a former version of the algorithm was presented. The developed algorithm is called *pivot searching and swapping* (PSS) algorithm [73], since it is based on the search and swap of the *pivots* of stopping sets. The novel concept of pivot of a stopping set will be introduced in the next section. According to the proposed approach, the parity-check matrix for iterative burst correction is generated by only performing column permutations on the input parity-check matrix. Hence, the sparseness of the matrix is not altered by the algorithm, and the iterative decoder applied to the new parity-check matrix leads, on the BEC, to exactly the same performance as the original one. If the parity-check matrix received in input by the algorithm is designed for achieving good performance on the BEC, then the code obtained after the application of the PSS algorithm provides good joint correction of independent erasures and single erasure bursts, within a single-step decoding scheme, only exploiting the iterative decoder.

5.2 Pivots of Stopping Sets

In this section, the novel concept of pivots of a stopping set is introduced. It is given proof that the minimum number of pivots for any stopping set is two, and an efficient algorithm for finding some pivots of a given stopping set is developed.

For any LDPC code, and for any stopping set of the LDPC code, we define *subgraph induced by the stopping set* the bipartite graph composed of the variable nodes which are part of the stopping set, the check nodes connected (necessarily at least once) to these variable nodes, and the edges connecting such variable nodes and such check nodes. The key concept of *pivot* of a stopping set is defined next.

Definition 5.2. *Let $\mathcal{G}$ be the subgraph induced by a stopping set $\mathcal{S}$ of an LDPC code. A variable node V is called pivot of the stopping set if the following property holds: if the value of V is known and the value of all the other variable nodes of $\mathcal{G}$ is unknown, then the iterative decoder applied to $\mathcal{G}$ is able to successfully recover from the erasure pattern.*

According to this definition, if the variable node V is pivot for a stopping set $\mathcal{S}$, then the set of variable nodes $\mathcal{S}/\{V\}$ is not a stopping set for the LDPC code and contains no stopping sets.

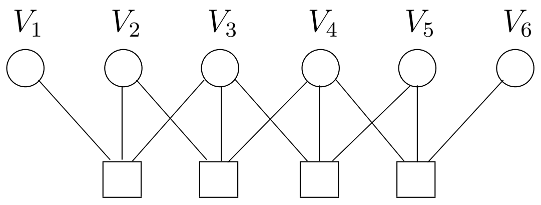


Figure 5.1: Example of subgraph induced by a stopping set of size 6, with no pivots.

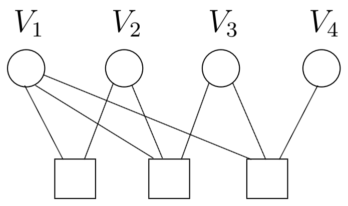


Figure 5.2: Example of subgraph induced by a stopping set of size 4. The variable nodes V_1 and V_2 are pivots for the stopping set, the variable nodes V_3 and V_4 are not pivots.

As recalled in the previous section, the iterative decoder is not able to recover from a starting erasure pattern caused by the channel when this erasure pattern includes at least one stopping set. In particular, the set of variable nodes which remain uncorrected at the end of the decoding process is the maximal stopping set included in the starting erasure pattern, i.e. the union of all the stopping sets included in it. The residual graph at the end of the decoding process is the subgraph induced by the stopping set. What we point out with the above definition is that, among the variable nodes in this residual graph, one should distinguish between pivot and non-pivot variable nodes: if the value of at least one of the pivots was known, then the decoding would be successful. This is the basic idea exploited in the optimization algorithm presented in the next section.

It is important to underline that not all the stopping sets have pivots. For instance, the stopping set of size 6 with the induced subgraph depicted in Fig. 5.1 has no pivots, while the stopping set of size 4 with the induced subgraph depicted in Fig. 5.2 has two pivots, V_1 and V_2. The concept of *span of pivots*, defined next, will be used in Section 5.3.

Definition 5.3. *Let $\mathcal{S}$ be a stopping set with pivots, and let V_p and V_q be, respectively, the pivot of $\mathcal{S}$ with minimum index and the pivot of $\mathcal{S}$ with maximum index. Then, the span of the pivots of $\mathcal{S}$ is then defined as $q - p + 1$* [2].

The next lemma points out an important property of the structure of stopping sets characterized by the presence of pivots.

Lemma 5.1. *No stopping set $\mathcal{S}$ with pivots exists whose induced subgraph is composed by disjoint graphs.*

[2]The fact that a stopping set with pivots has at least two pivots will be proved in Theorem 5.1.

Proof. It is well known that the union of stopping sets is a stopping set [30]. Hence, the union of stopping sets with unconnected induced subgraphs is a stopping set. Let $\mathcal{G}$ be the subgraph induced by a stopping set, and let $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$, with $\mathcal{G}_1$ and $\mathcal{G}_2$ unconnected. In such a condition, even if a variable node V_α is pivot with respect to $\mathcal{G}_1$, it cannot be pivot for the whole stopping set, because no variable node in $\mathcal{G}_2$ can be corrected from the knowledge of the value of V_α only. Analogously, even if a variable node V_β is pivot with respect to $\mathcal{G}_2$, it cannot be pivot for the whole stopping set. Hence, the stopping set has no pivots. □

For a given stopping set of an LDPC code, the problem of finding all the stopping set pivots could be in principle solved by considering the subgraph induced by the stopping set, and by trying, for each variable node, whether the property expressed by Definition 5.2 is verified. However, there are some complexity issues when following this approach. In fact, this technique would be computationally onerous, especially for stopping sets with large size, and when iteratively used as a subroutine of some algorithm (like that one proposed in the next section). The complexity of this pivot searching algorithm could be reduced by exploiting the following lemma, which defines a necessary condition for a variable node to be a stopping set pivot.

Lemma 5.2. *Necessary condition for a variable node belonging to a stopping set $\mathcal{S}$ to be pivot for $\mathcal{S}$, is that the variable node is connected to at least one check node with degree 2 in the subgraph induced by $\mathcal{S}$.*

Proof. If no such check node is connected to the variable node, even if the value of the variable node is known, no further correction can be performed by the iterative decoder. In fact, after the elimination from the graph of the variable node and of all the edges connected to it, every check node continues to have a degree at least 2. □

According to Lemma 5.2, the search can be restricted only to the variable nodes which are connected to at least one check node of degree 2 in the subgraph induced by the stopping set.

We propose next an alternative and more efficient algorithm for the search of stopping set pivots. This algorithm is in general not able to find all the pivots of a given stopping set, and its success is bound to the condition that at least one pivot for the stopping set is already available. However, for the purposes of the optimization algorithm of LDPC on burst erasure channels described in the next section, where two pivots for each stopping set are always available, this algorithm comes out to be extremely effective. The proposed pivot searching algorithm is based on the following lemma. It defines a simple, sufficient condition for a variable node to be a stopping set pivot.

Lemma 5.3. *Sufficient condition for a variable node V_α belonging to a stopping set $\mathcal{S}$ to be pivot for $\mathcal{S}$, is that there exists some $V_\beta \in \mathcal{S}$ and some check node such that V_β is pivot for $\mathcal{S}$, the check node has degree 2 in the subgraph induced by $\mathcal{S}$ and it is connected to V_α and V_β.*

Proof. Let C be the check node connected to V_α and V_β, with degree 2 in the subgraph induced by $\mathcal{S}$. If the value of the variable node V_α is known, and the value of all the other variable nodes in $\mathcal{S}$ is unknown, then C is capable to correct the variable node V_β. Since V_β is a pivot of $\mathcal{S}$ by hypothesis, then all the variable nodes of the stopping set will be corrected. □

Combining Lemma 5.2 and Lemma 5.3 it is possible to prove the following result.

Theorem 5.1. *If a stopping set $\mathcal{S}$ has pivots, then it has at least two pivots.*

Proof. Let the variable node V_α be a pivot of $\mathcal{S}$. For Lemma 5.2, V_α must have at least one connection towards a check node C, with degree 2 in the subgraph induced by $\mathcal{S}$. Let V_β be the second variable node connected to C. From Lemma 5.3, this is sufficient to conclude that V_β is a pivot of $\mathcal{S}$. Thus, the number of pivots is at least two. □

The variable nodes V_α and V_β in the statement of Lemma 5.3 will be referred to as neighboring pivots. If it is known that the variable node V is a pivot of a certain stopping set, then all its neighboring pivots can be found by looking, among the check nodes connected to V, for check nodes with degree 2 in the subgraph induced by the stopping set. Based on Lemma 5.3, we propose the following pivot searching algorithm for a stopping set $\mathcal{S}$ of an LDPC code. As remarked above, the hypothesis is that at least one pivot of the stopping set is already available at the beginning of the algorithm.

Pivot Searching Algorithm.

- [*Initalization*] Set $\mathcal{P}^{(0)}$ equal to the available (non-empty) set of pivots of $\mathcal{S}$. Set $\hat{\mathcal{P}}^{(0)} = \mathcal{P}^{(0)}$.
- [$\mathcal{P}^{(\ell)}$ *expansion*] For each stopping set pivot $V \in \hat{\mathcal{P}}^{(\ell)}$, apply Lemma 5.3 in order to find the set $\hat{\mathcal{P}}(V)$ of the neighboring pivots of V. Set

$$\mathcal{P}^{(\ell+1)} = \Big(\bigcup_V \hat{\mathcal{P}}(V)\Big) \cup \mathcal{P}^{(\ell)}. \tag{5.1}$$

Set

$$\hat{\mathcal{P}}^{(\ell+1)} = \mathcal{P}^{(\ell+1)} / \mathcal{P}^{(\ell)}. \tag{5.2}$$

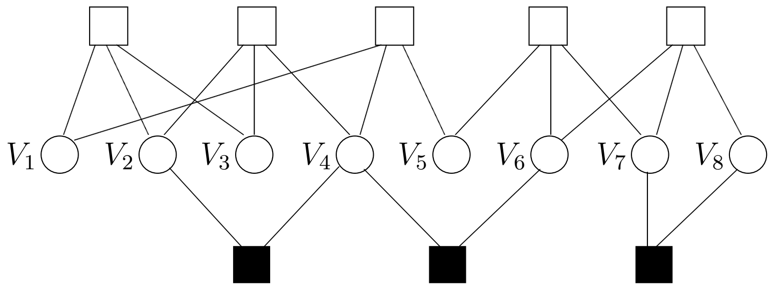


Figure 5.3: Example of subgraph induced by a size-8 stopping set with pivots $\{V_2, V_4, V_6, V_7, V_8\}$. If $\mathcal{P}^0 = \{V_2\}$, then the set of pivots found by the pivot searching algorithm is $\{V_2, V_4, V_6\}$.

- [*Stopping criterion*] If $\hat{\mathcal{P}}^{(\ell+1)}$ is equal to the empty set, stop and return $\mathcal{P}^{(\ell)}$. Else, set $\ell = \ell + 1$ and goto the $\mathcal{P}^{(\ell)}$ *expansion* step.

If only one pivot is available at the beginning of the algorithm ($|\mathcal{P}^{(0)}| = 1$), since at least one neighboring pivot must exist for the available pivot, the minimum number of pivots returned by the algorithm is 2. At each step of the algorithm, the sufficient condition expressed by Lemma 5.3 is applied to the new pivots found at the previous step. The algorithm is stopped as soon as no new pivots are found.

Example 5.1. *Let us consider the stopping set of size 8 whose induced subgraph is depicted in Fig. 5.3, where the variable nodes V_2, V_4, V_6, V_7 and V_8 are supposed to be the stopping set pivots. In this figure, the check nodes with degree 2 in the subgraph induced by the stopping set, and connecting the neighboring pivots, have been depicted as filled square nodes, while the other check nodes have been depicted as non-filled square nodes. If $\mathcal{P}^{(0)} = \hat{\mathcal{P}}^{(0)} = \{V_2\}$, then it follows $\mathcal{P}^{(1)} = \{V_2, V_4\}$, $\hat{\mathcal{P}}^{(1)} = \{V_4\}$, $\mathcal{P}^{(2)} = \{V_2, V_4, V_6\}$, $\hat{\mathcal{P}}^{(2)} = \{V_6\}$, and $\mathcal{P}^{(3)} = \{V_2, V_4, V_6\}$, $\hat{\mathcal{P}}^{(3)} = \{\}$. The set of pivots $\mathcal{P}^{(2)}$ is returned by the algorithm. Note that the pivots V_7 and V_8 cannot be found by the algorithm, for $\mathcal{P}^{(0)} = \{V_2\}$.*

This pivot searching algorithm is exploited in the optimization algorithm for LDPC codes on burst erasure channels, presented in the next section. The key for the application of the pivots searching algorithm is the following observation: if for a given LDPC code with maximum guaranteed resolvable burst length $L_{\max}$, a burst of length $L_{\max} + 1$ is non-resolvable, then two pivots of the maximal stopping set included in the burst can be always immediately found.

5.3 Optimization Algorithm for LDPC Codes on Burst Erasure Channels

After having defined the concept of stopping set pivots, in this section we present the LDPC codes optimization algorithm for burst erasure channels.

For an LDPC code with maximum guaranteed resolvable erasure burst length $L_{\max}$, any single erasure burst of length $L \leq L_{\max}$ can be corrected by the iterative decoder independently of the burst position within the codeword of length n. On the contrary, there exists at least one erasure burst of length $L_{\max} + 1$, starting on some variable node V_j, which is non-correctable by the iterative decoder. This implies that this erasure burst includes some stopping sets. Next, it is proved that the maximal stopping set included in the burst (defined as the union of all the stopping sets included in the burst) has at least two pivots. Specifically, the variable nodes V_j and $V_{j+L_{\max}}$, i.e. the first and the last variable nodes in the burst, are pivots.

Theorem 5.2. *Let $L_{\max}$ be the maximum guaranteed resolvable burst length of an LDPC code under iterative decoding, and let the erasure burst of length $L_{\max} + 1$ starting on the variable node V_j and ending on the variable node $V_{j+L_{\max}}$, be non-correctable. Then, V_j and $V_{j+L_{\max}}$ are pivots for the maximal stopping set included in the burst.*

Proof. Let $\mathcal{S}$ be the maximal stopping set included in the non-correctable erasure burst of length $L_{\max} + 1$, and let $\mathcal{G}$ be the subgraph induced by $\mathcal{S}$. By hypothesis, an erasure burst of length $L_{\max}$, starting on the variable node V_j and ending on the variable node $V_{j+L_{\max}-1}$, can be corrected by the iterative decoder. This implies that if the value of the variable node $V_{j+L_{\max}}$ is known, then the iterative decoder applied to $\mathcal{G}$ is able to successfully correct all the variable nodes in the maximal stopping set. Hence, $V_{j+L_{\max}}$ is a pivot of $\mathcal{S}$.

Analogously, an erasure burst of length $L_{\max}$, starting on the variable node V_{j+1} and ending on the variable node $V_{j+L_{\max}}$, can be corrected by the iterative decoder. By reasoning in the same way, it is proved that V_j is a pivot of $\mathcal{S}$. □

The theorem implies that the pivots' span of the maximal stopping set included in the erasure burst of length $L_{\max} + 1$ is equal to $L_{\max} + 1$. By combining Lemma 5.1 with Theorem 5.2 we also obtain the following result on the structure of the maximal stopping set included in a non-correctable erasure burst of length $L_{\max} + 1$.

Corollary 5.1. *Let $L_{\max}$ be the maximum guaranteed resolvable burst length of an LDPC code under iterative decoding. Let the erasure burst of length $L_{\max} + 1$, starting on the variable node V_j, be non-correctable, and let $\mathcal{S}$ be the maximal stopping set included in the*

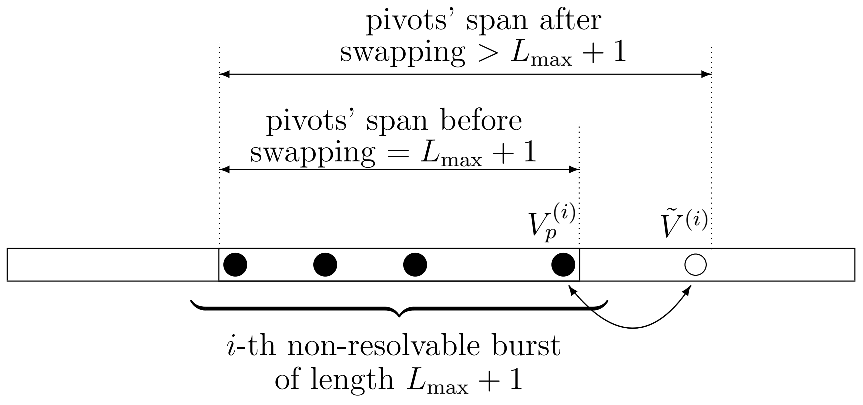

pivots' span after swapping $> L_{\max}+1$
pivots' span before swapping $= L_{\max}+1$
$V_p^{(i)}$
$\tilde{V}^{(i)}$
i-th non-resolvable burst of length $L_{\max}+1$

Figure 5.4: The pivot $V_p^{(i)}$ of the stopping set $\mathcal{S}_{\max}^{(i)}$ is swapped with the variable node $\tilde{V}^{(i)}$ (not in $\mathcal{B}^{(i)}$) in order to make the pivots' span greater than $L_{\max}+1$.

burst. Then, the subgraph $\mathcal{G}$ induced by $\mathcal{S}$ is composed of non-disjoint bipartite graphs, i.e. a path exists from any variable node in $\mathcal{G}$ to any other variable node in $\mathcal{G}$.

The optimization algorithm for LDPC codes on burst erasure channels is described next. It receives an LDPC code parity-check matrix in input, and returns an LDPC code parity-check matrix with improved performance in environments where erasures occur in bursts. This algorithm performs some permutations of the input LDPC code variable nodes, in order to increase its maximum guaranteed resolvable burst length, thus improving its burst erasure correction capability. Neither the input code degree distribution nor the connections between the variable and the check nodes are modified by the algorithm. As a consequence, the input LDPC code and the LDPC code returned by the algorithm have the same degree distribution and the same performance on the memory-less erasure channel.

Consider an LDPC code with maximum guaranteed resolvable burst length $L_{\max}$, let $\mathcal{V}$ denote the ensemble of all its variable nodes, and suppose that the iterative decoder is not able to successfully recover from a number N_B of erasure bursts of length $L_{\max}+1$ (some of the non-correctable erasure bursts might be partly overlapped). Let $V^{(i,f)}$ and $V^{(i,l)}$ be, respectively, the first and the last variable node of the i-th uncorrectable burst, with $i = 1, \dots, N_B$, and let $\mathcal{B}^{(i)}$ be the set of all variable nodes included in the i-th burst. According to Theorem 5.2, the i-th uncorrectable burst contains a maximal stopping set $\mathcal{S}_{\max}^{(i)}$ which includes $V^{(i,f)}$ and $V^{(i,l)}$ among its pivots, and other pivots of this stopping set can be eventually found by the pivot searching algorithm presented in the previous section. Suppose that one of these pivots is swapped with a variable node $\tilde{V}^{(i)}$ not included in $\mathcal{B}^{(i)}$ such that, after the swapping, the pivots' span of $\mathcal{S}_{\max}^{(i)}$ becomes larger than $L_{\max}+1$ (see Fig. 5.4). For any starting erasure pattern given by an erasure burst of length $L_{\max}+1$, the value of at least one pivot of $\mathcal{S}_{\max}^{(i)}$ is now known, and the considered stopping set will be resolvable for any possible position of such burst.

If this procedure is applied to each uncorrectable erasure burst, each maximal stopping set $\mathcal{S}_{\max}^{(i)}$ ($i = 1, \ldots, N_B$) becomes resolvable for any possible burst position. This does not necessarily imply that the erasure burst length $L_{\max} + 1$ will be resolvable at the end of the swapping procedure, since any swap could in principle reduce the pivots' span of some other stopping set. On the other hand, all our numerical results reveal that this approach is indeed very effective up to values of the erasure burst length L extremely close to $\lfloor q^* N \rfloor$.

If the sequence of N_B variable node permutations makes the erasure burst length $L_{\max} + 1$ resolvable for any position of the burst, then the new burst length $L_{\max} + 2$ is considered. On the contrary, a failure is declared, all the permutations are refused, and a new sequence of N_B permutations is performed. The algorithm ends when a maximum number $F_{\max}$ of subsequent failures is reached, for the same burst length. The $L_{\max}$ optimization algorithm is formalized in the following. The algorithm input is an LDPC code with maximum guaranteed resolvable burst length $L_{\max}$.

Pivot Searching and Swapping (PSS) Algorithm

- [*Initialization*] Set $L = L_{\max} + 1$.

- [*Pivot searching step*] Set $F = 0$. Find all the uncorrectable erasure bursts of length L, and let the number of such bursts be N_B. For each $i = 1, \ldots, N_B$, find all the pivots which are neighbors of $V^{(i,f)}$ and all the pivots which are neighbors of $V^{(i,l)}$. Let $\mathcal{P}_i$ be the set of pivots found for the burst i.

- [*Pivot swapping step*] For each $i = 1, \ldots, N_B$:

  1- Randomly choose a pivot $V_p^{(i)}$ in $\mathcal{P}^{(i)}$.

  2- If $V_p^{(i)} \neq V^{(i,f)}$ and $V_p^{(i)} \neq V^{(i,l)}$, then randomly choose a variable node

$$\tilde{V}^{(i)} \in \mathcal{V} / \Big(\mathcal{B}^{(i)} \cup \big(\bigcup_{j \neq i} \mathcal{P}^{(j)} \big) \cup \{ \tilde{V}^{(1)}, \ldots, \tilde{V}^{(i-1)} \} \Big), \tag{5.3}$$

  and swap $V_p^{(i)}$ and $\tilde{V}^{(i)}$.

  Else, if $V_p^{(i)} = V^{(i,f)}$, then randomly choose a variable node $\tilde{V}^{(i)}$ as from (5.3), and such that the index of $\tilde{V}^{(i)}$ is smaller than the index of $V^{(i,f)}$. Swap $V_p^{(i)}$ and $\tilde{V}^{(i)}$.

  Else, if $V_p^{(i)} = V^{(i,l)}$, then randomly choose a variable node $\tilde{V}^{(i)}$ as from (5.3), and such that the index of $\tilde{V}^{(i)}$ is larger than the index of $V^{(i,l)}$. Swap $V_p^{(i)}$ and $\tilde{V}^{(i)}$.

- [*Stopping criterion*] If the erasure burst length is not resolvable, set $F = F + 1$. If $F = F_{\max}$, stop and return the new LDPC code with $L_{\max} = L - 1$. If $F < F_{\max}$ goto *Pivot swapping step.*

  Else, if the erasure burst length L is resolvable, set $L = L + 1$ and goto *Pivot searching step.*

For each non-resolvable burst, the randomly chosen pivot $V_p^{(i)}$ is swapped with a variable node $\tilde{V}^{(i)}$ in order to guarantee that pivots' span of $\mathcal{S}_{\max}^{(i)}$ after the swapping is greater than $L_{\max} + 1$. If $V_p^{(i)} \neq V^{(i,f)}$ and $V_p^{(i)} \neq V^{(i,l)}$, then $\tilde{V}^{(i)}$ is randomly chosen in $\mathcal{V}/\mathcal{B}^{(i)}$, with the exclusion of the available pivots for the maximal stopping sets of the other non-correctable bursts, and of the variable nodes already swapped for the previously considered bursts. If $V_p^{(i)} = V^{(i,f)}$, there are some cases where the pivots' span of $\mathcal{S}_{\max}^{(i)}$ after the swapping might be not greater than $L_{\max}+1$, i.e. when the index of $\tilde{V}^{(i)}$ is larger than, and sufficiently close to, the index of $V^{(i,l)}$. For this reason, if $V^{(i,f)}$ is selected, $\tilde{V}^{(i)}$ is chosen among the variable nodes with index smaller than $V^{(i,f)}$. Analogously, if $V^{(i,l)}$ is selected, $\tilde{V}^{(i)}$ is chosen among the variable nodes with index larger than $V^{(i,l)}$.

The PSS algorithm is extremely flexible and can be in principle applied to any LDPC code, independently of its structure, code rate and codeword length. For instance, it can be applied to either regular or irregular LDPC codes, both computer generated (e.g. IRA [17], eIRA[3] [74][75] or GeIRA [76] codes generated according to the PEG algorithm [77, 78, 16], protograph codes [19, 79]) and algebraically generated (e.g. LDPC codes based on finite geometries [5, 80]). The optimized code returned by the algorithm has the same performance as the input code on the memory-less erasure channel, but is characterized by an increased capability of correcting single erasure bursts. Then, the PSS algorithm can be used within a two-step design approach, consisting in first generating a good LDPC code for the memory-less erasure channel, and then improving it for burst correction. This approach leads to LDPC codes with good performance in environments where the erasures are independent, and where the erasures occur in bursts. The algorithm can be also applied to already implemented LDPC codes: in this case, it can be interpreted as a tool for the design of an *ad hoc* interleaver which will increase the robustness of the code to erasure bursts.

[3]There actually is a difference between IRA codes and eIRA codes, albeit small. In particular, the row weight for the systematic part of the parity-check matrix is constant according to the definition of an IRA code in [17]. Further, according to [17], the systematic part of the parity-check matrix must be a (low-density) generator matrix for an IRA code (meaning that it has more columns than rows). Neither of these contraints are necessary for eIRA codes.

	Margulis	PEG irregular	PEG IRA	PEG GeIRA	PEG regular
(N, K)	$(2640, 1320)$	$(1008, 504)$	$(2000, 1000)$	$(2048, 1024)$	$(4608, 4033)$
R	0.5	0.5	0.5	0.5	0.8752
$N-K$	1320	504	1000	1024	575
$L_{\max}$	1033	86	403	495	287
$\lfloor q^* N \rfloor$	1133	473	861	946	445
$L_{\max}^{\text{PSS}}$	1135	446	852	914	425

Table 5.1: Original and improved values of $L_{\max}$ for the five investigated LDPC codes.

5.4 Numerical Results

In this section, some numerical results on the $L_{\max}$ improvement achievable by applying the PSS algorithm are shown. Five examples are provided. The first four examples are given for LDPC codes with rate $R = 1/2$ and different construction methods; the fifth one for a rate $R = 0.8752$ LDPC code. The four rate-1/2 codes are, respectively, a $(3, 6)$-regular $(2640, 1320)$ LDPC code with Margulis construction [81, 82], an irregular $(1008, 504)$ LDPC code generated with the PEG algorithm, a $(2000, 1000)$ IRA code generated with the PEG algorithm and a $(2048, 1024)$ GeIRA code generated with the PEG algorithm. The IRA and GeIRA codes construction was performed by first generating, respectively, the double-diagonal and multi-diagonal part of the parity-check matrix corresponding to the parity bits, and then generating the systematic part with the PEG algorithm, also considering the 1s already positioned in the parity part. The IRA code is characterized by uniform check node distribution and by a regular systematic part of the parity-check matrix, with all the variable nodes corresponding to the systematic bits having degree 5. The GeIRA code is characterized by feedback polynomial (for the recursive convolutional encoder) $g(D) = 1+D+D^{420}$, and by uniform check node distribution. The degree multiplicity for the variable nodes corresponding to the parity bits is 1(1), 419(2), 604(3), while the degree multiplicity of the variable nodes corresponding to the systematic bits is 885(3), 85(13), 54(14). Finally, the rate 0.8752 code is a $(4, 32)$-regular $(4608, 4033)$ LDPC code generated with the PEG algorithm (the parity-check matrix for this code is (256×4608), with one redundant row). The bipartite graphs of the $(2640, 1320)$ Margulis code and of the $(1008, 504)$ irregular code are both available in [83], where the degree distribution of the irregular code is also specified. The bipartite graph of the $(2000, 1000)$ IRA code, $(2048, 1024)$ GeIRA code and high rate code were generated independently.

The results obtained with the application of the PSS algorithm (with $F_{\max} = N$) to the five codes are summarized in Table 5.1. In this table, $N - K$ represents the maximum possible value for $L_{\max}$, which can be obtained by generating the parity-check matrix in burst correction form and applying to it the quadratic complexity decoding algorithm,

288	289	290	300	322	328	330	334	335	337	340
1	3	3	1	1	1	1	1	1	1	1
344	352	354	356	361	362	365	368	373	374	376
1	1	1	1	2	1	1	1	3	2	8
377	378	379	380	381	382	383	384	385	386	387
2	6	4	5	3	5	9	6	8	6	9
388	389	390	391	392	393	394	395	396	397	398
5	13	6	2	5	9	7	12	3	9	7
399	400	401	402	403	404	405	406	411	412	414
8	12	7	5	18	1	6	1	2	1	3
415	416	418	419	420	421	422	423	424	425	426
3	2	2	2	3	5	4	6	5	4	9

Table 5.2: Number of uncorrectable burst positions and corresponding erasure burst lengths for the $(4608, 4033)$ regular LDPC code .

as explained in [71]. For each code, the value of $L_{\max}$ for the original code, and the estimate of the maximum value of $L_{\max}$ achievable with column permutations ($\lfloor q^* N \rfloor$, as suggested in [63]), are also shown. At the bottom of the table, $L_{\max}^{\text{PSS}}$ denotes the maximum guaranteed resolvable burst length for the code returned by the algorithm.

The starting value of $L_{\max}$ for the Margulis code (1033) was already quite close to $\lfloor q^* N \rfloor$. On the contrary, the values of $L_{\max}$ exhibited by the other codes (86, 403, 495 and 287 respectively) were quite poor with respect to $\lfloor q^* N \rfloor$, especially for the irregular $(1008, 504)$ PEG code. When applied to the Margulis code, the PSS algorithm returned a code with an excellent value of $L_{\max}^{\text{PSS}}$, even larger than $\lfloor q^* N \rfloor$. This result leads to a relevant conclusion: it is possible to construct finite length LDPC codes with moderate codeword length, such that $L_{\max} > \lfloor q^* N \rfloor$. Furthermore, for the irregular PEG code, the IRA code, the GeIRA code and the high rate regular code, the values of $L_{\max}^{\text{PSS}}$ produced by algorithm were quite close to (though lower than) $\lfloor q^* N \rfloor$. These examples reveal the extreme effectiveness of the PSS algorithm for a wide range of LDPC construction methods. As a comparison, in [66, Example 3], a $(4, 32)$-regular $(4608, 4033)$ quasi cyclic LDPC code, whose construction is based on circulant permutation matrices, is proposed for burst erasure correction. This code is characterized by $L_{\max} = 375$. As it can be observed in Table 5.1, the original $(4, 32)$-regular code generated by the PEG algorithm has a value of $L_{\max}$ smaller than 375; however, the PSS algorithm was able to improve this value beyond 375, up to 425.

In order to give a more precise idea about the $L_{\max}$ improvement capability of the PSS algorithm, consider Table 5.2, where the details of how the algorithm worked for the $(4, 32)$-regular code are provided. In each table entry two integer numbers are shown: the integer on the top is a value of erasure burst length, while the integer on the bottom is the

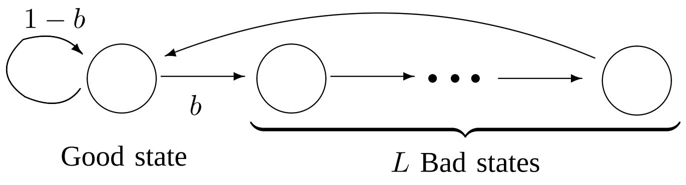


Figure 5.5: Constant length burst erasure channel model.

number of uncorrectable positions registered for that erasure burst length (denoted by N_B in the formalization of the algorithm proposed in the previous section). The burst lengths not shown in the table are those ones for which $N_B = 0$ was registered. Hence, for instance, the first erasure burst length that was recognized as non-resolvable for some burst position, was $L = 288$: for this length, one non-resolvable burst position was found. By applying the pivot searching and swapping principle, a column permutation was obtained which made the length $L = 288$ resolvable for any burst position. Assuming this permuted version of the parity-check matrix, the burst length $L = 289$ was investigated, and three burst positions where recognized as non-resolvable ($N_B = 3$). Again, a column permutation was found that guaranteed the length $L = 289$ to be resolvable for any burst position, and so on up to the burst length $L = 426$, for which the algorithm failed. As it can be observed from Table 5.2, for some values of L the algorithm was able to correct a relatively large number of uncorrectable burst positions, e.g. $N_B = 12$ for $L = 395$ and $L = 400$, $N_B = 13$ for $L = 389$ and $N_B = 18$ for $L = 403$.

We investigated through simulation the performance of the original and of the PSS-optimized $(1008, 504)$ PEG irregular codes, in terms of residual bit erasure rate and decoding failure rate (both evaluated on all the encoded bits) over the constant length burst erasure channel (CLBuEC) depicted in Fig. 5.5. The CLBuEC is described by a Markov chain with a Good state with erasure probability $p_G = 0$, and L Bad states each with erasure probability $p_B = 1$. The channel executes a state transition for each transmitted bit, moving from the Good state to the Bad states with transition probability b, thus generating a burst of erasures of length L. After the generation of the last erasure in the burst, the channel returns in the Good state.

The performance curves of the two codes are shown in Fig. 5.6 and in Fig. 5.7, respectively, for erasure burst length $L = 128$, $L = 256$, $L = 384$. The parity-check matrix of the original code is that given is [83]; the parity-check matrix of the optimized code is given in Appendix C. It is quite evident the performance gain introduced by the optimization performed by the PSS algorithm.

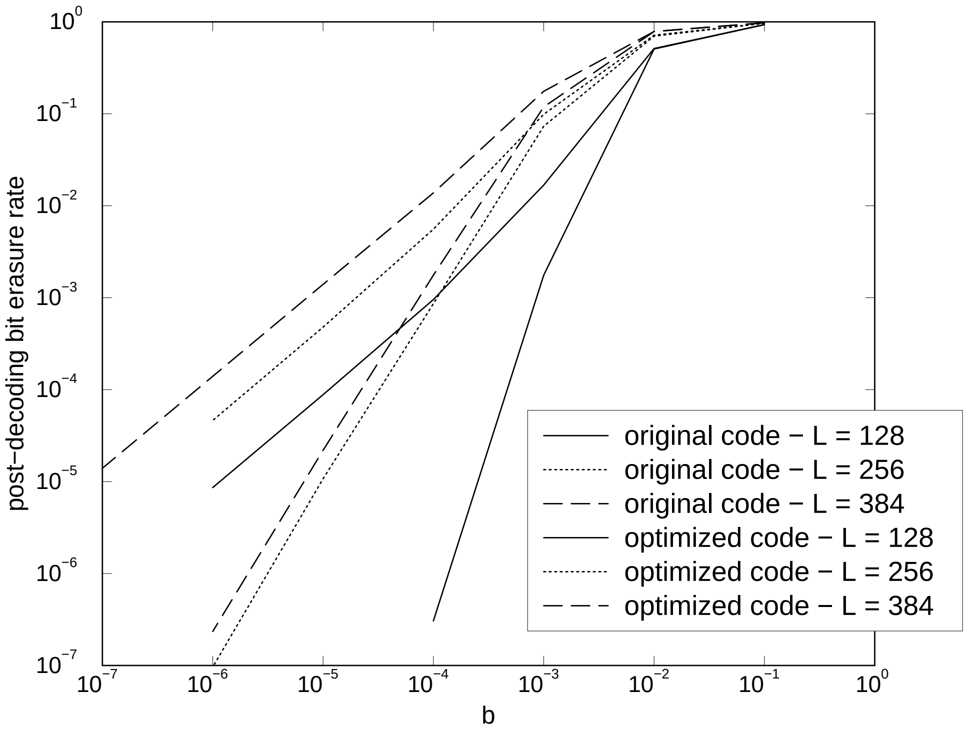


Figure 5.6: Residual bit erasure rate for the (1008, 504) irregular PEG code with $L_{\max} = 86$ and for the PSS-optimized code with the parity-check matrix given in Appendix C (with $L_{\max} = 446$).

5.5 Final Remarks

In this chapter, a simple and effective algorithm for the optimization of LDPC codes on burst erasure channels, under iterative decoding, has been developed. The application of the proposed algorithm to a given LDPC parity-check matrix leads to a new parity-check matrix, characterized by properly permuted columns, with a notable improvement in terms of maximum guaranteed resolvable burst length. At each step of the algorithm, the columns to be permuted are carefully chosen on the basis of a local stopping set pivot analysis for the uncorrectable burst positions. The optimized code has the same performance as the original code on the BEC. Hence, if the input parity-check matrix is optimized in order to achieve good performance on the memory-less erasure channel, then the resulting code can be used for communication both in scenarios with independent erasures and in scenarios where the erasures occur in bursts, by only exploiting the linear complexity iterative decoder. Numerical results have been presented, showing the effectiveness of the proposed approach for a wide range of LDPC code constructions.

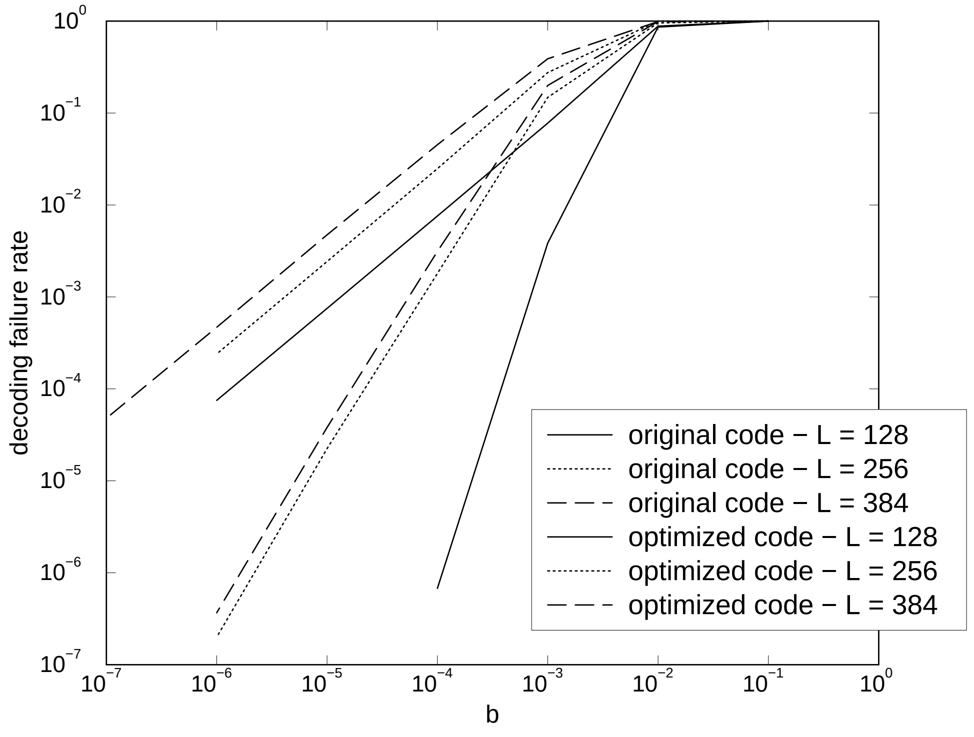


Figure 5.7: Decoding failure rate for the $(1008, 504)$ irregular PEG code with $L_{\max} = 86$ and for the PSS-optimized code with the parity-check matrix given in Appendix C (with $L_{\max} = 446$).

Chapter 6

Conclusion

In this thesis, some issues about asymptotic and finite length design and analysis of binary codes, whose structure is based on sparse bipartite graphs, and whose decoding is performed through iterative algorithms, have been investigated. The coding techniques subject of this thesis, namely LDPC, GLDPC and D-GLDPC codes, have been presented in Chapter 1. In Chapters 2, 3 and 4, asymptotic analysis and design of these codes on the BEC has been considered. More specifically, in Chapter 2, a method for the analysis of GLDPC codes and D-GLDPC codes on the BEC, based on EXIT chart, has been described. This method permits to overcome the problem consisting in the impossibility to evaluate the EXIT function for the check or variable component codes, in situations where the information functions or split information functions for the component code are unknown. The analysis technique has been then combined with DE algorithm in order to generate some optimal GLDPC and D-GLDPC edge distributions. In Chapter 3, the stability condition for LDPC codes on the BEC has been generalized to GLDPC codes and D-GLDPC codes. It is proved that, in both cases, the stability condition only involves the component codes with minimum distance 2. Furthermore, a condition called derivative matching was defined, which is sufficient for a GLDPC or D-GLDPC code to achieve the stability condition with equality. If this condition is satisfied, the threshold of D-GLDPC codes (whose generalized variable nodes have all minimum distance at least 3) and GLDPC codes can be expressed in closed form. In Chapter 4, a family of check-regular LDPC degree distributions, fulfilling the derivative matching condition on the BEC, and are referred to as p-positive distributions, has been analyzed. It was shown that the p-positive distribution have a special property: the threshold optimization problem within the p-positive class can be solved in some cases with analytic methods, without using any numerical optimization tool like DE. It was also shown that these distributions can achieve the BEC capacity, by proving that the binomial degree distributions belong to the p-positive family.

Chapter 5 has then moved towards finite length LDPC codes, and their design over non-memory-less erasure channels. In this chapter, a simple, general-purpose and effective tool for the design of LDPC codes for iterative correction of bursts of erasures, namely the PSS algorithm, has been presented. This algorithm executes permutations of carefully chosen columns of the parity-check matrix, after a local analysis of particular variable nodes called stopping set pivots. This algorithm can be in principle applied to any LDPC code. If the input parity-check matrix is designed for achieving good performance on the memory-less erasure channel, then the code obtained after the application of the PSS algorithm provides good joint correction of independent erasures and single erasure bursts. Numerical results were provided in order to show the effectiveness of the PSS algorithm when applied to different categories of LDPC codes.

Bibliography

[1] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference.* San Francisco, California: Morgan Kaufmann, 1988.

[2] R. Diestel, *Graph Theory (Graduate Texts in Mathematics)*, 3rd ed. Springer-Verlag, 2005.

[3] R. Gallager, *Low-Density Parity-Check Codes.* Cambridge, Massachussets: M.I.T. Press, 1963.

[4] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[5] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, Nov. 2001.

[6] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

[7] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[8] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[9] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[10] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, Oct. 2001.

[11] S. Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," *IEEE Comm. Lett.*, vol. 5, pp. 58–60, Feb. 2001.

[12] K. Price and R. Storn, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, pp. 341–359, 1997.

[13] P. Oswald and M. A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp. 364–373, Dec. 2002.

[14] M. Chiani and A. Ventura, "Design and performance evaluation of some high-rate irregular low-density parity-check codes," in *Proc. of IEEE Global Telecommunications Conf.*, San Antonio, USA, Nov. 2001.

[15] C. Di, R. Urbanke, and T. Richardson, "Weight distribution of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 11, pp. 4839–4855, Nov. 2006.

[16] X. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[17] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proc. of Int. Symp. on Turbo codes and Related Topics*, Brest, France, Sept. 2000.

[18] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inform. Theory*, vol. 50, pp. 1788–1793, Aug. 2004.

[19] D. Divsalar, C. Jones, S. Dolinar, and J. Thorpe, "Protograph based LDPC codes with minimum distance linearly growing with block size," in *Proc. of IEEE Global Telecommunications Conference*, St. Louis, USA, Nov./Dec. 2005.

[20] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, Sept. 1981.

[21] Y. Wang and M. Fossorier, "Doubly-generalized low-density parity-check codes," in *Proc. of IEEE Int. Symp. on Inform. Theory*, Seattle, USA, July 2006.

[22] F. Peng, W. E. Ryan, and R. D. Wesel, "Surrogate-channel design of universal LDPC codes," *IEEE Communications Letters*, vol. 10, no. 6, June 2006.

[23] G. Calzolari, M. Chiani, F. Chiaraluce, R. Garello, and E. Paolini, "New requirements and trends for channel coding in future space missions," *Proc. IEEE*, Jan. 2007, submitted.

[24] H. Xiao and A. H. Banihashemi, "Graph-based message-passing schedules for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 52, no. 12, pp. 2098–2105, Dec. 2004.

[25] H. Song, R. M. Todd, and J. R. Cruz, "Low density parity check codes for magnetic recording channels," *IEEE Trans. Magn.*, vol. 36, no. 5, pp. 2183–2186, Sept. 2000.

[26] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569–584, Feb. 2001.

[27] M. A. Shokrollahi, "New sequences of linear time erasure codes approaching the channel capacity," in *Proc. of Int. Symp. Applied Algebra, Algebraic Algoritmhs, and Error Correcting Codes (Lecture Notes in Computer Science),* M. Fossorier, H. Imai, S. Lin and A. Poli, Eds. Berlin, Germany: Springer-Verlag, 1999, pp. 65–76.

[28] T. Richardson, "Error floors of LDPC codes," in *Proc. of Allerton Conf. on Communication, Control and Computing*, Monticello, USA, Oct. 2003.

[29] A. Amraoui, A. Montanari, and R. Urbanke, "How to find good finite-length codes: From art towards science," *presented at the 4th Int. Symp. on Turbo Codes and Related Topics*, 2006.

[30] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, 2002.

[31] M. Lentmaier and K. Zigangirov, "On generalized low-density parity-check codes based on Hamming component codes," *IEEE Comm. Lett.*, vol. 3, pp. 248–250, Aug. 1999.

[32] J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (Tanner) codes," in *Proc. of IEEE Int. Conf. on Communications*, Vancouver, Canada, June 1999.

[33] J. Chen and R. Tanner, "A hybrid coding scheme for the GilbertElliott channel," *IEEE Trans. Commun.*, vol. 54, no. 10, pp. 564–571, Oct. 2006.

[34] N. Miladinovic and M. Fossorier, "Generalized LDPC codes with Reed-Solomon and BCH codes as component codes for binary channels," in *Proc. of IEEE Global Telecomm. Conf.*, St. Louis, USA, Dec. 2005.

[35] S. Abu-Surra, G. Liva, and W. E. Ryan, "Design and performance of selected classes of Tanner codes," in *UCSD Workshop on Information Theory and its Applications*, San Diego, USA, Feb. 2006.

[36] ——, “Low-Floor Tanner Codes via Hamming-Node or RSCC-Node Doping,” in *16th AAECC (Applied Algebra, Algebraic Algorithms, and Error Correcting Codes) Symposium*, Las Vegas, USA, Feb. 2006.

[37] E. Paolini, M. Fossorier, and M. Chiani, “Analysis of generalized LDPC codes with random component codes for the binary erasure channel,” in *Proc. of Int. Symp. of Information Theory and its Applications*, Seoul, Korea, Nov. 2006.

[38] G. Liva, W. E. Ryan, and M. Chiani, “Quasi-cyclic Generalized LDPC codes with low error floors,” *IEEE Trans. Commun.*, 2007, to be published.

[39] ——, “Design of quasi-cyclic Tanner codes with low error floors,” in *Proc. International Symposium on Turbo codes and Related Topics*, Münich, Germany, Apr. 2006.

[40] G. Yue, L. Ping, and X. Wang, “Generalized low-density parity-check codes based on hadamard constraints,” *IEEE Trans. Inform. Theory*, vol. 53, no. 3, pp. 1058–1079, Mar. 2007.

[41] E. Paolini, M. Fossorier, and M. Chiani, “Analysis of doubly-generalized LDPC codes with random component codes for the binary erasure channel,” in *Proc. of Allerton Conf. on Communications, Control and Computing*, Monticello, USA, Sept. 2006.

[42] ——, “Generalized and doubly-generalized LDPC codes with random component codes for the binary erasure channel,” *IEEE Trans. Inform. Theory*, to be submitted.

[43] S. ten Brink, “Convergence of iterative decoding,” *Electronics Letters*, vol. 35, no. 10, pp. 806–808, May 1999.

[44] A. Ashikhmin, G. Kramer, and S. ten Brink, “Extrinsic information transfer functions: Model and erasure channel properties,” *IEEE Trans. Inform. Theory*, vol. 50, pp. 2657–2673, Nov. 2004.

[45] E. Sharon, A. Ashikhmin, and S. Litsyn, “Analysis of low-density parity-check codes based on EXIT functions,” *IEEE Transactions on Communications*, vol. 54, no. 8, pp. 1407–1414, Aug. 2006.

[46] T. Helleseth, T. Kløve, and V. I. Levenshtein, “On the information function of an error-correcting code,” *IEEE Trans. Inform. Theory*, vol. 43, pp. 549–557, Mar. 1997.

[47] S. Chung, “On the construction of some capacity-approaching coding schemes,” Ph.D. dissertation, Massachusset Institute of Technology, Cambridge, Massachussets, Sept. 2000.

[48] J. Chen and M. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Communications Letters*, vol. 6, no. 5, pp. 208–210, May 2002.

[49] W. Tan and J. R. Cruz, "Analyzing low-density parity-check codes on partial response channels with erasures using density evolution," *IEEE Trans. Magn.*, vol. 40, no. 5, pp. 3411–3418, Sept. 2004.

[50] C. C. Wang, S. R. Kulkarni, and H. V. Poor, "Density evolution for asymmetric memoryless channels," *IEEE Trans. Inform. Theory*, vol. 51, no. 12, pp. 4216–4236, Dec. 2005.

[51] M. A. Shokrollahi and R. Storn, "Design of efficient erasure codes with differential evolution," in *Proc. of IEEE Int. Symp. on Inform. Theory*, Sorrento, Italy, June 2000.

[52] T. Richardson and R. Urbanke, *Modern Coding Theory* (preprint), 2006. [Online]. Available: http://lthcwww.epfl.ch/mct/index.php

[53] E. Paolini and M. Fossorier, "Generalized stability condition for generalized and doubly-generalized LDPC codes," Jan. 2007, submitted for publication.

[54] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of ldpc code ensembles," *IEEE Trans. Inform. Theory*, vol. 51, pp. 929–953, Mar. 2005.

[55] L. Bazzi, T. Richardson, and R. Urbanke, "Exact thresholds and optimal codes for the binary-symmetric channel and Gallager's decoding algorithm A," *IEEE Trans. Inform. Theory*, vol. 50, no. 9, pp. 2010–2021, Sept. 2004.

[56] E. Paolini and M. Chiani, "On the threshold of right regular low-density parity-check codes for the erasure channel," in *Proc. of IEEE Vehicular Technology Conf. Spring*, Stokholm, Sweden, June 2005.

[57] T. Hehn, A. Dönmez, and J. B. Huber, "Exact thresholds for LDPC codes transmitted over binary erasure channels," in *Proc. of Allerton Conf. on Communications, Control and Computing*, Monticello, USA, Sept. 2005.

[58] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, pp. 670–678, Apr. 2004.

[59] E. Paolini and M. Chiani, "A class of LDPC erasure distributions with closed-form threshold expression," in *Proc. of IEEE Int. Conf. on Communications*, Glasgow, Scotland, June 2007.

[60] E. Barbeau, *Polynomials.* Springer Verlag, 2003.

[61] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proc. of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 150–159.

[62] M. Yang and W. Ryan, "Performance of efficiently encodable low-density parity-check codes in noise bursts on the EPR4 channel," *IEEE Trans. Magn.*, vol. 40, no. 2, Mar. 2004.

[63] F. Peng, M. Yang, and W. Ryan, "Simplified eIRA code design and performance analysis for correlated Raileigh fading channels," *IEEE Trans. Wireless Commun.*, vol. 5, no. 4, Apr. 2006.

[64] E. Paolini, M. Chiani, and G. P. Calzolari, "Long erasure correcting codes: the new frontier for zero-loss in space applications?" in *Proc. of Int. Conf. on Space Operations*, Rome, Italy, June 2006.

[65] T. Wadayama, "Ensemble analysis on minimum span of stopping sets," in *Proc. of Inform. Theory and its Applications Workshop*, La Jolla, USA, Feb. 2006.

[66] Y. Y. Tai, L. Zeng, L. Lan, S. Song, S. Lin, and K. Abdel-Ghaffar, "Algebraic construction of quasi-cyclic LDPC codes - Part II: For AWGN and binary random and burst erasure channels," *Applied Algebra, Algebraic Algorithms and Error-Correcing Codes, Lecture Notes in Computer Science*, vol. 3857, Feb. 2006.

[67] Y. Y. Tai, L. Lan, L. Zeng, S. Lin, and K. Abdel-Ghaffar, "Algebraic construction of quasi-cyclic LDPC codes for the AWGN and erasure channels," *IEEE Trans. Commun.*, vol. 54, pp. 1765–1774, Oct. 2006.

[68] D. Divsalar, S. Dolinar, and C. Jones, "Protograph LDPC codes over burst erasure channels," in *Proc. of Military Communications Conference*, Washington, USA, Oct. 2006.

[69] E. Paolini and M. Chiani, "Improved low-density parity-check codes for burst erasure channels," in *Proc. of IEEE Int. Conf. on Communications*, Istanbul, Turkey, June 2006.

[70] G. Hosoya, H. Yagi, T. Matsushima, and S. Hirasawa, "Performance of low-density parity-check codes for burst erasure channels," in *Proc. of Int. Symp. on Inform. Theory and its Applications*, Seoul, Korea, Nov. 2006.

[71] M. Fossorier, "Universal burst error correction," in *Proc. of Int. Symposium on Inform. Theory*, Seattle, USA, Jan. 2006.

[72] H. Pishro-Nik and F. Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 50, pp. 439–454, Mar. 2004.

[73] E. Paolini and M. Chiani, "Construction of near-optimum burst erasure correcting low-dnesity parity-check codes," *IEEE Trans. Commun.*, 2007, submitted.

[74] M. Yang, Y. Li, and W. Ryan, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Trans. Commun.*, vol. 52, pp. 564–571, Apr. 2004.

[75] Y. Zhang, W. Ryan, and Y. Li, "Structured eIRA codes with low floors," in *Proc. of IEEE Int. Symp. on Inform. Theory*, Adelaide, Australia, Sept. 2005.

[76] G. Liva, E. Paolini, and M. Chiani, "Simple reconfigurable low-density parity-check codes," *IEEE Communications Letters*, vol. 9, no. 3, pp. 258–260, Mar. 2005.

[77] H. Xiao and A. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Communications Letters*, vol. 8, no. 12, pp. 715–717, Dec. 2004.

[78] Z. Chen and S. Bates, "Construction of low-density parity-check convolutional codes through progressive edge-growth," *IEEE Communications Letters*, vol. 9, no. 12, pp. 1058–1060, Dec. 2005.

[79] D. Divsalar, S. Dolinar, and C. Jones, "Low-rate LDPC codes with simple protograph structure," in *Proc. of IEEE International Symposium on Information Theory*, Adelaide, Australia, Sept. 2005.

[80] S. Lin, H. Tang, and Y. Kou, "On a class of finite geometry low density parity check codes," in *IEEE International Symposium on Information Theory*, Washington, USA, June 2001.

[81] G. A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, 1982.

[82] D. MacKay and M. Postol, "Weaknesses of Margulis and Ramanujan-Margulis lowdensity parity-check codes," *Electronic Notes in Theoretical Computer Science*, vol. 74, 2003.

[83] D. MacKay, “Encyclopedia of sparse graph codes.” [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

Appendix A

EXIT Functions for $(31, 21)$ Generalized Check Nodes

A.1 $(31, 21)$ Check Node with $d_{\min} = 2$

- Generator matrix (systematic representation):

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0
```

- EXIT function of the $(31, 21)$ $d_{\min} = 2$ code as a check node on the BEC:

$$
\begin{aligned}
I_E(I_A) = \frac{1}{31}\, I_A^6\, (28 &+ 56\, I_A + 99\, I_A^2 + 150\, I_A^3 + 385\, I_A^4 + 672\, I_A^5 + 871\, I_A^6 + 784\, I_A^7 - 570\, I_A^8 \\
&- 6240\, I_A^9 - 16490\, I_A^{10} - 34794\, I_A^{11} - 34637\, I_A^{12} + 45340\, I_A^{13} + 350490\, I_A^{14} \\
&+ 835098\, I_A^{15} + 85836\, I_A^{16} - 7079304\, I_A^{17} - 7440650\, I_A^{18} + 69872946\, I_A^{19} \\
&- 132276186\, I_A^{20} + 126483644\, I_A^{21} - 68247469\, I_A^{22} + 19922550\, I_A^{23} \\
&- 2462578\, I_A^{24})
\end{aligned}
$$

A.2 $(31, 21)$ Check Node with $d_{\min} = 3$

- Generator matrix (systematic representation):

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 1
```

- EXIT function of the $d_{\min} = 3$ code as a check node on the BEC:

$$\begin{aligned}I_E(I_A) = \frac{1}{31}\, I_A^6\, (&14 + 16\, I_A + 99\, I_A^2 + 190\, I_A^3 + 385\, I_A^4 + 864\, I_A^5 + 1183\, I_A^6 + 1218\, I_A^7 - 60\, I_A^8\\ &- 4416\, I_A^9 - 16592\, I_A^{10} - 39942\, I_A^{11} - 73663\, I_A^{12} - 26780\, I_A^{13} + 339654\, I_A^{14}\\ &+ 1471492\, I_A^{15} + 1360887\, I_A^{16} - 9234432\, I_A^{17} - 26205375\, I_A^{18} + 135813782\, I_A^{19}\\ &- 231220494\, I_A^{20} + 210207452\, I_A^{21} - 109990359\, I_A^{22} + 31442850\, I_A^{23}\\ &- 3827942\, I_A^{24})\end{aligned}$$

A.3 EXIT Function for the $(31, 21)$ BCH Check Node

- EXIT function of the $(31, 21)$ BCH code as a check node on the BEC:

$$\begin{aligned}I_E(I_A) = \frac{1}{31} I_A^{11}\, (&3720 + 8432\, I_A^4 - 117180\, I_A^6 - 470580\, I_A^8 + 429660\, I_A^9 + 1783430\, I_A^{10}\\ &+ 8277930\, I_A^{11} - 8559720\, I_A^{12} - 118804650\, I_A^{13} + 400115274\, I_A^{14}\\ &- 599316975\, I_A^{15} + 510710760\, I_A^{16} - 257365575\, I_A^{17} + 71968920\, I_A^{18}\\ &- 8663415\, I_A^{19})\end{aligned}$$

A.4 Expected Information Functions over $\mathcal{G}_*^{(31,21)}$

The expected un-normalized information functions $\mathbb{E}_{\mathcal{G}_*^{(31,21)}}(\tilde{e}_g)$ are given next. The expected EXIT function over $\mathcal{G}_*^{(31,21)}$ can be computed according to these parameters.

$$\begin{aligned}
\tilde{e}_0 &= 0\\
\tilde{e}_1 &= 31\\
\tilde{e}_2 &= \frac{592309685955342566724971891408683046414604485580096933506881080}{636892283181047003977585113353969020718521606746980893309547}\\
\tilde{e}_3 &= \frac{8588487123181173185324131293088604202821491339941690270662594965}{636892283181047003977585113353969020718521606746980893309547}\\
\tilde{e}_4 &= \frac{8015916037838239743184632792257201290156909854315017353367594438 0}{636892283181047003977585113353969020718521606746980893309547}\\
\tilde{e}_5 &= \frac{54107371599521226895209586941489691587192050988153556651079600285 7}{636892283181047003977585113353969020718521606746980893309547}\\
\tilde{e}_6 &= \frac{281357769113596540084587922346888786700106278599873098176303748027 2}{636892283181047003977585113353969020718521606746980893309547}\\
\tilde{e}_7 &= \frac{1172319863187604301745149780684101517191133369319665790737689733332 5}{636892283181047003977585113353969020718521606746980893309547}\\
\tilde{e}_8 &= \frac{4019356647063290821871197785226484750725564740761331375825366828015 0}{636892283181047003977585113353969020718521606746980893309547}
\end{aligned}$$

$$\tilde{e}_9 = \frac{1155551603231353612252625531648778909313850111715143714985752335125 75}{6368922831810470039775851133539690207185216067469808933 09547}$$

$$\tilde{e}_{10} = \frac{217278586160770068978757020258047688710427664145979035412372887204 40}{48991714090849769536737316411843770824501662057460068716119}$$

$$\tilde{e}_{11} = \frac{593147299039452032361390101724201634745846861571830739995403196060365}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{12} = \frac{1078371330736844476386531655687104491556843051296352213393666138720450}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{13} = \frac{131322336700481750208437430118928812713011694606696909138302276947575}{48991714090849769536737316411843770824501662057460068716119}$$

$$\tilde{e}_{14} = \frac{2363204199007316556146268657262786276368077223830814249476925823123500}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{15} = \frac{2868243271771558717628008545433901135639515314857557099986993816947265}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{16} = \frac{3056730058539853596072990587380288605099622013667134459538529215978060}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{17} = \frac{2860879802455801190261784724688780325589006976353574351075448640555275}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{18} = \frac{2348626262296534142275142011175884942145042401557173548753713090060600}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{19} = \frac{1686386363776456599182439145445589096129349438262301544076359017348375}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{20} = \frac{1053955498571955370070323689968743059520967746898811547367642801069040}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{21} = \frac{569368373576094165718663734191557751306298941812915054428206409028455}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{22} = \frac{263820783206035192101409752856689602938118050496066991872455432418925}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{23} = \frac{104343527523236650925519731542102361064256192554028972440340601591875}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{24} = \frac{34977138610506386695220685358539908405275113495844278411074825611125}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{25} = \frac{9821670325423964121544284121179260874564512121193459383899585275161}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{26} = \frac{2269775739862184527285851831187461732760259851942837109920601119857}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{27} = \frac{420620994776339095702712222829569444119979105760024298039720501405}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{28} = \frac{4623712216939972846994280507272787457945298148987936019321288005}{48991714090849769536737316411843770824501662057460068716119}$$

$$\tilde{e}_{29} = \frac{6218963652531595220416862738351052380477293591348986700425556155}{636892283181047003977585113353969020718521606746980893309547}$$

$$\tilde{e}_{30} = 651$$

$$\tilde{e}_{31} = 21$$

Appendix B

Expected Split Information Functions Over $\mathcal{G}_*^{(31,10)}$

For each pair (g, h), the value of $\mathbb{E}_{\mathcal{G}_*^{(n,k)}}[\tilde{e}_{g,h}]$ is shown.

$(0, 0)$	0.000000
$(0, 1)$	10.000000
$(0, 2)$	90.000000
$(0, 3)$	360.000000
$(0, 4)$	840.000000
$(0, 5)$	1260.000000
$(0, 6)$	1260.000000
$(0, 7)$	840.000000
$(0, 8)$	360.000000
$(0, 9)$	90.000000
$(0, 10)$	10.000000
$(1, 0)$	31.000000
$(1, 1)$	619.696970
$(1, 2)$	4180.909091
$(1, 3)$	14854.545455
$(1, 4)$	32454.545455
$(1, 5)$	46635.272727
$(1, 6)$	45169.090909
$(1, 7)$	29298.181818
$(1, 8)$	12207.272727
$(1, 9)$	2945.151515
$(1, 10)$	310.000000

$(2,0)$	929.545460
$(2,1)$	13931.827127
$(2,2)$	83495.694746
$(2,3)$	277802.986569
$(2,4)$	581531.486141
$(2,5)$	809603.803052
$(2,6)$	763439.857130
$(2,7)$	482230.392439
$(2,8)$	194887.070264
$(2,9)$	45339.771631
$(2,10)$	4650.000000
$(3,0)$	13467.433057
$(3,1)$	179317.441605
$(3,2)$	1006449.585126
$(3,3)$	3208664.213253
$(3,4)$	6508607.018652
$(3,5)$	8824698.372438
$(3,6)$	8109830.375383
$(3,7)$	4978791.005197
$(3,8)$	1950251.927063
$(3,9)$	443897.711452
$(3,10)$	44950.000000
$(4,0)$	125522.211766
$(4,1)$	1565284.881525
$(4,2)$	8418703.803610
$(4,3)$	26009974.366845
$(4,4)$	51390645.817435
$(4,5)$	67930160.930020
$(4,6)$	60730222.638597
$(4,7)$	36211885.518441
$(4,8)$	13900267.005602
$(4,9)$	3126911.158904
$(4,10)$	314650.000000
$(5,0)$	845253.865429
$(5,1)$	10100832.042276
$(5,2)$	52649058.013457
$(5,3)$	158452844.393846
$(5,4)$	305272587.758980
$(5,5)$	392722484.407236
$(5,6)$	341324193.126170
$(5,7)$	199590339.030473
$(5,8)$	75753992.769304
$(5,9)$	16938261.839087
$(5,10)$	1699110.000000

$(6, 0)$	4377027.468771
$(6, 1)$	50692322.288212
$(6, 2)$	257399841.476255
$(6, 3)$	755439013.948270
$(6, 4)$	1416766955.160452
$(6, 5)$	1772671029.950008
$(6, 6)$	1511436152.949736
$(6, 7)$	874077445.537898
$(6, 8)$	329790392.823242
$(6, 9)$	73513729.266952
$(6, 10)$	7362810.000000
$(7, 0)$	18104402.514149
$(7, 1)$	204262885.368240
$(7, 2)$	1011477588.621147
$(7, 3)$	2890038143.968905
$(7, 4)$	5272640732.082114
$(7, 5)$	6473140714.757799
$(7, 6)$	5458981280.977422
$(7, 7)$	3138482814.778333
$(7, 8)$	1180561370.808930
$(7, 9)$	262753466.330030
$(7, 10)$	26295750.000000
$(8, 0)$	61278874.870362
$(8, 1)$	674258327.754605
$(8, 2)$	3250657245.841676
$(8, 3)$	9036290308.186577
$(8, 4)$	16177501390.832565
$(8, 5)$	19645275018.586632
$(8, 6)$	16470870944.496225
$(8, 7)$	9440885525.858236
$(8, 8)$	3545803903.398328
$(8, 9)$	788566749.071551
$(8, 10)$	78887250.000000
$(9, 0)$	172310501.718899
$(9, 1)$	1845932393.777827
$(9, 2)$	8658745260.240742
$(9, 3)$	23620651928.758492
$(9, 4)$	41829907984.078880
$(9, 5)$	50501396528.333168
$(9, 6)$	42213762526.091072
$(9, 7)$	24159367759.986790
$(9, 8)$	9066768085.980492
$(9, 9)$	2015617201.155661
$(9, 10)$	201600750.000000

$(10, 0)$	406105259.031647
$(10, 1)$	4232996948.215034
$(10, 2)$	19485756113.129803
$(10, 3)$	52581925597.006966
$(10, 4)$	92577683785.290939
$(10, 5)$	111433993455.688950
$(10, 6)$	93004742327.039581
$(10, 7)$	53186604303.028770
$(10, 8)$	19952685854.441792
$(10, 9)$	4434787593.051601
$(10, 10)$	443521650.000000
$(11, 0)$	808117906.179592
$(11, 1)$	8266504806.757377
$(11, 2)$	37642581226.773872
$(11, 3)$	100989667423.850220
$(11, 4)$	177273408970.402830
$(11, 5)$	213055479762.149380
$(11, 6)$	177683167816.872130
$(11, 7)$	101572437814.320790
$(11, 8)$	38097021033.105110
$(11, 9)$	8466822492.859234
$(11, 10)$	846723150.000000
$(12, 0)$	1377751350.754546
$(12, 1)$	13941539171.830137
$(12, 2)$	63117431868.306053
$(12, 3)$	168828298401.211940
$(12, 4)$	295903729272.282840
$(12, 5)$	355358041286.772280
$(12, 6)$	296245904646.636960
$(12, 7)$	169316034816.289400
$(12, 8)$	63499639864.851669
$(12, 9)$	14111711999.961872
$(12, 10)$	1411205250.000000
$(13, 0)$	2037610380.543325
$(13, 1)$	20499562089.524292
$(13, 2)$	92530075841.779861
$(13, 3)$	247125806334.909060
$(13, 4)$	432802996402.281130
$(13, 5)$	519563677452.488340
$(13, 6)$	433053178012.444400
$(13, 7)$	247482812158.157590
$(13, 8)$	92810528116.392349
$(13, 9)$	20625058924.688747
$(13, 10)$	2062530750.000000

$(14, 0)$	2635658996.247985
$(14, 1)$	26437090892.463089
$(14, 2)$	119149418185.684980
$(14, 3)$	317976053905.264040
$(14, 4)$	556672238878.893070
$(14, 5)$	668135295493.918820
$(14, 6)$	556833025385.680420
$(14, 7)$	318205621708.205020
$(14, 8)$	119329981260.671370
$(14, 9)$	26518092868.001831
$(14, 10)$	2651825250.000000
$(15, 0)$	2996204732.017513
$(15, 1)$	30007962209.370403
$(15, 2)$	135139530917.046190
$(15, 3)$	360510612785.162780
$(15, 4)$	631014901368.111450
$(15, 5)$	757290713546.031620
$(15, 6)$	631105948440.099850
$(15, 7)$	360640644084.295650
$(15, 8)$	135241867918.032530
$(15, 9)$	30053929141.184380
$(15, 10)$	3005401950.000000
$(16, 0)$	3000797257.771355
$(16, 1)$	30030989542.642181
$(16, 2)$	135191339148.062320
$(16, 3)$	360579484782.960880
$(16, 4)$	631074718195.666750
$(16, 5)$	757326042328.804440
$(16, 6)$	631120195604.878300
$(16, 7)$	360644443553.303160
$(16, 8)$	135242478565.066650
$(16, 9)$	30053974374.967899
$(16, 10)$	3005401950.000000
$(17, 0)$	2649794052.065346
$(17, 1)$	26508100059.804852
$(17, 2)$	119309330796.488590
$(17, 3)$	318188737213.128910
$(17, 4)$	556857002955.348750
$(17, 5)$	668244433134.972530
$(17, 6)$	556877040334.643550
$(17, 7)$	318217360072.837100
$(17, 8)$	119331867867.530850
$(17, 9)$	26518232619.768288
$(17, 10)$	2651825250.000000

$(18,0)$	2061741735.977990
$(18,1)$	20621365051.890923
$(18,2)$	92805029214.516251
$(18,3)$	247491929330.167630
$(18,4)$	433121247542.450440
$(18,5)$	519751720145.155210
$(18,6)$	433129026486.351260
$(18,7)$	247503041726.521360
$(18,8)$	92813779562.936020
$(18,9)$	20625299782.430264
$(18,10)$	2062530750.000000
$(19,0)$	1410935895.637917
$(19,1)$	14110706835.014940
$(19,2)$	63501214193.475197
$(19,3)$	169340616236.542110
$(19,4)$	296349618042.984740
$(19,5)$	355621665491.057070
$(19,6)$	296352272855.257140
$(19,7)$	169344408760.863220
$(19,8)$	63504200693.666870
$(19,9)$	14112049866.195944
$(19,10)$	1411205250.000000
$(20,0)$	846642625.851372
$(20,1)$	8466829241.714141
$(20,2)$	38101638405.118507
$(20,3)$	101605578240.330920
$(20,4)$	177810819966.280430
$(20,5)$	213373618796.164000
$(20,6)$	177811613513.982850
$(20,7)$	101606711870.315510
$(20,8)$	38102531122.008568
$(20,9)$	8467230712.741138
$(20,10)$	846723150.000000
$(21,0)$	443500682.090548
$(21,1)$	4435111758.931985
$(21,2)$	19958239039.290668
$(21,3)$	53222285612.595879
$(21,4)$	93139275312.001175
$(21,5)$	111767295669.561040
$(21,6)$	93139481931.197479
$(21,7)$	53222580781.642380
$(21,8)$	19958471482.763126
$(21,9)$	4435216295.019461
$(21,10)$	443521650.000000

$(22, 0)$	201596031.086607
$(22, 1)$	2015983928.055483
$(22, 2)$	9071980816.419312
$(22, 3)$	24192019698.340012
$(22, 4)$	42336096470.266403
$(22, 5)$	50803352963.353203
$(22, 6)$	42336142969.085526
$(22, 7)$	24192086125.088341
$(22, 8)$	9072033127.246248
$(22, 9)$	2016007453.870089
$(22, 10)$	201600750.000000
$(23, 0)$	78886342.244510
$(23, 1)$	788867965.618235
$(23, 2)$	3549916067.560413
$(23, 3)$	9466456476.624792
$(23, 4)$	16566310760.206268
$(23, 5)$	19879580067.927399
$(23, 6)$	16566319704.808643
$(23, 7)$	9466469254.615534
$(23, 8)$	3549926130.206060
$(23, 9)$	788872491.126374
$(23, 10)$	78887250.000000
$(24, 0)$	26295603.119677
$(24, 1)$	262956766.313146
$(24, 2)$	1183307102.431586
$(24, 3)$	3155487811.854520
$(24, 4)$	5522105600.447498
$(24, 5)$	6626527878.359177
$(24, 6)$	5522107047.725414
$(24, 7)$	3155489879.393436
$(24, 8)$	1183308730.616802
$(24, 9)$	262957498.564207
$(24, 10)$	26295750.000000
$(25, 0)$	7362790.480560
$(25, 1)$	73628002.498014
$(25, 2)$	331326231.049636
$(25, 3)$	883536909.210881
$(25, 4)$	1546189847.562929
$(25, 5)$	1855427970.941902
$(25, 6)$	1546190039.895925
$(25, 7)$	883537183.972246
$(25, 8)$	331326447.424111
$(25, 9)$	73628099.809193
$(25, 10)$	7362810.000000

$(26, 0)$	1699107.946839
$(26, 1)$	16991089.744220
$(26, 2)$	76459926.969643
$(26, 3)$	203893169.413272
$(26, 4)$	356813073.447346
$(26, 5)$	428175704.321289
$(26, 6)$	356813093.677939
$(26, 7)$	203893198.314117
$(26, 8)$	76459949.729055
$(26, 9)$	16991099.979930
$(26, 10)$	1699110.000000
$(27, 0)$	314649.839246
$(27, 1)$	3146499.197017
$(27, 2)$	14159248.196824
$(27, 3)$	37757997.605193
$(27, 4)$	66076497.921044
$(27, 5)$	79291798.772426
$(27, 6)$	66076499.505010
$(27, 7)$	37757999.868003
$(27, 8)$	14159249.978786
$(27, 9)$	3146499.998429
$(27, 10)$	314650.000000
$(28, 0)$	44949.991654
$(28, 1)$	449499.958311
$(28, 2)$	2022749.906383
$(28, 3)$	5393999.875666
$(28, 4)$	9439499.892065
$(28, 5)$	11327399.936267
$(28, 6)$	9439499.974301
$(28, 7)$	5393999.993147
$(28, 8)$	2022749.998899
$(28, 9)$	449499.999918
$(28, 10)$	44950.000000
$(29, 0)$	4649.999784
$(29, 1)$	46499.998922
$(29, 2)$	209249.997580
$(29, 3)$	557999.996786
$(29, 4)$	976499.997210
$(29, 5)$	1171799.998353
$(29, 6)$	976499.999336
$(29, 7)$	557999.999823
$(29, 8)$	209249.999972
$(29, 9)$	46499.999998
$(29, 10)$	4650.000000

$(30, 0)$	310.000000
$(30, 1)$	3100.000000
$(30, 2)$	13950.000000
$(30, 3)$	37200.000000
$(30, 4)$	65100.000000
$(30, 5)$	78120.000000
$(30, 6)$	65100.000000
$(30, 7)$	37200.000000
$(30, 8)$	13950.000000
$(30, 9)$	3100.000000
$(30, 10)$	310.000000
$(31, 0)$	10.000000
$(31, 1)$	100.000000
$(31, 2)$	450.000000
$(31, 3)$	1200.000000
$(31, 4)$	2100.000000
$(31, 5)$	2520.000000
$(31, 6)$	2100.000000
$(31, 7)$	1200.000000
$(31, 8)$	450.000000
$(31, 9)$	100.000000
$(31, 10)$	10.000000

Appendix C

Parity-Check Matrix for the $(1008, 504)$ Code with $L_{\max} = 446$

Each line corresponds to a check node. For each check node, the indexes (from 0 to 1007) of the variable nodes connected to that check node are specified.

8	19	41	108	212	508	547	593
183	191	235	282	525	673	866	874
2	89	130	233	345	495	674	894
225	295	364	472	675	962	969	975
174	221	257	293	428	477	676	850
186	385	415	536	677	755	882	1006
6	102	186	235	340	498	678	949
3	247	347	361	541	679	844	987
13	142	473	481	498	590	648	810
7	38	422	452	527	680	843	969
95	125	362	523	681	808	816	874
75	107	263	324	361	682	977	1007
9	107	254	381	515	683	844	850
41	48	219	389	409	438	597	652
145	201	226	523	551	685	900	901
24	59	97	107	259	267	414	686
16	164	184	223	345	409	427	687
119	130	226	362	372	397	688	1002
7	59	89	170	181	507	685	917
223	229	232	447	678	933	959	1001
18	20	156	174	253	377	481	805
225	256	479	496	647	863	875	998
37	107	133	274	481	691	958	1006
275	377	500	547	692	836	936	1006

18	23	24	302	722	844	889	996
82	140	268	336	385	414	694	850
99	461	476	490	580	695	981	992
36	118	346	477	696	971	994	998
73	120	174	219	280	431	541	697
11	77	119	316	461	574	683	851
164	186	302	567	698	928	929	994
29	201	243	347	587	699	843	860
162	287	426	477	489	700	845	878
25	133	284	325	421	701	969	991
34	184	244	310	543	702	856	902
150	223	433	543	703	849	883	998
14	27	89	359	388	482	704	875
194	246	428	429	469	543	705	863
76	291	706	844	846	863	917	957
33	66	89	151	347	701	832	1001
66	341	476	481	707	818	836	999
271	292	308	543	570	708	814	922
58	102	307	388	504	582	642	844
14	174	325	444	503	523	709	919
128	245	294	311	364	597	710	863
313	543	555	711	832	854	858	911
80	224	296	416	567	712	814	846
73	211	276	414	585	595	814	838
68	94	387	506	555	820	830	998
45	130	314	338	515	541	715	870
179	325	434	456	547	716	845	933
58	255	301	417	647	841	939	949
234	246	335	451	572	717	841	848
36	224	317	343	522	523	535	718
49	73	212	224	354	419	719	998
130	173	305	337	422	525	720	944
212	276	458	540	583	721	962	1006
159	184	232	525	558	693	850	916
15	18	153	364	369	388	443	723
18	37	55	405	507	724	837	841
24	205	529	543	559	668	822	934
60	107	118	200	436	509	590	725
96	119	170	283	312	338	726	869
36	119	399	425	499	727	877	912
207	230	570	728	827	912	950	981
60	170	262	315	318	541	729	837
57	70	174	175	498	730	932	969
11	62	151	490	731	922	925	989
41	43	107	423	732	822	845	929

170	184	462	545	730	835	836	970
255	463	465	523	531	549	674	1001
85	246	278	531	580	733	936	989
24	313	321	462	498	734	914	921
127	170	227	235	571	725	809	989
219	255	425	518	680	836	885	911
92	400	441	499	518	735	841	846
41	85	224	568	736	835	841	950
41	55	118	467	489	575	737	908
91	130	153	245	396	738	868	922
52	71	211	454	477	580	933	975
72	79	364	480	739	830	841	1001
30	78	316	329	543	577	740	971
206	283	313	417	525	586	661	863
48	236	306	448	681	896	900	936
85	90	130	179	204	741	836	940
40	41	99	243	300	742	875	952
199	515	590	743	871	879	936	973
258	400	494	515	555	744	933	983
121	201	252	269	412	506	745	922
52	81	364	393	462	743	956	981
146	186	414	557	746	886	902	989
151	219	457	525	747	815	837	878
84	191	203	347	395	530	741	826
33	101	364	397	424	523	748	898
118	288	313	327	400	418	749	819
119	184	215	335	388	391	459	750
88	89	107	721	814	851	995	997
128	130	498	591	751	837	960	987
84	164	188	244	293	499	582	737
99	119	267	483	752	852	998	1007
476	753	848	859	903	966	974	976
75	89	147	339	754	848	863	870
23	52	325	398	428	522	755	1000
37	52	252	348	362	756	875	984
93	118	196	233	287	420	476	757
125	241	313	361	758	829	969	979
233	239	308	398	523	590	733	972
316	432	525	555	565	759	884	962
246	261	263	333	351	369	498	677
244	325	352	514	591	728	838	900
118	332	336	429	760	837	840	981
118	198	205	276	465	761	844	997
76	93	130	137	298	805	843	949
7	90	153	404	533	762	837	976

98	101	244	264	476	763	931	1001
68	99	218	235	428	573	764	970
144	266	555	698	876	900	953	1001
118	141	237	245	316	320	765	913
279	323	388	499	503	507	543	766
96	121	259	276	374	376	691	933
25	48	99	350	395	479	767	966
61	76	152	457	522	558	768	845
125	148	212	290	534	596	749	827
18	105	187	517	555	754	827	949
1	235	264	411	554	588	902	925
106	141	149	200	770	828	935	1001
85	125	219	229	501	763	844	990
7	235	344	359	560	771	825	863
109	164	170	491	508	510	744	935
11	94	201	202	545	547	550	844
19	25	52	111	275	459	464	772
25	112	327	388	470	569	694	836
83	99	113	157	184	325	738	891
47	114	173	300	311	325	773	833
43	76	365	590	730	850	880	965
18	47	116	196	316	409	474	746
246	304	525	554	691	713	846	899
363	371	543	774	889	891	935	936
8	73	107	138	168	255	347	657
212	233	313	599	775	830	847	862
7	232	282	316	380	589	600	776
7	25	121	356	601	768	888	996
80	239	255	276	282	375	775	987
313	326	523	538	602	777	848	991
25	186	477	516	583	603	778	904
75	170	276	349	393	499	779	858
52	123	338	385	557	604	705	830
125	378	382	554	579	780	851	935
78	126	379	397	605	720	933	976
54	78	173	224	431	439	507	678
92	362	606	707	828	893	965	989
24	54	246	411	464	597	646	983
35	129	519	543	607	733	812	837
42	136	364	451	541	608	727	850
19	156	326	484	609	707	863	943
7	47	203	510	781	849	904	1004
74	78	186	444	505	553	813	998
206	411	445	481	562	610	652	843
7	173	202	223	256	576	611	774

18	60	134	282	494	520	669	900	
177	223	241	276	783	890	935	984	
135	145	244	255	477	784	806	849	
231	235	388	612	683	831	915	1001	
18	118	427	450	613	763	888	1006	
53	212	245	277	390	414	785	921	
139	173	327	424	581	652	850	913	
87	165	295	320	399	411	786	814	
35	52	82	144	222	392	787	827	989
7	50	142	174	537	732	828	883	
47	138	143	247	295	525	772	965	
256	257	411	614	772	815	900	981	
51	186	316	394	467	615	788	824	
25	36	245	555	556	593	670	757	
18	392	449	616	789	863	934	940	
62	141	263	364	477	546	553	685	
24	362	384	386	400	611	790	930	
166	174	555	617	762	816	846	981	
223	252	255	498	616	759	920	1005	
58	282	327	460	618	780	954	1006	
22	41	337	498	645	897	972	976	
7	60	244	364	401	767	842	1000	
47	175	199	362	373	402	619	693	
41	57	296	370	477	523	620	783	
48	337	362	614	712	850	924	1004	
121	154	156	164	403	522	595	771	
174	199	244	572	779	834	909	959	
155	163	423	615	706	900	969	989	
56	121	173	242	467	480	787	931	
81	121	122	338	621	704	836	930	
58	259	436	491	676	845	930	941	
124	219	590	602	791	875	882	924	
162	177	218	244	498	586	787	872	
47	233	437	473	567	622	716	997	
173	442	446	494	740	911	976	989	
52	209	224	266	566	590	760	1007	
78	121	161	336	364	407	620	792	
107	177	205	522	530	793	936	942	
36	223	265	270	373	400	536	794	
76	97	212	524	609	795	935	938	
68	212	254	336	434	476	513	796	
58	316	328	526	587	797	863	972	
25	98	141	223	224	455	623	798	
338	477	555	596	606	799	883	960	
5	219	313	317	424	491	796	891	

27	162	184	411	519	523	781	948	
58	114	313	357	481	624	735	823	
19	35	103	158	174	223	625	767	902
110	173	200	450	764	875	997		
77	411	626	793	818	841	933	978	
24	195	233	284	295	366	431	800	
125	153	414	627	750	820	828	833	
51	164	263	338	365	710	906	911	
76	236	474	626	745	814	853	1006	
169	173	541	591	628	768	969	988	
52	285	408	582	600	748	848	998	
79	302	304	310	467	773	845	848	
196	203	297	400	801	828	858	905	
61	130	146	499	623	710	961	976	
24	77	199	478	532	629	709	855	
30	246	384	385	481	571	630	671	
118	222	322	354	364	385	442	778	
65	152	170	295	682	808	848	986	
16	99	187	313	382	578	797	989	
0	124	152	174	353	385	554	759	
46	201	316	451	610	768	828	967	
176	201	255	358	529	631	748	850	
25	75	76	148	176	244	271	669	
45	323	411	516	522	711	976	992	
65	224	263	478	570	632	780	861	
171	233	244	291	489	522	633	699	
41	123	125	232	254	433	481	680	
186	295	424	453	563	634	802	826	
311	368	388	411	567	756	934	967	
76	180	203	219	251	534	567	731	
24	89	418	613	702	824	947	981	
184	286	300	332	349	659	848	969	
182	325	347	373	635	666	827	978	
39	83	170	174	245	374	587	784	
139	234	235	271	463	788	875	969	
78	160	245	298	708	837	901	911	
59	255	424	440	801	863	881	975	
68	492	556	636	803	824	912	922	
185	345	411	590	637	778	845	943	
18	53	64	99	112	590	804	993	
91	201	394	476	495	591	781	845	
233	245	314	478	638	747	814	881	
41	119	191	258	265	514	779	814	
47	82	122	155	235	541	597	695	
4	11	173	355	400	799	826	956	

24	325	335	802	837	864	880	992	
28	199	282	331	404	476	800	823	
35	101	108	199	333	339	900	954	
44	63	252	354	499	525	765	936	
99	165	424	522	639	760	930	963	
66	466	554	562	802	895	981	1006	
233	499	540	568	625	796	826	828	
190	199	286	359	414	605	791	912	
132	179	244	345	424	706	819	843	
172	199	219	295	559	640	689	843	
30	158	305	555	619	686	875	1006	
47	48	159	192	280	806	843	935	
245	361	546	640	717	825	933		
35	71	115	288	522	567	807	848	
173	245	347	532	705	714	932	940	
7	11	41	366	631	714	726	894	
11	177	295	477	617	687	887	926	
35	212	364	411	535	708	821	899	940
60	201	330	595	688	907	912	1006	
41	164	236	347	483	585	623	789	
56	99	102	245	260	338	524	728	
119	448	547	555	590	735	839	974	
50	164	173	197	551	641	775	912	
104	236	255	470	642	799	845	875	
77	108	263	353	362	541	643	937	
239	478	541	542	554	692	961	971	
201	362	524	644	701	814	892	964	
31	99	385	459	564	645	766	827	
164	463	471	512	545	644	845	900	
11	48	222	299	611	751	865	933	
89	182	303	400	423	808	868	976	
173	231	262	282	345	385	466	665	
276	467	809	823	856	870	955	969	
135	541	567	588	646	836	841	910	
28	89	315	523	525	582	616	717	
64	187	223	391	522	740	806	875	
119	174	209	355	562	635	716	863	
48	160	283	501	522	539	810	814	
10	270	364	463	554	690	715	848	
7	186	208	478	502	627	700	965	
6	19	239	574	639	785	850	934	
9	125	242	365	755	843	936	998	
78	129	149	362	525	646	752	826	
118	224	246	290	422	426	679	968	
11	85	107	180	313	757	879	895	

57	113	121	260	411	414	534	809	
19	130	172	309	333	590	617	693	
78	319	337	400	743	847	889	922	
68	223	382	440	648	751	843	877	
63	163	420	462	522	541	671	828	
58	121	124	367	478	552	569	711	
11	70	76	104	226	382	476	884	
58	210	262	342	476	739	883	912	
35	551	592	618	789	857	933	981	
29	121	342	498	608	804	826	935	
41	153	223	649	754	836	942	980	
18	38	106	266	295	649	788	936	
219	414	451	636	664	946	981	1003	
78	333	467	472	650	784	844	893	
263	509	526	628	719	736	836	933	
58	78	302	378	603	699	827	926	
76	110	366	523	537	673	837	879	
119	203	213	244	253	478	628	803	
18	52	61	89	193	651	750	951	
52	233	261	298	316	373	804	945	
219	362	527	543	544	684	782	810	
58	246	347	371	419	612	776	931	
300	498	554	569	598	795	844	865	
78	88	156	255	396	554	634	787	
138	181	380	541	734	863	909	922	
116	164	282	283	390	651	719	922	
52	124	188	216	498	653	792	843	
35	325	454	462	478	654	703	869	
121	346	385	406	631	725	844	992	
49	107	121	217	235	655	677	999	
164	259	316	511	549	670	953	969	
99	125	127	437	806	837	873	919	
189	245	263	510	684	696	887	1006	
48	174	186	241	249	512	798	992	
198	251	338	388	587	655	789	989	
31	224	313	388	397	514	689	1003	
11	276	397	412	517	676	837	957	
133	357	420	424	554	565	709	922	
17	78	107	184	421	548	785	940	
130	316	366	497	573	640	778	989	
24	224	316	330	410	527	656	758	
24	35	385	451	548	590	638	662	952
184	199	528	630	798	845	885	999	
62	94	170	178	191	220	255	467	
196	276	334	388	561	720	910	998	

11	117	246	250	325	657	745	823	
24	25	87	220	432	635	712	875	
99	167	400	469	731	949	988	1006	
197	203	481	524	589	718	886	981	
58	99	283	285	310	497	675	934	
100	130	224	601	804	823	827	982	
19	194	347	364	439	562	658	781	
3	76	601	726	900	976	995	999	
73	125	130	430	478	504	511	620	
48	86	290	372	637	785	969	998	
364	400	459	481	493	602	698	908	
107	300	615	724	827	862	863	898	
228	341	464	478	722	912	935	989	
19	356	496	551	598	729	828	922	
107	154	204	302	476	555	644	770	
32	164	373	414	592	675	841	918	
10	199	309	387	591	786	934	976	
4	17	313	554	597	619	729	1001	
7	109	481	499	654	800	945	997	
214	219	236	325	475	495	792	934	
19	373	476	594	633	766	920	936	
1	282	407	452	826	836	844	884	
25	170	333	435	563	659	736	981	
78	395	414	438	703	829	863	931	
255	325	337	360	388	627	794	927	
47	252	350	621	764	848	933	938	
57	132	447	522	543	650	803	850	
25	74	276	329	347	643	738	975	
118	162	487	521	648	766	934	1001	
107	212	316	326	622	812	980	1002	
233	263	327	360	654	813	837	928	
47	238	411	499	660	689	770	985	
0	35	96	282	400	578	786	935	966
99	478	661	776	817	848	897	900	
161	184	224	347	366	387	801	817	
86	263	269	453	529	875	888	935	
119	460	522	554	614	746	834	915	
89	121	137	199	399	783	827	857	
35	76	383	499	658	817	888	930	
201	263	288	295	363	402	805	852	
47	150	207	477	529	590	599	807	
19	177	211	303	334	386	481	837	
30	48	338	554	662	722	923	979	
120	256	338	498	633	673	841	890	
12	52	125	203	289	630	800	852	

67	124	238	388	416	477	813	922	
100	477	499	542	622	792	931	969	
184	369	379	424	481	505	583	811	
125	141	312	388	637	697	955	1006	
385	423	663	777	814	821	933	968	
140	142	151	184	186	201	560	744	
21	35	121	362	424	690	692	823	985
13	77	215	282	424	625	674	998	
186	203	421	541	783	906	915	941	
12	245	415	525	567	600	808		
68	567	657	790	864	866	935	976	
61	170	203	340	477	605	761	903	
61	212	240	520	634	686	843	848	
203	227	243	326	525	794	814	937	
125	235	259	352	444	486	687	814	
170	441	475	525	664	771	891	998	
25	192	443	489	543	660	695	841	
26	35	250	665	875	884	891	934	
19	126	467	475	555	807	886	999	
348	468	477	607	734	843	902	912	
48	196	282	307	666	764	860	934	
58	482	590	679	828	892	913	975	
235	263	385	551	593	758	840	951	
21	224	405	414	518	555	888	909	
186	246	272	366	639	770	836	867	
35	42	164	256	599	762	769	1006	
20	68	246	282	408	524	656	806	
87	127	184	294	647	936	946	1001	
19	178	195	338	527	664	780	814	
297	381	385	656	790	845	922	966	
235	358	492	603	777	836	845	852	
183	223	245	561	610	739	827	919	
19	203	382	523	621	713	798	927	
67	219	485	667	802	827	930	1001	
18	263	281	313	401	486	650	756	
2	400	467	567	661	795	850	867	
15	47	95	276	428	476	638	749	
219	281	331	641	662	948	989	998	
89	103	119	228	290	416	797	936	
25	199	239	306	410	413	481	584	
58	159	273	290	299	554	791	900	
316	377	400	505	529	912	964	986	
119	169	250	323	326	767	827	845	
413	414	498	562	668	774	845	855	
57	255	263	583	624	867	923	981	

58	223	240	362	528	723	803	929
125	295	577	663	807	841	872	909
5	52	53	201	246	248	496	765
131	212	295	567	629	799	981	990
125	186	499	524	668	747	958	993
199	463	641	737	841	905	916	1006
52	68	131	171	301	527	802	900
78	89	487	488	499	667	807	
68	193	295	478	480	505	782	850
1	263	276	413	488	800	934	965
212	222	246	581	584	666	803	1001
414	552	566	587	684	772	976	1001
11	30	435	541	575	771	827	947
105	118	373	424	742	839	876	935
87	218	351	467	724	922	976	977
24	26	203	424	663	689	919	974
174	185	282	801	824	848	896	950
35	48	83	89	191	321	694	944
424	469	478	533	777	790	918	936
68	289	292	414	489	786	936	956
75	484	498	523	564	567	796	878
7	162	467	742	814	861	932	973
22	244	567	626	727	811	875	989
214	222	279	338	347	467	682	921
203	319	344	430	793	843	850	1001
47	131	164	213	313	344	507	753
48	111	166	276	343	366	799	922
48	124	136	295	500	549	555	809
118	119	370	404	406	609	843	930
190	423	458	476	525	607	715	828
233	375	388	467	539	659	808	843
68	162	306	538	813	844	969	982
76	318	467	478	481	536	588	963
201	274	325	383	567	632	791	936
130	295	325	510	636	775	831	993
167	376	411	624	779	812	844	912
130	133	201	249	268	645	702	998
121	359	485	502	697	848	934	1000
41	277	524	669	769	850	884	989
27	58	170	173	217	269	336	544
44	69	224	510	642	773	837	912
48	184	278	361	367	604	794	912
18	347	476	570	576	696	914	959
39	248	523	541	690	793	852	934
68	134	186	255	459	618	704	859

34	221	276	338	753	804	828	868
233	266	293	362	521	583	612	976
47	343	435	478	812	836	875	1005
11	40	216	385	501	765	900	919
19	235	272	774	841	854	883	957
53	173	233	471	579	667	812	841
46	53	170	362	385	456	655	801
68	117	244	575	590	795	873	915
19	219	332	445	487	809	900	966
47	115	119	237	582	658	732	922
210	359	543	550	649	718	827	976
76	266	468	554	608	700	907	933
199	368	660	769	810	824	828	933
147	199	212	449	522	606	776	813
68	89	118	232	403	588	853	899
203	334	411	810	842	876	909	969
100	281	467	515	543	594	651	828
151	168	230	653	723	828	836	934
18	223	235	328	513	643	773	909
12	162	233	338	389	511	689	912
157	244	417	493	570	688	844	935
11	72	87	203	245	281	629	985
143	174	212	281	322	400	632	741
25	67	201	282	671	672	761	830
41	76	324	613	672	784	823	912
7	24	273	338	360	665	752	909
11	68	133	446	604	681	934	939
69	78	124	164	246	455	721	874
32	208	653	797	915	935	981	
32	156	189	347	424	499	788	871

Acknowledgment

During my doctoral course period I have been constantly encouraged, supported and advised by several people: had it not been for them, I would have never made this thesis in shape. It turns to be a pleasant part that I have now the opportunity to express my gratitude to all of them.

I am deeply indebted to my supervisor, Prof. Marco Chiani, for offering me the study chance and the research environment, for supporting me, for continuously stimulating me with his instructions, and for allowing me to spend a research period in the United States, under the supervision of Prof. Marc Fossorier. I am grateful to Prof. Marc Fossorier for hosting me in his laboratory at the University of Hawai'i at Manoa and for always supporting me. I deeply appreciate his insightful ideas and his many hours spent with me discussing my research.

I wish to dedicate a special thank to my friends and former officemates Matteo Mazzotti, Gianluigi Liva and Andrea Giorgetti (we had a great time together) and to Prof. Davide Dardari for never denying me his precious suggestions any time I needed them. I also wish to thank all my labmates, colleagues and friends in Cesena, especially Enrico Maria Vitucci, Mirko Viroli, Matteo Lucchi, Stefano Severi, Paola Pulini, Alessandro Ricci, and all the people carrying out every day their working activity at the II Engineering Faculty of the University of Bologna. I also wish to thank all the friends I have in Fano, my town (and here the list would be really too long).

I will never forget the kindness of people living in Hawai'i, especially Bruse Eckmann (without his daily help I'd never had such a great time in Hawai'i), Louise Marie Giuseffi, Eric Moennich, and all the people and Ph.D. students (Yige Wang, Wenyi Jin, Zigui Yang) I have met at the EE Department of the UH.

Finally, I would like to express boundless thanks to my wife Manuela, whose love and infinite patience enabled this work, to my parents Fausto and Carla, my brother Fabio and my sisters Letizia and Silvia, for their encouragement in getting a doctoral degree.

Vedi, in questi silenzi in cui le cose
s'abbandonano e sembrano vicine
a tradire il loro ultimo segreto,
talora ci si aspetta
di scoprire uno sbaglio di Natura,
il punto morto del mondo, l'anello che non tiene,
il filo da disbrogliare che finalmente ci metta
nel mezzo di una verità.
Lo sguardo fruga d'intorno,
la mente indaga accorda disunisce
nel profumo che dilaga
quando il giorno piú languisce.
Sono i silenzi in cui si vede
in ogni ombra umana che si allontana
qualche disturbata Divinità.

E. Montale, *I limoni*