UNIVERSITÀ DI PARMA

DEPARTMENT OF ENGINEERING AND ARCHITECTURE

MASTER OF SCIENCE IN COMMUNICATION ENGINEERING
(LAUREA MAGISTRALE)

ENHANCED DECODERS FOR QUANTUM ERROR CORRECTION

Advisor:

Giulio Colavolpe

Co-Advisors:

Balazs Matuz, Francisco Lázaro Blasco

Master Thesis of:

Stefano Tinelli

ACADEMIC YEAR 2021/2022

Table of Contents

Acronyms **v**

Introduction **1**

1 Quantum error correction **3**
1.1 Preliminaries . . . 3
1.2 Quantum Error Correction . . . 5
1.3 Stabilizer Codes . . . 7
1.4 Binary/4-ary representation of quantum codes . . . 8
1.5 CSS codes . . . 10
1.6 QLDPC codes . . . 11
1.7 Measurement and Quantum Channel . . . 12
1.7.1 Depolarizing Channel . . . 12
1.8 Degeneracy in Quantum Codes . . . 13
1.9 Optimal decoding rule for QEC codes . . . 13

2 Classical Decoding Algorithms **15**
2.1 Belief Propagation . . . 15
2.1.1 Belief Propagation Syndrome Decoding . . . 16
2.2 Information Set Decoding with Bit-Flipping . . . 18
2.2.1 Information Set Decoding of Quantum Codes . . . 19
2.2.2 Reprocessing . . . 22
2.2.3 Smart Bit-Flipping . . . 23
2.3 Classical Ordered Statistics Decoding . . . 24
2.4 Quantum Codes and Ordered statistics . . . 25

3 Enhanced Decoding Algorithms **29**
3.1 Non-Binary Belief Propagation . . . 29
3.2 Non-Binary Belief Propagation with Ordered Statistics over $\mathbb{F}_2$ . . . 33
3.2.1 Ordering with $\mathbb{F}_4$ reliability . . . 34
3.2.2 Algorithm Flow-Chart . . . 35
3.3 Non-Binary Belief Propagation and Information Set Decoding over $\mathbb{F}_2$ . . . 36

3.4 Correction of Y-Errors 38
3.4.1 Symbol Flipping 39
3.4.2 Modified Gaussian Elimination 41

4 Numerical Results **43**
4.1 Toric Codes 43
4.2 Hyper-Graph Product Codes 47
4.3 Single Parity-Check Product Code 48

Conclusions **51**

Appendix **53**

Bibliography **55**

Acronyms

AI artificial intelligence

APP a posteriori probability

BP belief propagation

BF bit flipping

BSC binary-symmetric channel

BM benchmark

CD coset decoding

CN check node

CSS Calderbank-Shor-Steane

GE gaussian elimination

HPC hyper-graph product code

ISD information-set decoding

LDPC low-density parity-check

LLR log-likelihood ratio

LR likelihood ratio

MAP maximum *a posteriori*

ML maximum likelihood

MRP most-reliable positions

OSD order-statistics decoding

PCM parity-check matrix

QEC quantum error correction

QLDPC quantum low-density parity-check

SF symbol flipping

SPCP single parity-check product

VN variable node

Introduction

In the last years, governments and industry giants became more and more interested in quantum computing, owing to the capability of quantum computers to find solutions to problems a classical computer might never hope to solve. Solutions may be found thanks to two characteristics of quantum mechanics: superposition and entanglement. The first is the feature of the quantum information not relying on ones and zeros (bit) but on their superposition, taking characteristics of both (qubit). The second property ensures that two quantum particles stay connected, regardless of the distance and what affects one particle does affects also the other. The more qubits entangled within a quantum computer, the greater its computational power can grow. These made quantum information a central resource used in quantum processing, enabling powerful new ways to compute and making it suitable for studies involving an enormous quantity of data. Several applications may gain from this innovation in terms of computing capability, from Industrial design to logistics, to finance and even artificial intelligence (AI) [1]. Unfortunately, even if quantum computing seems to be one of the most promising technologies there are many challenges along the route, such as the susceptibility to errors making quantum computation difficult to scale. Although a simple quantum processor performing a few operations on a handful of qubits might be possible, this is not enough to exploit the full potential of quantum computing. quantum error correction (QEC) may represent the solution to the failure imposed by quantum mechanics, but differently to classical coding, the environment in which quantum information exists is subject to severe constraints. Firstly, the measurement because observing directly a qubit to check for errors damages it by changing it forever without a chance to use its information. Secondly, the no-cloning theorem posits the impossibility of making a perfect copy of an unknown quantum state. Hence, to obtain a realistic copy of a state in a given time we have to store the entire process up to the moment we want to copy it [2]. The literature has shown how those two obstacles may be overcome to produce effective QEC codes, the first example of which was Shor's code [3]. The aim of this thesis is to find good decoders suitable for those codes following a heuristic approach. In the thesis, multiple decoders are proposed and tested using several well-known QEC codes, highlighting their advantages and disadvantages.

Chapter 1

Quantum error correction

In this chapter, a brief review of quantum error correction is performed starting from its definitions. Furthermore, the channel together with some of the codes, both adopted for the simulations in this thesis, are presented.

1.1 Preliminaries

In classical communication, the elementary unit that carries information is a bit, which may take values of 0 or 1. The quantum analog to the classical bit is called a qubit. A qubit can be in the superposition of two information states, zero and one, and hence present characteristics of both of them [4]. Analytically, the quantum state of the qubit $|\psi\rangle$ may be represented as the weighted superposition of the state $|0\rangle$ and state $|1\rangle$ which form the basis of the two-dimensional Hilbert space. We have

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \tag{1.1}$$

where α_0 and α_1 are complex numbers satisfying the normalization condition according to

$$|\alpha_0|^2 + |\alpha_1|^2 = 1. \tag{1.2}$$

Let us now consider a system with K qubits. If a qubit is represented as the superposition of states zero and one (see (1.1)), a quantum state of K qubits is represented by the superposition of the 2^K quantum states which are the basis of the K-dimensional Hilbert space. The base vectors can be indexed by a string of length K, and we obtain

$$\sum_s \alpha_s|s\rangle \tag{1.3}$$

As per the single qubit, the weighting factors α_s take 2^K possible values which are constrained to normalization, according to the equation

$$\sum_s |\alpha_s|^2 = 1. \tag{1.4}$$

The general expression of a two qubits state is reported below with the normalization constraint over the weighting factor:

$$\begin{aligned} &\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \\ &|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1. \end{aligned} \tag{1.5}$$

The unitary transformation of a quantum state vector, e.g., induced by the rotation of the spin of a particle, may be described by the *Pauli matrices*, in Box 1.1. The operators can be cast in four types respectively named: I,X, Y and Z. The operator I represents the identity matrix and does not apply any perturbation to the qubit.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Box 1.1: Pauli operators

Differently from I, the other operators affect the qubit by changing its state. In Box 1.2 we exemplify the effect of these operators applied to a generic single qubit state.

$$|0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad X(\alpha_0|0\rangle + \alpha_1|1\rangle) = \alpha_0|1\rangle + \alpha_1|0\rangle$$

$$Y(\alpha_0|0\rangle + \alpha_1|1\rangle) = i(\alpha_0|1\rangle - \alpha_1|0\rangle)$$

$$|1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad Z(\alpha_0|0\rangle + \alpha_1|1\rangle) = \alpha_0|0\rangle - \alpha_1|1\rangle$$

Box 1.2: actions of the Pauli operators

The operators X, Z and Y may be seen as a bit flip, a phase flip and a combination of the two (ignoring the phase factor i), respectively. By multiplying two different Pauli operators, it is possible to construct the remaining one (this feature will be exploited in the construction of the binary representation of the quantum codes in Section 1.4):

$$\begin{aligned} XY &= iZ & YX &= iZ \\ YZ &= iX & ZY &= iX \\ XZ &= iY & ZX &= iY \end{aligned} \tag{1.6}$$

The Pauli operators affecting a state composed of N qubits are described as a sequence of Pauli operators $M_i \in \{I, X, Z, Y\}$ with $i \in \{1, \ldots, N\}$ weighted by a global phase factor $c \in \{i, -i, 1, -1\}$ which is usually dropped since the global phase does not influence the measurement outcomes. The Pauli operators applied on N qubits $|i_1 i_2 \ldots i_N\rangle$ can be formally written as:

$$M_1 M_2 \cdots M_N \ |i_1 i_2 \ldots i_N\rangle = M_1|i_1\rangle \otimes M_2|i_2\rangle \otimes \cdots \otimes M_N|i_N\rangle \tag{1.7}$$

For instance,

$$IXZ\left(|000\rangle + |111\rangle\right) = |010\rangle - |101\rangle \tag{1.8}$$

A sequence of Pauli operators may be described with a shorter notation which accounts only for non-identity operators and indexes them with the index of the qubit they affect. For example, the Pauli operators in (1.8) may be represented as $X_2 Z_3$. The K-qubit Pauli group affecting the state of K qubits can be written as

$$G_K = \pm\{I, X, Z, Y\}^K \tag{1.9}$$

and given any two elements $P, Q \in G_K$ the following holds [5]

$$\begin{array}{lll} \text{if} & PQ = QP & \Rightarrow \quad \text{P,Q commute} \\ \text{if} & PQ = -QP & \Rightarrow \quad \text{P,Q anti-commute.} \end{array} \tag{1.10}$$

This means that two elements of G_K commute if the number of positions in which they differ from each other, not including the identity, is even. If this number is odd then the two operators are said to anti-commute. For example, the $XXIIZ$ commute with $YIIIX$, since both positions one and five differ, while all the other positions either include the identity or do not differ. In contrast, the operator $IZIII$ anti-commutes with $XXIIZ$, because only position two accounts for two different non-identity operators in both sequences.

1.2 Quantum Error Correction

Quantum codes aim to encode a quantum state of K (logical) qubits into a quantum state of N (physical) qubits. In a quantum system, qubits may be subject to different types of error. In particular one may have a bit-flip, a phase-flip or both at the same time resulting in the operators X, Y and Z, respectively. The first code able to correct both bit-flips and phase-flips was introduced by Shor in 1995 [3]. In this code a single (logical) qubit is encoded in nine (physical) qubits according to

$$\begin{aligned} |\bar{0}\rangle &= \frac{1}{\sqrt{8}}\left(|000\rangle + |111\rangle\right)\left(|000\rangle + |111\rangle\right)\left(|000\rangle + |111\rangle\right) \\ |\bar{1}\rangle &= \frac{1}{\sqrt{8}}\left(|000\rangle - |111\rangle\right)\left(|000\rangle - |111\rangle\right)\left(|000\rangle - |111\rangle\right). \end{aligned} \tag{1.11}$$

The bar over the state zero (or one) on the right-hand side of (1.11) indicates that a logical zero (or one) is encoded to the state on the left-hand side.

If the first qubit is subject to a flip, hence affected by the error $E = X_1 = XIIIIIIII$, then the Pauli operators $M_i = Z_1Z_2$ and $M_j = Z_2Z_3$, respectively, anti-commute and commute with E, then:

$$\begin{aligned} M_iE &= S_iEM_i \\ M_jE &= S_jEM_j \end{aligned} \tag{1.12}$$

The previous equation provides the couple $S_iS_j = -1 + 1$. Those two values are the result of measuring M_i, M_j if the error E occurs. If the error appears over the second qubit, the two measurements would lead to $-1 - 1$ and if it happens to be over the third, the measurement would have been $+1 - 1$. These operators provide a full mapping over the first three-qubit and they give a diagnosis analogous to the classical syndrome. Similarly Z_4Z_5, Z_5Z_6 and Z_7Z_8, Z_8Z_9 provide the same diagnosis, respectively, over the second triplet of qubits and over the last. The result of the bit-flip, operator X, over every bit is shown in Table 1.1. The columns of the Table can be indicate where the error

Error stabilizer	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Y_1
$ZZIIIIIII$	-1	-1	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	-1
$IZZIIIIII$	$+1$	-1	-1	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$
$IIIZZIIII$	$+1$	$+1$	$+1$	-1	-1	$+1$	$+1$	$+1$	$+1$	$+1$
$IIIIZZIII$	$+1$	$+1$	$+1$	$+1$	-1	-1	$+1$	$+1$	$+1$	$+1$
$IIIIIIZZI$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	-1	-1	$+1$	$+1$
$IIIIIIIZZ$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	-1	-1	$+1$

Table 1.1: stabilizer for the detection of bit-flip operator X of the Shor code and the corresponding syndrome

occurred, similar the syndrome tables for to classical code. As mentioned above the code introduced by Shor is able of correcting also the phase flips. The diagnostic operators used for this purpose are $XXXXXXIII$ and $IIIXXXXXX$ and the relative quantum syndrome is reported in Table 1.2.

Error stabilizer	X_1	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9	Y_1
$XXXXXXIII$	$+1$	-1	-1	-1	-1	-1	-1	$+1$	$+1$	$+1$	-1
$IIIXXXXXX$	$+1$	$+1$	$+1$	$+1$	-1	-1	-1	-1	-1	-1	$+1$

Table 1.2: stabilizer for the detection of phase-flip operator Z of the Shor code

From the columns of the Table relative to a pure phase-flip can be detected which block

underwent a phase-flip but not which of the three elements composing the block has been affected by the flip. Since the block imposes that the signs of the three elements are multiplied, if an error occurred on one of them, changing the sign of the block would correct this error no matter which element was affected by it. Since the error Y accounts for both the bit-flip and the phase-flip, an error of this type over any position can be detected by checking both tables. For Example, an error $E = Y_1$ would have both the output of X_1 over the first Table and Z_1 in the second, as it is shown in the last column of both tables.

1.3 Stabilizer Codes

Stabilizer codes are the quantum analog to linear codes. Precisely, a stabilizer group $\mathcal{S}$ is a set of Pauli operators on N qubits which commute with each other. Hence any $S_i \in \mathcal{S}$ commute with any $S_j \in \mathcal{S}, \forall i, j$. The diagnostic operators reported in the previous section, both Table 1.1 and 1.2, commute with each other, in fact, they compose the set of generators of a stabilizer group $\mathcal{S}$. The complete group of stabilizers is composed of all the sets which commute with each other and are constructed under multiplication from the original set, e.g.

$$(ZZIIIIIII)\text{x}(IIIXXXXXX) = ZZIXXXXXX \tag{1.13}$$

A state $|\psi\rangle$ is defined to be a codeword of the stabilizer $\mathcal{S}$ if the following holds:

$$S_i|\psi\rangle = |\psi\rangle \quad \forall i \tag{1.14}$$

meaning that it has to be a $+1$ eigenstate for all the stabilizers. Recall the definition of a state of K-qubit, (1.3), then the definition of codeword would lead to the encoding reported in (1.11), the original Shor code defined to evaluate the stabilizers. In quantum communication the error may be defined as a set of Pauli operators taking a state $|\psi\rangle$ to a corrupted state $E_\alpha|\psi\rangle$ with E_α being a collection of operators. The corrupted state may be the ± 1 eigenstate of the stabilizers, which depends on whether or not the error operators commute or anti-commute with the stabilizers. In case the error operator E_α commutes with stabilizer S_i, we have

$$S_i E_\alpha|\psi\rangle = E_\alpha S_i|\psi\rangle = E_\alpha|\psi\rangle$$

whereas when E_α anti-commutes with stabilizer S_i, we have

$$S_i E_\alpha|\psi\rangle = -E_\alpha S_i|\psi\rangle = -E_\alpha|\psi\rangle.$$

Whether the error commute or anti-commute with the stabilizer determines the result of the equation above, consequently, the commutative property of the error defines a sequence of ± 1, which elements account for the different stabilizers. This sequence is the equivalent of the syndrome in classical coding theory. In quantum mechanics, any measurement of the state that yields information about it degrades it. The evaluation of

the syndrome depends only on the relation between the error and the stabilizers, not on the state of the qubit, hence it does not degrade it. The syndrome can be used to find the error pattern yielding the corrupted state and this error can be applied to the latest in order to correct it.

1.4 Binary/4-ary representation of quantum codes

Recall that Y operators may be represented as the product of the X and Z operators, see (1.13). In this section this characteristics will be adopted to map the quantum over the classical coding theory. Firstly, by mapping the Pauli operators onto $(\mathbb{F}_2)^2$ according to:

$$I \to (0,0) \quad X \to (0,1) \quad Z \to (1,0) \quad Y \to (1,1) \tag{1.15}$$

The $(N-K)$ stabilizer generators of the $[N,K]$ stabilizer code reported in the tables of Section1.1 can be used as rows of the binary representation of parity-check matrix H of that code. Matrix H has size $[N-K, 2N]$, from now on $n = 2N$. The first N columns define the matrix H_X and have a 1 only in positions where there are stabilizers whose binary representation has a one on the right digit (X and Y). The remaining N columns define the matrix H_Z and have a 1 only in positions where there are stabilizers whose binary representation has a one on the left digit (Z and Y). This representation derives from the fact that the product of the X and the Z stabilizer, with a phase factor, produce the Y stabilizer. Similarly, the *bit-wise addition* of (0,1) and (1,0) produce (1,1), hence Y. The matrix constructed with this method is

$$H = [H_X|H_Z] = \left[\begin{array}{ccccccccc|ccccccccc} 1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1 \end{array}\right]. \tag{1.16}$$

The commutative property which applies to the Pauli operators now becomes the *symplectic product* or *twisted product*. Given two binary vectors $\boldsymbol{v} = [\boldsymbol{v}_X, \boldsymbol{v}_Z]$ and $\boldsymbol{u} = [\boldsymbol{u}_X, \boldsymbol{u}_Z]$, their twisted product can be computed as

$$\begin{aligned} \boldsymbol{v} \odot \boldsymbol{u} &= \boldsymbol{v}_X \cdot \boldsymbol{u}_Z + \boldsymbol{u}_X \cdot \boldsymbol{v}_Z = \\ &= \sum_{i=1}^{N} v_{Xi} u_{Zi} + \sum_{i=1}^{N} u_{Xi} v_{Zi} \end{aligned} \tag{1.17}$$

with all the sums modulo 2. In a stabilizer group, by definition, any stabilizer has to commute with any other stabilizer. The commutative property among stabilizers is translated

into the orthogonality of the rows of H with respect to the twisted product. In fact $\forall m$ with $m \in \{1, \ldots, N-K\}$ the following holds,

$$\boldsymbol{r}_m \odot \boldsymbol{r}_{m'} = \boldsymbol{r}_{X_m} \cdot \boldsymbol{r}_{Z_{m'}} + \boldsymbol{r}_{X_{m'}} \cdot \boldsymbol{r}_{Z_m} = 0 \mod 2 \quad \forall m' \in \{1, \ldots, N-K\} \tag{1.18}$$

with $\boldsymbol{r}_{X_m}$ a row of H_X and $\boldsymbol{r}_{Z_m}$ a row of H_Z. The quantum syndrome is named after its capability of mapping the measurement of the error affecting the qubits. This mapping is performed thanks to the commutative property of the stabilizers with respect to the error. The symplectic product of the parity-check matrix (PCM) H and an error vector $\boldsymbol{e}$ represented as $[\boldsymbol{e}_X|\boldsymbol{e}_Z]$ is evaluated according to:

$$S = H \odot \boldsymbol{e}^T = H_X \boldsymbol{e}_Z^T + H_Z \boldsymbol{e}_X^T \tag{1.19}$$

Since the symplectic product $\odot$ can be seen as the product of $[H_X|H_Z]$ with $[\boldsymbol{e}_Z|\boldsymbol{e}_X]^T$ from now on the binary representation of the error vector will be $[\boldsymbol{e}_Z|\boldsymbol{e}_X]$ instead of $[\boldsymbol{e}_X|\boldsymbol{e}_Z]$.

For example, the binary representation of the error vector $E = X_1$ assuming $N = 7$ is $\boldsymbol{e}_X = [100000000]$ and $\boldsymbol{e}_Z = [000000000]$. With the PCM and the error, both in binary form, it is possible to evaluate the syndrome according to (1.19), $\boldsymbol{S} = [00100000]$. This result agrees with the first column of Tables 1.2 and 1.1, if the -1 and $+1$ of the table are mapped onto 1 and 0, respectively.

The Pauli operators may also be mapped onto the Galois Field(4) elements with the equivalent symbol notation

$$I \to 0 \quad X \to 1 \quad Z \to \alpha \quad Y \to \bar{\alpha} \tag{1.20}$$

To properly adopt this representation the sum and product in $\mathbb{F}_4$ are reported in Box (1.3)

$+$	0	1	α	$\bar{\alpha}$
0	0	1	α	$\bar{\alpha}$
1	1	0	$\bar{\alpha}$	α
α	α	$\bar{\alpha}$	0	1
$\bar{\alpha}$	$\bar{\alpha}$	α	1	0

$\times$	0	1	α	$\bar{\alpha}$
0	0	0	0	0
1	0	1	α	$\bar{\alpha}$
α	0	α	$\bar{\alpha}$	1
$\bar{\alpha}$	0	$\bar{\alpha}$	1	α

Box 1.3: Addition and Multiplication in $\mathbb{F}_4$

According to this isomorphism, the multiplication of Pauli operators became the addition in $\mathbb{F}_4$. Furthermore, because the Pauli operators are mapped over a 4-ary symbol it is possible to represent PCM as a 4-ary matrix. The example of the Shor code for 9-qubits

encoder then becomes

$$\hat{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ \alpha & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & \alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & \alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha & \alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & \alpha \end{bmatrix}. \tag{1.21}$$

The syndrome can be evaluated with this version of the PCM. The commutative property of the stabilizer code is represented by the field trace of the inner product. Consistently with the other two representations, also in this case the syndrome is binary. An important feature of the field trace is the following: $\mathrm{Tr}(0) = \mathrm{Tr}(1) = 0$ and $\mathrm{Tr}(\alpha) = \mathrm{Tr}(\bar{\alpha}) = 1$. According to [6] Two operators in the set of stabilizers commute iff the inner product of their images, the vectors over GF(4), is either 1 or 0, hence their trace is zero. Consequently, the syndrome can be written as:

$$S_i = \mathrm{Tr}\langle \hat{H}_i, \hat{E} \rangle \tag{1.22}$$

with $\hat{H}_i$ a column of the parity check matrix with elements in GF(4). The quantum to classical isomorphism is summarised in Table 1.3.

Pauli	$(\mathbb{F}_2)^2$	$\mathbb{F}_4$
I	$(0,0)$	0
X	$(0,1)$	1
Z	$(1,0)$	α
Y	$(1,1)$	$\bar{\alpha}$
Multiplication	Bit-wise Addition	Symplectic Product
Commutativity	Addition	Trace of the Inner product

Table 1.3: Summary isomorphism Quantum and Classical codes

1.5 CSS codes

An important class of quantum codes are *CCS Codes* [7], named after Calderbank, Shor and Steane, which have a parity check matrix in the form

$$H = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix} \tag{1.23}$$

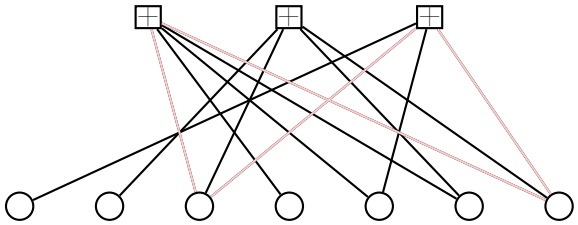


Figure 1.4: Tanner graph of the Hamming code (1.25) with $N = 7$, and $m = 3$

with both H_1 and H_2 having the same size, $(M \times N)$. The commutativity constraint (1.18) of a quantum stabilizer code translates into

$$H = [H_X|H_Z] \Rightarrow H_X H_Z^T = H_1 H_2^T = 0. \tag{1.24}$$

In the special case in which the two matrices H_1 and H_2 are equal, the code is referred to as a *dual-containing CSS code*. For example, the $(7, 4)$ Hamming code may be used for the creation of a code of this type. For instance, let

$$H_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{1.25}$$

In this case, we have $H_1 H_1^T = 0$ since all the rows present an even number of ones. Hence, the constraint over the rows is satisfied.

1.6 QLDPC codes

Due to practical reasons, one of the most interesting class of quantum error correction codes is quantum low-density parity-checks (LDPCs). An LDPC code is a code with a sparse parity check matrix H. LDPCs codes are typically represented as a bipartite or (Tanner) graph [8]. For example, in Figure 1.4 the Tanner graph of the $(7, 4)$ Hamming code with parity check matrix H_1 is shown. The performance of LDPCs under belief propagation decoding is known to deteriorate when short cycles are present in its Tanner graph [9]. A cycle is defined as a path over a graph which closes back on itself, highlighted in Figure 1.4. Thus, when designing an LDPC code, one usually tries to obtain a large *girth*, which corresponds to the length of the smallest cycle in the graph.

In order to satisfy the constrain reported in (1.24), the PCM H of the dual containing Calderbank-Shor-Steane (CSS) code must present rows which overlap with each other on an even number of positions. This, by definition, leads to a Tanner graph with an enormous number of small loops (of length 4). One can drop the dual containing CSS constraint to get codes with better properties.

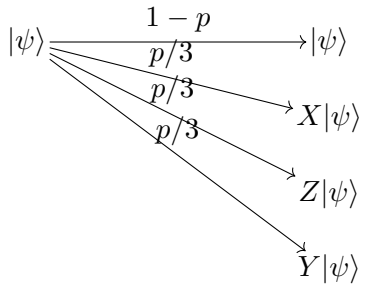


Figure 1.5: Transition probability of the depolarizing Channel with probability p

1.7 Measurement and Quantum Channel

In this section, a brief explanation of the effects of measurement and how a quantum channel may be modelled in order to apply the analog to classical decoding techniques in a quantum environment is presented. The measurement of the quantum information itself represents a constraint over the employment of different decoding algorithms. As introduced in the previous sections, the quantum bit (qubit), differently from the classical bit, can be in a superposition of states (1.1). Let us consider a generic quantum state $|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle$. When the qubit is measured, it collapses to the state $|0\rangle$ with probability $|\alpha_0|^2$ and to $|1\rangle$ with the probability $|\alpha_1|^2$. As a workaround one usually measures the effect of the error in form of syndrome (see preceding discussion) and applies a correction to the corrupted quantum states.

Another important feature of the quantum channel, which imposes strict constraints over the decoding scheme, is called *decoherence*. In a quantum channel, it is also known as quantum noise and represents the principal impairment over the practical implementation of a quantum communication system [10]. It is also described as destructive interaction between the environment and the qubit [3]. The channel in quantum communication is strongly hardware dependent. Usually a canonical model is adopted, the depolarizing channel, which will be employed also in this study.

1.7.1 Depolarizing Channel

A widely adopted model is the depolarizing channel model with error probability p. It should be recalled that a quantum state of N qubits is subject to a random Pauli error

$$E = E_1 \otimes \cdots \otimes E_N \tag{1.26}$$

where all $E_i \in \{I, X, Z, Y\}$ affecting the single qubit may cause independently a bit-flip, a phase-flip or both with the same probability

$$P(E_i = X) = P(E_i = Z) = P(E_i = Y) = p/3. \tag{1.27}$$

The depolarizing channel defined by the error probability p assumes that the errors

X,Y,Z occur with the same error probability $p/3$. Then the channel is isomorphic to a *4-ary symmetric channel* [11]. In the decoding, as it will be shown later, the channel may be treated as two independent binary-symmetric channels (BSCs) with crossover probability $2p/3$ [10]. With one channel for the X and one for the Z, this approximation removes the correlation between these two types of errors.

1.8 Degeneracy in Quantum Codes

In classical coding theory, different error vectors applied to the same codeword lead necessarily to different received words. However, in quantum communication systems, it is possible that two different errors applied to the same quantum state $|\psi\rangle$ yield the same (corrupted) state $|\phi\rangle$. This effect is known as *degeneracy*. In particular, if we consider a quantum $|\phi\rangle$ that belongs to the $+1$ eigenspace of a stabilizer code (i.e., it is a codeword), we have that two operators E_α and E_β will yield the same quantum state whenever they differ by a stabilizer,

$$\begin{aligned} E_\alpha &= E_\beta S_i \\ E_\alpha|\phi\rangle &= E_\beta S_i|\phi\rangle = E_\beta|\phi\rangle. \end{aligned} \tag{1.28}$$

Degeneracy has deep implications in quantum error correction. For example, consider a quantum system in which the state $|\psi\rangle$ has been altered to $|\phi\rangle = E_\alpha|\psi\rangle$. As mentioned in the previous sections, in order to bring back the system to state $|\psi\rangle$, it is sufficient to apply (again) the same operator E_α,

$$E_\alpha|\phi\rangle = E_\alpha E_\alpha|\psi\rangle = |\psi\rangle.$$

Due to degeneracy, we have that the quantum system may also be brought back to its original state $|\psi\rangle$ by applying any operator $E_\beta = S_i E_\alpha$, where S_i is any of the stabilizers. In fact, for a given stabilizer code, all the possible error patterns may be grouped in equivalence classes or *cosets*. A coset is a set of error operators which differs by a stabilizer. Hence any error operator in the same coset applied to $|\psi\rangle$ yields the same state. Thus, given $E_j|\psi\rangle$ and E_i an error operator of the coset $\mathcal{G}$,

$$\forall E_i \in \mathcal{G} : E_i|\psi\rangle = E_j|\psi\rangle. \tag{1.29}$$

1.9 Optimal decoding rule for QEC codes

Since the syndrome is a result of a measurement, decoding is performed as in the classical case, taking into account some peculiarities of quantum codes.

A classical maximum likelihood (ML) decoder would consider all error patterns $\boldsymbol{e} = [\boldsymbol{e}_Z, \boldsymbol{e}_x]$ which yield the observed syndrome $\boldsymbol{S}$ and pick the most likely one:

$$\hat{e} = \underset{\boldsymbol{e}|H\boldsymbol{e}=\boldsymbol{S}}{\arg\max}\, p(\boldsymbol{e}|\boldsymbol{S}) \tag{1.30}$$

In the classical domain, the most likely error pattern associated with a given syndrome, for a BSC is the error pattern with the minimum weight. Unfortunately, due to (possible) degeneracy of quantum codes, this metric is not optimal in the quantum scenario. According to [12] an error pattern $\boldsymbol{E}$ can be decomposed into three different terms: a pure error, T, which represents the effective centralizer coset that contains $\boldsymbol{E}$, a logical operator,L, which represents the stabilizer coset that contains $\boldsymbol{E}$, and a stabilizer component, W, which represents the specific operator in the stabilizer associated with $\boldsymbol{E}$. The authors show that for quantum codes the optimal decoding rule is

$$\hat{L} = \arg\max_{L} p(L|\boldsymbol{S}) \tag{1.31}$$

also named the degenerate quantum maximum likelihood decoding rule. In [12] the authors also show that a list of candidates $\boldsymbol{e}|H\boldsymbol{e} = \boldsymbol{S}$ grouped into cosets has the optimal solution in the coset with the highest likelihood.

$$\hat{\mathcal{G}} = \underset{\mathcal{G} \text{ for which } \exists e \in \mathcal{G} | H\boldsymbol{e} = \boldsymbol{S}}{\arg\max} \sum_{\boldsymbol{e} \in \mathcal{G}} p(\boldsymbol{e}|\boldsymbol{S}) \tag{1.32}$$

Once the proper coset is identified, any member can be used to correct the error. This result is due to the degeneracy in quantum codes. Some of the decoders in the following chapters produce a list of candidates and the decision will be on the candidates with the minimum weight, of course the list created will be a reduce list with respect to all the possible candidates and the decision over the codeword with minimum weight will not coincide with the ML decoder. To evaluate the degenerate quantum maximum likelihood decoding rule (1.31), it should be computed the probability of each coset by adding the probabilities of all the operators that make up the coset. This approach will be simulated in the following by grouping the error pattern present in the list of candidates.

Chapter 2

Classical Decoding Algorithms

Several algorithms for different types of codes can be found in the literature. We focus on quantum LDPC codes and first discuss some decoding algorithms.

2.1 Belief Propagation

In this section, a brief description of binary belief propagation (BP) is reviewed. BP decoders usually yield an approximation of the symbol-wise maximum *a posteriori* (MAP) [8] decoders. They implement the symbol MAP rule when the code's bipartite graph is a tree, i.e., when it does not have cycles. Given an observed syndrome $\boldsymbol{S}$, a symbol-wise MAP decoder outputs the most likely error bit-wise, $\boldsymbol{e} = (e_1, e_2 \cdots, e_n)$. Formally,

$$\tilde{e}_i = \underset{e_i \in \mathbb{F}_2}{\operatorname{argmax}} P\left(e_i | \boldsymbol{S}\right). \tag{2.1}$$

The BP operates by exchanging messages over the Tanner graph of H. The Tanner graph provides a complete representation of an LDPC code [13]. Let us recall that a Tanner graph is described as a bipartite graph whose nodes may be check node (CN) or variable node (VN). CN c_i and VN v_j are joined by an edge if and only if the element h_{ij} of the parity-check matrix is 1. The Tanner graph of Shor's code, whose parity-check matrix is given in (1.16) is shown in Figure 2.1. Since it is a CSS code, its PCM can be written as

$$H = \begin{bmatrix} H_x & 0 \\ 0 & H_z \end{bmatrix}. \tag{2.2}$$

The PCM in the form 2.2 implies that the Tanner graph consists of two unconnected Tanner graphs corresponding to the two sub-matrices. In fact, the first two check nodes are linked only to half of the variable nodes while the other check nodes are linked only to the remaining half. This design allows in principle decoding the X and Z parts independently, although the depolarizing channel introduces dependence between the X and Z components of the qubits. Ignoring this dependence, the qubit error rate can be obtained as the sum of the bit error rates (BERs) of the two classical codes.

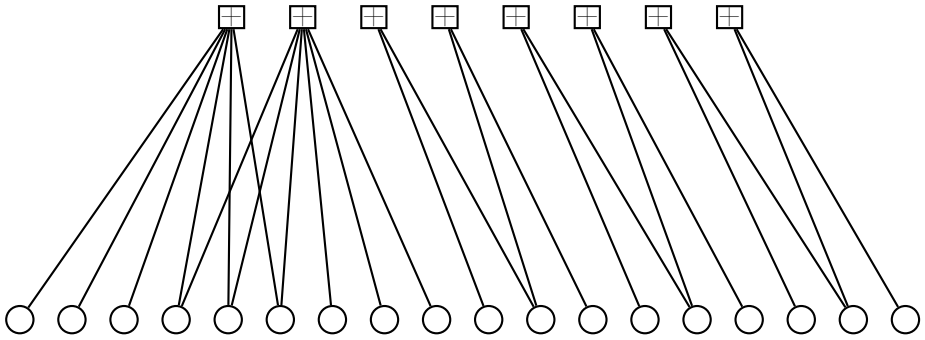


Figure 2.1: Tanner graph of the low-density parity-check code (see also (1.16)) with $n = 18$, and $m = 8$

BP decoding can be understood as an algorithm in which the CNs and VNs, exchange messages along the edges of the bipartite graph. In particular, BP is based on the exchange of extrinsic messages, which implies that the information passed to a node does not contain information already known at the receiving node. This algorithm relies on the assumption that the message passed is independent throughout the decoding process (i.e., it assumes absence of cycles in the Tanner graph). The information exchanged between nodes can be a probability, i.e., belief that the bit is one or zero, the ratio of thees two probabilities, called likelihood ratio (LR), or its logarithm, the log-likelihood ratio (LLR). In the thesis the LLR is employed.

2.1.1 Belief Propagation Syndrome Decoding

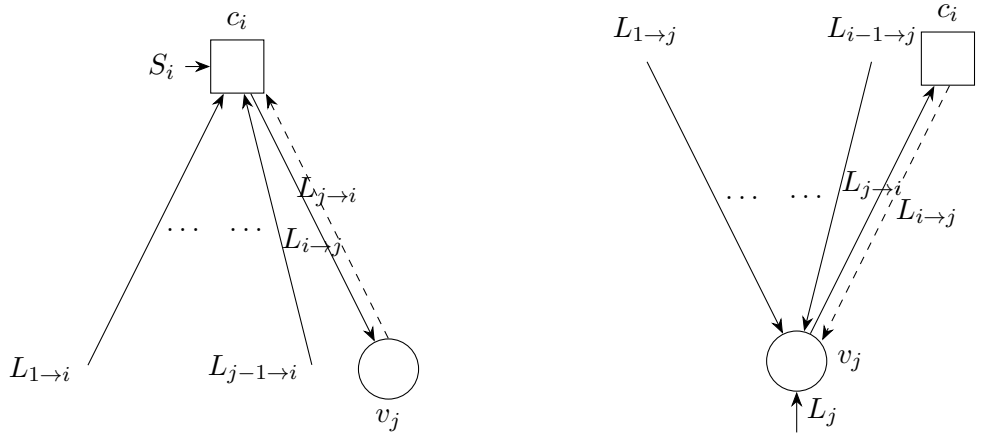
Syndrome decoding can be also illustrated on the code's Tanner graph. In this setup, the CNs still represent constraints on the code symbols. The VNs represent the error pattern, rather than a codeword. Hence, the parity checks might not be satisfied, but have a parity which is given by the syndrome vector. The task of the decoder is to recover the error pattern which yields the observed syndrome.

Given p, the error probability of the depolarizing channel (1.7.1), we may define a prior LLR for each of the VNs. For v_j we have,

$$L = L_j = \log\left[\frac{1 - 2p/3}{2p/3}\right]. \tag{2.3}$$

Hereby, we exploit the knowledge of the channel error probability and hence bias the error pattern to be sparse. The syndrome-based decoding algorithm is summarized in the following steps:

- **Initialization** : For each variable node v_j with $j \in \{1, \cdots, n\}$ evaluate the log-likelihood ratio according to (2.32).



(a) CN decoder (b) VN decoder

Figure 2.2: Node processors

- **CN decoder** : Let $N(c_i)$ being the set of neighbour nodes of c_i, a neighbour node is defined as a node connected to c_i,

$$N(c_i) = \{v_j : H_{ij} = 1\}. \tag{2.4}$$

Let the message $L_{i\to j}$ be the LLR at the output of c_i to v_j. According to [8] we have

$$L_{i\to j} = (-1)^{S_i} 2 \tanh^{-1}\left(\prod_{j'|v_{j'}\in N(c_i)\setminus v_j} \tanh\left(\frac{1}{2}L_{j'\to i}\right)\right) \tag{2.5}$$

- **VN decoder** : Let $N(v_j)$ being the set of neighbour nodes of v_j. The message $L_{j\to i}$ from v_j to c_i is

$$L_{j\to i} = L_j + \sum_{i'|c_{i'}\in N(v_j)\setminus c_i} L_{i'\to j} \tag{2.6}$$

- **A posteriori LLRs**: For $i \in \{1, ..., n\}$ compute

$$L_j^{\text{total}} = L_j + \sum_{i|c_i\in N(v_j)} L_{i\to j}. \tag{2.7}$$

Differently from the previous step, now all the incoming messages flowing into the variable node v_i are summed because this LLR does not need to be sent back to any other check node. In fact, L_j^{total} will be used for the hard decision.

- **Hard decision and Syndrome check**: Based on the LLRs evaluated in the previous step the hard decision over the symbols is done as

$$\begin{aligned} \tilde{e}_j &= 0 \quad \text{if } L_j^{\text{total}} > 0 \\ \tilde{e}_j &= 1 \quad \text{if } L_j^{\text{total}} < 0 \end{aligned} \tag{2.8}$$

With the estimated error vector $\tilde{\boldsymbol{e}}$ the syndrome $\tilde{\boldsymbol{S}}$ is computed according to (1.19). If the syndrome is equal to the syndrome observed from the measurements the algorithm stops. Otherwise, it goes to the next iteration and computes (2.5).

$$\text{if} \quad \tilde{\boldsymbol{S}} = \boldsymbol{S} \quad \Rightarrow \quad \hat{\boldsymbol{e}} = \tilde{\boldsymbol{e}}. \tag{2.9}$$

Under the assumption of statistically independent of the messages, BP is optimal, in the sense that the MAP yields exactly the a posteriori probability (APP) [9]. Thus, if we denote by γ the girth of the code considered, this condition (statistical independence) is true up to the $\gamma/2$-th iteration. Unfortunately, many quantum codes are characterized by a small girth. For example, in [14] the author shows that the CSS dual containing codes constructed from LDPC matrices lead to an enormous number of cycles of length 4. For these reasons, the performance of BP decoding is far from optimal for many QLDPC families. In the following sections, we describe other algorithms that may provide better performance.

2.2 Information Set Decoding with Bit-Flipping

In [15] Gallager proposed two algorithms for the decoding of the LDPC codes over a binary-symmetric channel (BSC). In his paper, the author assumed that the received sequence was accessible at the receiver. Starting from this information the decoder evaluates all the parity-check sums and then flips any digit in the hard decision of the received sequence $\boldsymbol{r}$ that are contained in a fixed number of unsatisfied check-sums. Consider the following transmitted codeword $\boldsymbol{c}$ and error pattern introduced by the channel $\boldsymbol{e}$:

$$\begin{aligned} \boldsymbol{c} &= \left[\begin{array}{cccccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \\ \boldsymbol{e} &= \left[\begin{array}{cccccccccccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{aligned} \tag{2.10}$$

and consider the seven-qubit CSS code from [6] in its binary matrix representation:

$$H = \left[\begin{array}{cccccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \tag{2.11}$$

Given the received sequence and syndrome, it is possible to extract the unsatisfactory checks, highlighted below.

$$\boldsymbol{r} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \quad \boldsymbol{S} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.12}$$

From the example, it is clear that flipping any of the first three digits of $\boldsymbol{r}$ would lead to a non-zero syndrome, which is different from the previous. But by flipping the fourth digit of the received sequence and computing the syndrome the result would be an all-zeros vector. In fact, in this case, the error is exactly on the fourth digit in (2.10). Generally, if the syndrome evaluated with the modified hard-decision is a non-zero vector then the procedure is performed again. This algorithm stops when a given number of iterations is reached or the vector obtained through the bit-flipping is a codeword, and hence it satisfies

$$\boldsymbol{S}' = H\boldsymbol{r}'^{T} = \boldsymbol{0} \tag{2.13}$$

2.2.1 Information Set Decoding of Quantum Codes

When it comes to quantum codes, there are two approaches that may be followed with the PCM is given in its binary form. In the first you split the system of equations into two systems: an X-part and a Z-part which are solved independently. The advantage is some reduction in complexity: e.g., $O((2N)^3)$ versus $O(2N^3)$ for gaussian elimination (GE). While the second, keep one system of equations; we go for this, since we want to try to account for the correlation between X,Z error. For quantum codes the sequence $\boldsymbol{r}$ can not be observed (see Section 1.7). In fact, only the syndrome $\boldsymbol{S}$ is accessible, and hence it is not possible to follow the same approach proposed by Gallager. For this reason, a decoding scheme focused exclusively on the syndrome is investigated.

The idea of information-set decoding (ISD) was first introduced by Prange [16]. The algorithm selects an information set and reconstructs a message from the corresponding entries, then re-encodes this message. This procedure assumes that a received sequence with no errors in the information set may be reprocessed without any changes in the information set and produce a correct codeword. For instance, given a code $\mathcal{C}$ defined by its generator matrix in systematic form $G = [I_m|P]$, the information set may be defined as the first K positions of the columns of G, hence $B_K = \{1, 2, ..., K\}$. Assume $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ the received sequence with $\boldsymbol{c} \in \mathcal{C}$ the transmitted codeword and $\boldsymbol{e} \in \{0, 1\}$ the error introduce by the channel. Let $\boldsymbol{e}_{B_K}$ be the entries of $\boldsymbol{r}$ corresponding to the positions in the information set. We may assume that no errors in the information set

happened (later we revert this assumption), i.e., $e_i = 0$, $\forall i \in B_K$. All other error values e_i, $\forall i \in [1, ..., n] \setminus B_K$, i.e., are assumed to be unknown. Then, we have to solve the system of equations

$$\boldsymbol{S} = H\boldsymbol{e}^T \tag{2.14}$$

For now, we define the information set as the $n-h$ right-most digits of the error sequence $\boldsymbol{e}$. To solve (2.14) the parity-check matrix needs to be into identity form which can be done by GE. This is the starting point of our algorithm.

Gaussian Elimination

The idea is to obtain H in identity form,

$$H' = \begin{bmatrix} I_m & \boldsymbol{h}'_{m+1} & \cdots & \boldsymbol{h}'_n \end{bmatrix} \tag{2.15}$$

where I_m is the $m \times m$ identity-matrix and $\boldsymbol{h}'_i$ is the $i-th$ column of the matrix after the operations, with $m = N - K$. If the rank is lower than the number of rows of the matrix Gaussian elimination would provide H' in the form

$$H' = \begin{bmatrix} 1 & 0 & \cdots & 0 & & & \\ 0 & 1 & \cdots & 0 & \vdots & & \vdots \\ \vdots & & \ddots & & \boldsymbol{h}'_{h+1} & \cdots & \boldsymbol{h}'_n \\ 0 & \cdots & 0 & 1 & \vdots & & \vdots \\ & \cdots & & \mathsf{O}_{(m-h)\times n} & \cdots & & \end{bmatrix} \tag{2.16}$$

where $\mathsf{O}_{(m-h)\times n}$ is the $(m-h)\times n$ matrix of all zeros. Usually in quantum codes, some redundancy is introduced resulting in a matrix having the rank (h) lower than the number of rows.

All row operations on the parity-check matrix need to be applied to the syndrome as well. The permutations among the columns and among the rows are named, respectively, $\mu_1(.)$ and $\phi_1(.)$. In the following, GE is performed over a PCM step-by-step to investigate for illustration. For simplicity, the matrix used in the example is full-rank $(m = h)$ and together with the syndrome $\boldsymbol{S}$, is reported in 2.12.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \quad \boldsymbol{S} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.17}$$

Since the first column is not in the identity form, the first row, highlighted in red, is summed to the second and the third, highlighted in blue. The same is done over the

syndrome. Once the first row is in standard form the algorithm passes to the second column. Here the position $(2,2)$ reports a zero, which means that the second row has to be exchanged. The function search for a row with an index greater than 2, which has a one in the $2nd$ position. In this case the $3rd$ row:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \boldsymbol{S} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.18}$$

The rows are swapped and the new row is summed to any other row which presents a one in the second column, to obtain also the second column in standard form. This procedure is followed until a column with all zeros below the position considered is encountered. In this example, the case occurs when the $4th$ column is considered.

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2.19}$$

In this case, the column without ones in the $4th$ position or below is switched with the $(h+1)th$ column, which in this example is the $9th$. When the identity matrix on the left side is achieved then the procedure is finished and as output is provided the PCM in the new set-up (H'), the syndrome ordered according to the rows operations performed over the matrix ($\boldsymbol{S'}$) and the permutation applied to the columns (μ_1). The results for the example presented are the following:

$$H' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \boldsymbol{S'} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 1 & 2 & 3 & 9 & 5 & 10 & 11 & 12 & 4 & 6 & 7 & 8 & 13 & 14 \end{bmatrix} \tag{2.20}$$

The columns highlighted in 2.20 represent the information set for this code. From now on, the sequence resulting from the encoding of the all-zeros sequence will be named the candidate for order-zero reprocessing. Orders higher than zero have to be defined and to do it reprocessing based on bit flipping (BF) should be introduced.

2.2.2 Reprocessing

We consider the information set $\boldsymbol{e}_{B_k}$ which was chosen to be a length-k all-zero vector assuming no errors in the information set, and the PCM full-ranked. We revert this assumption and create a list of error patterns, among which we select the most probable one. This is done by setting some entries of $\boldsymbol{e}_{B_k}$ to one.

Rearranging the components of $\boldsymbol{e}_{B_k}$ according to the permutation μ_1 leads to the sequence:

$$\boldsymbol{e'} = \boldsymbol{\mu_1}\left(\boldsymbol{e}_{B_k}\right) = \begin{pmatrix} e'_1 & e'_2 & \cdots & e'_n \end{pmatrix} \tag{2.21}$$

Note that the parity-check matrix is not in identity form (see (2.16)). Now we can solve (2.14) in order to obtain e'_i, assuming $(e'_{h+1}, \cdots, e'_n)$, with $e'_i = 0\ \forall i \in \{h+1, \ldots, n\}$. We refer to this step as encoding. We obtain a first candidate error pattern candidate $\boldsymbol{e}^0$,

$$\boldsymbol{e}^0 = \begin{pmatrix} e^0_1 & e^0_2 & \cdots & e^0_h & e'_{h+1} & \ldots & e'_n \end{pmatrix} \tag{2.22}$$

More precisely, for $1 \leq j \leq h$ the digits e^0_j are evaluated according to:

$$e^0_j = \sum_{i=h+1}^{n} h'_{ij} e'_i + s'_i \mod 2 \tag{2.23}$$

We can now flip some of the bits of $\boldsymbol{e}_{B_k}$ and repeat the procedure to generate more candidate error patterns. The algorithm can be split into several stages, also called orders ℓ of reprocessing. For the ℓ-th order the algorithm will make all possible changes of i of the $n-h$ right-most bits of $\boldsymbol{e'}$. For each change reconstruct the corresponding candidate $\boldsymbol{e}^j$ according to (2.23).

When all the

$$\sum_{i=0}^{\ell} \binom{n-h}{i} \tag{2.24}$$

possible candidates have been computed, order-ℓ reprocessing is finishes. Let $w_H(\boldsymbol{e}^j)$ denote the Hamming weight of a sequence. Then, we can assign to each candidate error pattern a penalty $w_H(\boldsymbol{e}^j)$. The final error pattern $\boldsymbol{e}^*$ is selected as the one with the lowest penalty. Of course, the final estimated $\hat{\boldsymbol{e}}^*$ can be obtained by permuting the components of $\boldsymbol{e}^*$ with the inverse permutation μ_1^{-1}, i.e.,

$$\hat{\boldsymbol{e}}^* = \mu_1^{-1}\left(\boldsymbol{e}^*\right) \tag{2.25}$$

In case we assume two independent BSCs which act on the different components of the codeword with equal error probability p, the most likely sequence is the one with the lowest number of bit-flips (ones), thus the one with the lowest penalty.

Consider now the example in 2.10. The first candidate corresponds to order-0 reprocessing, and the encoding would have the syndrome positions $\bar{B}_{n-h}$ and all zeros in positions B_{n-h}.

$$H' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad \boldsymbol{s}' = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Moving to $order-1$ reprocessing leads to the encoding of all the sequences which present a single bit-flip in the B_{n-h} positions. Consequently, out of the 7 candidates obtained with the encoding the one with a bit-flip over the 9th position has the minimum weight among the candidates. The unordered and ordered sequences appear as:

$$\boldsymbol{e}^4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\hat{\boldsymbol{e}}^4 = \mu_1^{-1}\left(\boldsymbol{e}^4\right) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In this case, $order-0$ reprocessing was not enough. This is due to the fact that the error was in the $n-h$ right-most digits, hence the information set decoding only was not able to correct it. If a single error had appeared in the complement of the information set, $order-0$ would have produced the error pattern searched.

2.2.3 Smart Bit-Flipping

Similarly to the Gallager algorithm, where the flips over the digit were performed only over unsatisfactory checks, also with the information set decoding and BF the flips may be applied over a reduced information set. Given the PCM and the syndrome in the form 2.20, a reduced list of candidates can be evaluated. The implementation consists of the search for the ones in the syndrome. Once they have been identified, the reprocessing is performed by changing only the bits which correspond to those unsatisfactory checksums. In the following the graphical representation is reported.

$$H' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad \boldsymbol{S}' = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In this example, only three bits are flipped in the reprocessing phase of the algorithm. This change affects the list of candidates whose number of elements, for order-ℓ reprocessing, changes from 2.24 to :

$$\sum_{i=0}^{\ell} \binom{n_{uc}}{i} \tag{2.26}$$

where n_{uc} is the number of bits which leads to unsatisfactory check-sums. This modification in the decoder does not affect strongly the performances but decreases the complexity since the encoding of the information set has to be performed in less time with respect to the canonical information-set decoding and bit-flipping.

2.3 Classical Ordered Statistics Decoding

Since their advent in [17], ordered-statistics-based algorithms have been widely investigated thanks to their good performance and moderate complexity. In [18] the author explores the advantages of searching the correct sequence over a reduced list of candidates, a list based on the reliability of the single bits. The soft-decision decoding scheme for binary linear block codes is composed of two steps: hard decision and reprocessing. In the decoding scheme presented by Fossorier, the hard decision is made over the re-ordered version of the received sequence instead of the actually received sequence. The ordering is made taking into account the reliability, ensuring however that an information set is selected. A discussion follows.

Consider a binary linear code C with the generator matrix G, given the received sequence $\boldsymbol{r}$ a first permutation $\lambda_1(.)$ is applied and a permuted sequence is obtained:

$$\boldsymbol{y} = \lambda_1(\boldsymbol{r}) = \left[\begin{array}{cccc} y_1 & y_2 & \ldots & y_n \end{array} \right] \tag{2.27}$$

with the reliability of the bits ordered in decreasing order:

$$|y_1| > |y_2| > \cdots > |y_n|. \tag{2.28}$$

The same permutation is applied also to the columns of the generator matrix, to obtain G' the generator matrix of a binary linear code C' equivalent to C. Later a second permutation, $\lambda_2(.)$ is applied to G' to take k independent columns in the first k positions, by keeping the ordering of the right-most columns ordered by reliability.

$$G'' = \lambda_2(\lambda_1(G)). \tag{2.29}$$

Consequently, row operations are performed over the generator matrix to obtain it in standard form (G_1). This last step may be carried out relying on Gaussian elimination, see Section 2.2.1. In fact, $\lambda_2(.)$ and $\mu_1(.)$ in Section 2.20 coincide. The second permutation is applied also to $\boldsymbol{y}$ and results in the vector $\boldsymbol{z}$, which is consistent with G_1. The code generated by the latest is equivalent to C' and C. Lastly, the hard-decision over the first k bits of $\boldsymbol{z}$ are encoded with G_1 to obtain the first candidate ($\boldsymbol{a}^0$) which is a

codeword of C_1. By performing the inverse permutation of λ_1 and λ_2 a codeword of C is obtained:

$$\hat{\boldsymbol{c}}^0 = {\lambda_1}^{-1}\left({\lambda_2}^{-1}\left(\boldsymbol{a}^0\right)\right) \tag{2.30}$$

The reprocessing is performed similarly to Section 2.2 but instead of encoding following (2.23), the encoding is a multiplication of the reprocessed vector of length k with the generator matrix. During order-ℓ reprocessing, for $1 \leq i \leq \ell$, all possible changes of i of the k bits in the most-reliable positions of $\boldsymbol{a}^0$ are performed and re-encoded to create the list of candidates. Note that the maximum number of bits eligible for bit flipping corresponds to the rank of the matrix, h. Thus, the number of total candidates for order-ℓ is

$$\sum_{i=0}^{\ell} \binom{h}{i}. \tag{2.31}$$

Finally, the candidate that minimizes the Euclidean distance to the received word is selected. In quantum communications, the reliability of the received sequence and the hard-decision are not accessible at the receiver. For those reasons, the hard-decision and the decision over the list of candidates in this scenario can not be made following [18], but the reprocessing based on the statistics may still be implemented if another decoder is applied to get the reliability of the bits. In the following section, this scenario is investigated. In [19] the authors proposed an equivalent decoding scheme which investigates the parity-check matrix instead of the generator matrix. This may be more interesting for the quantum codes because it relies on the syndrome for the creation of the list of candidates.

2.4 Quantum Codes and Ordered statistics

In [20] a decoding scheme is described which combines BP and order-statistics decoding (OSD), to overcome the gap between the BP decoder and the ML decoder. It is shown that the performance of the BP is improved thanks to the reprocessing performed according to the statistics of the single bits. In [21] the authors propose a similar approach but applied to quantum low-density parity-check (QLDPC) codes. Initially, the algorithm is presented with the use of a binary decoder, using the binary representation of the quantum symbols (1.15), but also non-binary versions are exploited. The paper investigates how OSD may be applied after BP to achieve better results. In the thesis, a similar approach is followed by applying BP in $\mathbb{F}_2$ first and then at every iterations perform OSD.

Without the received sequence it is not possible to evaluate the Euclidean distance between what has been received and the candidates. In fact, the syndrome and the parity-check matrix are the only tools available at the receiver, hence a procedure more biased towards [21] with respect to the one proposed by [20] is preferred. The estimated error pattern is provided by the previous iteration of the belief propagation. It is important to recall that at every iteration of BP (over $\mathbb{F}_2$) an estimate of the log-probability ratio of

each error position (e_i), given the observed syndrome $\boldsymbol{s}$, is produced,

$$L(e_i) = \log\left(\frac{P(e_i = 0|\boldsymbol{s})}{P(e_i = 1|\boldsymbol{s})}\right). \tag{2.32}$$

Next, the received sequence is obtained by applying a hard decision to the output of BP, i.e., to the log-likelihood ratios. For each bit in the binary representation of the quantum code, the absolute value of 2.32 is its reliability and may be employed in the OSD. Following [20], a test is initialized to decide when to stop the reprocessing. The test imposes that if the codeword produced by the OSD appears twice it is assumed to be the correct guess. The algorithm investigated in this thesis, for order-ℓ, is processed as follows:

- **BP decoding** : Performed in $\mathbb{F}_2$ according to Section 2.1
  - Initialization: evaluate the probability of each bit according to the syndrome.
  - CN update: compute the extrinsic information according to (2.5).
  - VN update: Process the incoming messages according to (2.6).
- **Early stopping criteria** Save the reliability of the bits evaluated in the previous step.
  - Create $\tilde{\boldsymbol{e}}$ such that $\tilde{e}_i = 1$ if $L(\tilde{e}_i) > 0$, and $\tilde{e}_i = 1$ if $L(\tilde{e}_i) < 0$
  - if $H\tilde{\boldsymbol{e}}^T = \boldsymbol{S}$ then the decoding algorithm halts and $\tilde{\boldsymbol{e}}$ is considered a valid decoding result:
    $$\hat{\boldsymbol{e}} = \tilde{\boldsymbol{e}}$$
  - Else, the algorithm starts the "Order-ℓ Reprocessing".
- **OSD decoding**: Based on the reliability values $|L(e_i)|$, sort them and the parity-check matrix (with increasing order) and determine its h least reliable independent columns. Those permutations are named $\lambda_1(.)$ and $\mu_1(.)$, respectively.
  - Apply the same permutations to $\tilde{\boldsymbol{e}}$
    $$\boldsymbol{e}' = \mu_1(\lambda_1(\tilde{\boldsymbol{e}})) \tag{2.33}$$
  - For $0 \leq i \leq \ell$ make all possible $\binom{n-h}{i}$ changes of i bits in the most-reliable positions (MRP) and for each change, determine the corresponding vector $\boldsymbol{e}_i$ according to (2.23).
- **Decision**:
  - Among the candidates selects the one with minimum Hamming weight ($\boldsymbol{e}^*$).
  - if $\boldsymbol{e}^* = \hat{\boldsymbol{e}}$ the test is satisfied and ${\lambda_1}^{-1}\left({\mu_1}^{-1}(\hat{\boldsymbol{e}})\right)$ is select as the decoded vector.

– Else, if the Hamming weight of $\boldsymbol{e}^*$ is lower than $\hat{\boldsymbol{e}}$ set the first as the decision at previous step

$$\text{If } w(\boldsymbol{e}^*) \leq w(\hat{\boldsymbol{e}}) \text{ then } \hat{\boldsymbol{e}} = \boldsymbol{e}^* \tag{2.34}$$

Start the next iteration with BP decoding.

In case the test or the early stopping criteria do not make any decision, the algorithm stops when the maximum number of iterations is reached. It should be also stated that at each iteration both the permutations introduced by the algorithm (2.33) are re-evaluated starting from the updated reliability $L(e_i)$. This affects the overall complexity of the algorithm since GE has to be performed every time. In this sense, some simplification may be applied as it will be shown in the following chapter.

Chapter 3

Enhanced Decoding Algorithms

This chapter elaborates on common decoding algorithms in the literature. It introduces modifications of those with the purpose of

a) simplifying those and hence making their practical implementation more appealing

b) improving their performance - if possible - to increase the error correction capabilities

c) gaining deeper insights into the capabilities of BP decoders with post-processing capabilities.

3.1 Non-Binary Belief Propagation

In Chapter 1, Section 1.1, the Pauli operators have been introduced but in the previous chapter, only their binary representation has been investigated. In this chapter, a study of their quaternary description is presented. Quartenary BP is well-known in the LDPC coding literature. For quantum LDPC codes we present a simplified version.

Consider the Tanner graph representation of a quartenary code. Opposed to the binary case, the edges of the *quantum Tanner graph* are labelled. In particular, the edge connecting CN c_i and VN v_j exists if and only if $h_{ij} \in \mathbb{F}_4 \setminus 0$. Then, its label corresponds to $h_{i,j}$. We consider the mapping between members of the Pauli group and elements of $\mathbb{F}_4$ from (1.20).Thus with slight abuse of notation, $h_{i,j}$ may take values, I X, Y, Z.

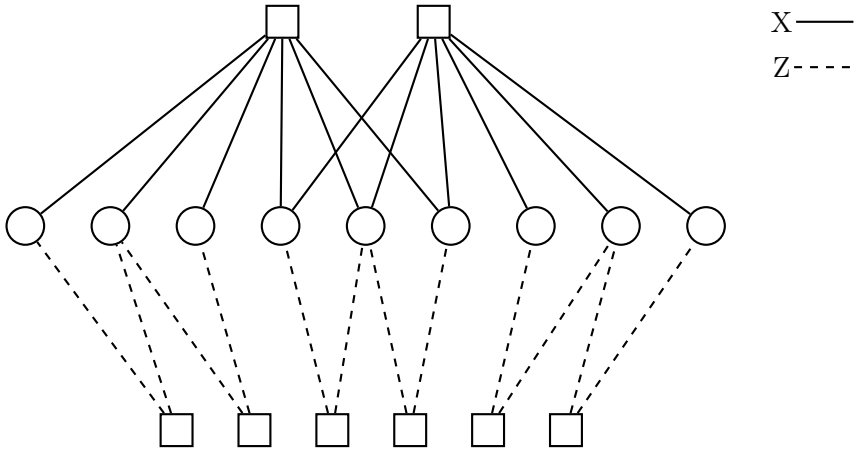


Figure 3.1: Quantum Tanner graph of the low-density parity-check code 1.21 with $N = 9$, and $m = 8$

We illustrate the graphical representation of a quantum Tanner graph following (1.21).

$$\hat{H} = \begin{bmatrix} X & X & X & X & X & X & I & I & I \\ I & I & I & X & X & X & X & X & X \\ Z & Z & I & I & I & I & I & I & I \\ I & Z & Z & I & I & I & I & I & I \\ I & I & I & Z & Z & I & I & I & I \\ I & I & I & I & Z & Z & I & I & I \\ I & I & I & I & I & I & Z & Z & I \\ I & I & I & I & I & I & I & Z & Z \end{bmatrix} \tag{3.1}$$

The messages exchanged among the nodes can be represented as vectors of four elements. Each element of the message vector represents the estimated probability of a given digit of the error pattern. Next, we will resort to a log-likelihood ratio representation in analogy to Section 2.1. Let the probabilities for the four Pauli operators $\{I, X, Z, Y\}$, be p_I,p_X,p_Z, and p_Y. We define the log-likelihood ratio (LLR) as

$$l = \left(0, \log \frac{p_X}{p_I}, \log \frac{p_Z}{p_I}, \log \frac{p_Y}{p_I}\right). \tag{3.2}$$

Observe that the first element is always zero and in principle can be removed. We leave it here for ease of presentation.

The non-binary version of the BP can thus be described by specifying the variable and check node processing, i.e., how the VNs and CNs compute the outgoing messages as a function of their input messages. We make the following observation. The computation in the CNs can be performed as for the binary case. This is a result of (1.22), i.e., two different Pauli operators yield the same field trace. Let p_1 be the sum of the probabilities of the Pauli operators which anti-commute with the stabilizer with the Pauli operator in

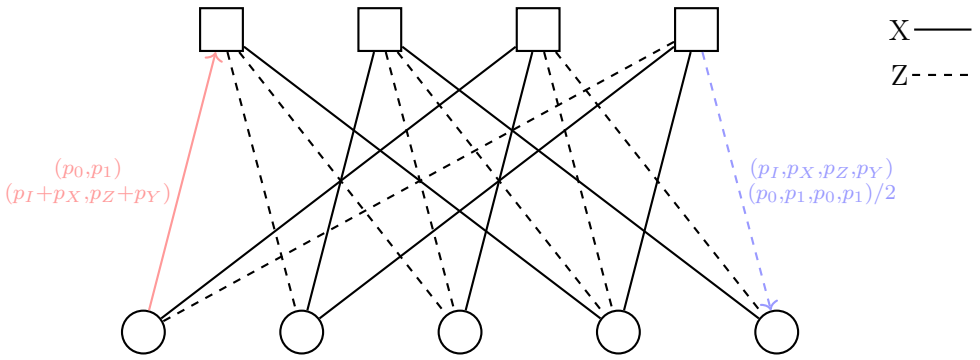


Figure 3.2: Quantum Tanner graph of the stabilizer for the five-qubit code 3.3 with $N = 5$, and $m = 4$

position ij of the PCM. Then $p_0 = 1 - p_1$ is the the sum of the probabilities of the Pauli operators which commute with the Pauli operator in position ij of the PCM. We can pass these two probabilities the CNs (or the respective LLR) and perform the computation as for the binary case. The variable-node processor processes the output messages in a quaternary fashion, meaning that the messages received at the variable node have to account for the probabilities p_I,p_X,p_Z and p_Y or the corresponding LLRs. Hence, we need a conversion from binary LLRs to 4-ary LLRs.

Consider the following PCM [6],

$$\hat{H} = \begin{bmatrix} \mathrm{X} & Z & Z & X & I \\ I & X & Z & Z & X \\ X & I & X & Z & Z \\ Z & X & I & X & \mathrm{Z} \end{bmatrix} \tag{3.3}$$

The edge of the Tanner graph relative to the element $H_{1,1}$ of $\hat{H}$ is highlighted in red, while the edge for $H_{5,5}$ is highlighted in blue. Consider the message over the red edge from v_1 to c_1, it may be converted from quaternary to binary by letting p_1 be the sum of p_Y and p_Z, and p_0 as $p_I + p_X$, since $H_{1,1} = X$. In contrast, the variable nodes have to receive the message in the quaternary form. The message from c_i to v_j has to convert the probabilities p_0 and p_1 to p_I,p_X,p_Z and p_Y. The conversion is performed by equally splitting the probability p_0 with the probabilities relative to the Pauli operator which commute with the Pauli operator in position ij of the PCM. Consider the message highlighted in blue in Figure(3.2), where $H_{5,5} = Z$, then $p_I = p_Z = p_0/2$ and $p_X = p_Y = p_1/2$.

Let us define soft-max operator,

$$\max^*(a, b) = \max(a, b) + \log(a + \exp(-|a - b|)) \tag{3.4}$$

We can now summarize the syndrome-based decoding algorithm in $\mathbb{F}_4$:

- **Initialization** : For each VN v_j with $j \in \{1, ..., N\}$ generate log-likelihood ratio vectors according to (3.2). As mentioned in Section 1.7.1 consider $p/3$ as the probability of any non-identity Pauli operator, then the non-binary LLR vector for v_j is

$$\boldsymbol{l}_j = \begin{bmatrix} 0 & \log\left(\frac{p}{3(1-p)}\right) & \log\left(\frac{p}{3(1-p)}\right) & \log\left(\frac{p}{3(1-p)}\right) \end{bmatrix} \tag{3.5}$$

  The message to the CNs has to be a scalar, hence the vector $\boldsymbol{l}_j$ cannot be sent as it is. The conversion consists in two soft-max operators, the first evaluated with the elements of $\boldsymbol{l}_j$ corresponding to the Pauli operators which commute with the Pauli operator of the edge through which the message is sent. The other, weighted (-1), accounts for the elements that anti-commute. According to (3.5) for any edge the initial message is

$$L^0_{j\to i} = \max^*\left[0, \log\left(\frac{p}{3\,(1-p)}\right)\right] - \max^*\left[\log\left(\frac{p}{3\,(1-p)}\right), \log\left(\frac{p}{3\,(1-p)}\right)\right] \tag{3.6}$$

- **CN decoder** : Let $N(c_i)$ being the set of neighbour nodes of c_i. Similarly to the binary decoder (2.2a) the check node c_i processes all the incoming messages. The incoming messages are in binary form, according to Figure(3.2), but are sent back in quaternary form to each neighbouring node. The message $L_{i\to j}$ is the LLR of the probabilities constrained to the check performed by c_i and the value of the syndrome bit $S_i \in \mathbb{F}_4$. According to [22] it may be approximated as:

$$L_{i\to j} = (-1)^{S_i} 2 \tanh^{-1}\left(\prod_{j'|v_{j'}\in N(c_i)\setminus v_j} \tanh\left(\frac{1}{2} L_{j'\to i}\right)\right) \tag{3.7}$$

- **VN decoder** : The binary LLR $L_{i\to j}$ from c_i to v_i needs to be converted into a non-binary LLR vector $\boldsymbol{l}_{i\to j}$. The conversion has to consider the Pauli operator of the element i, j of the PCM, by letting $\mathcal{Z}_{i,j}$ the set of Pauli operators (P) which commute with $H_{i,j}$ the message is:

$$l^{\mathrm{P}}_{i\to j} = \begin{cases} 0 & \text{for P} \in \mathcal{Z}_{ij} \\ -L_{i\to j} & \text{for P} \in \bar{\mathcal{Z}}_{ij}. \end{cases} \tag{3.8}$$

  Let $N(v_j)$ being the set of neighbour nodes of the variable node v_j. Then v_j processes all incoming messages ($\boldsymbol{l}_{i\to j} \in \mathbb{F}_4$) and provides back the extrinsic information to each neighbouring node in binary form. The vector message $\boldsymbol{l}_{j\to i}$ is

$$\boldsymbol{l}_{j\to i} = \boldsymbol{l}_j + \sum_{i'|c_{i'}\in N(v_j)\setminus c_i} \boldsymbol{l}_{i'\to j}. \tag{3.9}$$

  The message sent to $c_i \in N(v_j)$ is computed as follows,

$$L_{j\to i} = \max_{\mathrm{P}\in\mathcal{Z}_{ij}}{}^*\left(l^{\mathrm{P}}_{j\to i}\right) - \max_{\mathrm{P}\in\bar{\mathcal{Z}}_{ij}}{}^*\left(l^{\mathrm{P}}_{j\to i}\right). \tag{3.10}$$

- **Hard decision and Syndrome check**: At each variable node, the decoder selects the Pauli operator with the largest associated LLR.

$$\tilde{E}_j = \text{argmax}\left(\boldsymbol{l}_j + \sum_{i|c_i \in N(v_j)} \boldsymbol{l}_{i\to j}\right). \tag{3.11}$$

  With the estimated error vector $\tilde{E}$ the syndrome is computed according to 1.22 if the syndrome is equal to the syndrome given from the channel the algorithm stops (3.12). Otherwise, it goes to the next iteration (3.7).

$$\text{if} \quad \tilde{\boldsymbol{S}} = \boldsymbol{S} \quad \text{then} \quad \hat{\boldsymbol{E}} = \tilde{\boldsymbol{E}} \tag{3.12}$$

In Chapter 4 a performance comparison between this decoder and its binary version is presented. The latest is expected to be outperformed because the scenario introduced in this chapter exploits the correlation between the X and Z errors in the depolarizing channel. Next we aim at improving non-binary BP.

3.2 Non-Binary Belief Propagation with Ordered Statistics over $\mathbb{F}_2$

In this section, an approach similar to Section 2.4 is presented. In fact, the objective of this algorithm is to improve the performance of BP over $\mathbb{F}_4$ relying on an order-statistics decoding (OSD). This approach is similar to Algorithm 3 in [21], but instead of using the OSD as the post-processor, it is adopted in every iteration, as per the binary case presented in this thesis. Note that the OSD requires knowledge of the reliability of the symbols. From (3.11) we know that the hard decision over the symbol is based on the reliability vector calculated according to:

$$\begin{aligned} \boldsymbol{l}_j^{\text{total}} &= \boldsymbol{l}_j + \sum_{c_i \in N(v_j)} \boldsymbol{l}_{i\to j} \\ &= \begin{bmatrix} l_j^I & l_j^X & l_j^Y & l_j^Z \end{bmatrix} \end{aligned} \tag{3.13}$$

BP evaluates the l_j^{P} as, where $\text{P} \in \{I, X, Z, Y\}$

$$l_j^{\text{P}} = \log\left(\frac{p_{\text{P}}}{p_I}\right) \tag{3.14}$$

In Section 2.4 the absolute value of the LLR of the bits were used as reliability for ordering the sequence and the PCM. The elements of the non-binary LLR vector in (3.14) are normalized to the value of the identity operator. Hence, a comparison of two

symbol reliabilities (for sorting the vector) is an issue. Consequently, we compute the probabilities at node v_j is as

$$\tilde{P}_j^{\mathrm{P}} = \frac{\exp\left(l_j\right)}{\sum_P \exp\left(l_j^{\mathrm{P}}\right)}. \tag{3.15}$$

The maximum value among the elements of the vector is picked and its reliability is used for the ordering. The maximum value from above, for every variable in the graph, is stored in the following vector

$$\boldsymbol{p} = \begin{bmatrix} p_1 & p_2 & \dots & p_N \end{bmatrix}. \tag{3.16}$$

3.2.1 Ordering with $\mathbb{F}_4$ reliability

Chapter 2 has reported and explained, what has been the procedure for GE and why it has to be performed simultaneously with the syndrome. This is due to the fact that the same row operations applied to the PCM have to be also applied to the syndrome otherwise (2.23) is inconsistent with the creation of a list of codewords. In principle, applying GE to a non-binary system of equations is not a problem. However, the known term, i.e., the elements of the syndrome are obtained using the field trace operator, thus are binary. In other words, each equation puts a binary constraint (binary syndrome value) on the 4-ary symbols involved in the equation. Hence, their value cannot be uniquely resolved.

A remedy is to perform binary OSD based on the binary representation of the symbols and PCM. To do it, there are two methods the first is to store the 4-ary vector for each probability and then marginalize to obtain the probability of the X and Z symbols. For instance, the symbol A with a probability vector evaluated with (3.15) would lead to $P(A_X = 1) = \tilde{P}_A^X + \tilde{P}_A^Y$ and $P(A_Z = 1) = \tilde{P}_A^Z + \tilde{P}_A^Y$, with those is possible to order based on the bits. The second is to keep for the X and the Z part the same probability. Those two options, even if the first is more appropriate than the latest, lead to a similar ordering. In the following is reported the second approach leading to a binary representation of the PCM in the shape:

$$H = \begin{bmatrix} \boldsymbol{h}_{X1} & \boldsymbol{h}_{Z1} & \cdots & \boldsymbol{h}_{XN} & \boldsymbol{h}_{ZN} \end{bmatrix} \tag{3.17}$$

with $\boldsymbol{h}_{X_1} \in H_X$ and $\boldsymbol{h}_{Z_i} \in H_Z$ for $i = \{1, \dots, N\}$. The PCM of the stabilizer code in (1.21) in this configuration leads to

$$H = \left[\begin{array}{cccccccccccccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right] \tag{3.18}$$

This permutation combined with the sorting of the symbols is saved in a singular function named $\Lambda_1(.)$ to be applied to the column of the PCM in the form $H = [H_X|H_Z]$ and to the elements of the binary representation of the output of the BP $\boldsymbol{e} = [\boldsymbol{e}_Z|\boldsymbol{e}_X]$. In fact, when the permutation is applied to $\boldsymbol{e}$ the result is

$$\boldsymbol{\epsilon} = \Lambda_1(\boldsymbol{e}) = [\ \epsilon_{Z_1} \quad \epsilon_{X_1} \quad \epsilon_{Z_2} \quad \epsilon_{X_2} \quad \cdots \quad \epsilon_{Z_N} \quad \epsilon_{X_N}\ ]. \tag{3.19}$$

The reliability of the bit composing the symbols ordered in increasing order:

$$p_{\epsilon_{Z_1}} = \ p_{\epsilon_{X_1}} \leq \ p_{\epsilon_{Z_2}} = \ p_{\epsilon_{X_2}} \leq \ \cdots \ p_{\epsilon_{Z_N}} = \ p_{\epsilon_{X_N}} \tag{3.20}$$

Once the permutation for sorting the bits according to the reliability of the symbols has been defined, the remaining steps of the OSD are the same as the binary case.

3.2.2 Algorithm Flow-Chart

In the following the algorithm which combines BP over $\mathbb{F}_4$ and Ordered Statistic reprocessing in $\mathbb{F}_2$ is reported.

- **BP decoding** : Performed in $\mathbb{F}_4$ according to Section 3.1
  - Initialization: Evaluated the vector of LLRs for each symbol and send the scalar LLR to the check nodes 3.6.
  - CN update: compute the extrinsic information and send the vector message according to 3.8.
  - VN update: Process the incoming messages according to 3.9.
- **Early stopping criteria** : Save the reliability of the symbols evaluated in the previous step.
  - Compute the symbols according to 3.11 and express $\tilde{\boldsymbol{E}}$ in binary form ($\tilde{\boldsymbol{e}}$).
  - if $H\tilde{\boldsymbol{e}}^T = \boldsymbol{S}$ then the decoding algorithm halts and $\tilde{\boldsymbol{e}}$ is considered a valid decoding result:
    $$\hat{\boldsymbol{e}} = \tilde{\boldsymbol{e}}$$
  - Else, the algorithm starts the "Order-ℓ Reprocessing".
- **OSD decoding**: Based on the reliability values 3.16, sort the bits of $\tilde{\boldsymbol{e}}$ and the binary parity-check matrix (with increasing order) and determine its h least reliable independent columns. Those permutations are named $\Lambda_1(.)$ (3.19) and $\mu_1(.)$, respectively.
  - Apply the same permutations to $\tilde{\boldsymbol{e}}$
    $$\boldsymbol{e}' = \mu_1(\Lambda_1(\tilde{\boldsymbol{e}})) \tag{3.21}$$

  - For $0 \leq i \leq \ell$ make all possible $\binom{n-h}{i}$ changes of i bits in the MRP and for each change, determine the corresponding vector $\boldsymbol{e}_i$ according to 2.23.

- **Decision**:
  - Among the candidates, select the one with minimum Hamming weight ($\boldsymbol{e}^*$).
  - if $\boldsymbol{e}^* = \hat{\boldsymbol{e}}$ the test is satisfied and ${\Lambda_1}^{-1}\left({\mu_1}^{-1}(\hat{\boldsymbol{e}})\right)$ is selected as the decoded vector.
  - Else, if the Hamming weight of $\boldsymbol{e}^*$ is lower than $\hat{\boldsymbol{e}}$, set the former as the decision at previous step

$$\text{If } w(\boldsymbol{e}^*) \leq w(\hat{\boldsymbol{e}}) \text{ then } \hat{\boldsymbol{e}} = \boldsymbol{e}^* \tag{3.22}$$

    Start the next iteration with BP decoding.

Similarly to the procedure presented in the previous chapter, the test or the early stopping criteria are the requirements to achieve in order to stop the decoder and provide a decision. If they are not met then the algorithm stops when the maximum number of iterations is reached. This algorithm improves the performances with respect to BP thanks to the use of the reliability of the symbols. It also improves with respect to its binary version since the correlation between X and Z is exploited, at least in BP. Regarding the complexity, analogously to the algorithm presented in Section 2.4, GE has to be performed at every iteration to find the permutation μ_1, and for this reason, the algorithm loses in speed and gains in complexity. A possible solution will be presented in the next section.

3.3 Non-Binary Belief Propagation and Information Set Decoding over $\mathbb{F}_2$

In Section 2.4 and Section 3.2 two decoding schemes which employ the combination of BP and OSD have been investigated, in order to bridge the gap between BP and maximum likelihood decoding. In the flow-charts of the two algorithms, it is reported that the search for the independent columns of the PCM, with the consequent modification of the latest to the form:

$$H' = \begin{bmatrix} I_m & \boldsymbol{h}'_{m+1} & \cdots & \boldsymbol{h}'_n \end{bmatrix} \tag{3.23}$$

is performed at every iteration. This modification implies the use of GE (2.2.1). From the literature, it is given that such an algorithm has a complexity that scales with the cube of the dimension of the matrix [23]. In the case under study, the complexity of GE is $\mathcal{O}(n^3)$. It is obvious that adding this step at every iteration increases the complexity of the whole algorithm. For this reason, in this section, a different approach is presented, with the same goal as the previous decoders but less complex. Differently from the previous algorithms in this case the statistics of the symbols will not be used in the decoder. Instead, GE is performed only once before the beginning of the decoding algorithm. The

matrix H in the form identity (see (3.23)) is given to the decoder, together with the permutation of the columns ($\mu_1(.)$) and a function which is able to generate a version of the detected syndrome consistent with the permutations applied by the GE, named Φ.

The syndrome $\boldsymbol{S}'$ allows to perform $H'[\boldsymbol{e}'_u, \boldsymbol{e}'_k]^T = \boldsymbol{S}'$. Then, you can split the equation as $\boldsymbol{e}_u = B\boldsymbol{e}_k^T + \boldsymbol{S}'$. This way, instead of having a complexity of $\mathcal{O}(n^3)$ to obtain the same result in terms of matrices and vectors, only a matrix multiplication is needed, whose complexity is $\mathcal{O}(n^2)$. This method has the advantage of performing GE once and not at every iteration.
The list of candidates is created starting from the binary version of the hard-decision of the BP with the use of ISD and BF. The algorithm is fully described by the flow-chart reported below:

- **GE and Syndrome modification** : Performed according to Section 2.2.1 and before starting decoding. Then pass to the actual decoder the following results:

$$H' \quad , \quad \boldsymbol{S}' = \Phi * \boldsymbol{S} \quad , \quad \mu_1 \tag{3.24}$$

  together with the actual Parity-check matrix H and the actual Syndrome $\boldsymbol{S}$.

- **BP decoding** : Performed over $\mathbb{F}_4$ according to Section 3.1 with H and S.
  - Initialization: Evaluated the vector of LLRs for each symbol and send the scalar LLR to the check nodes 3.6.
  - CN update: compute the extrinsic information and send the vector message according to 3.8.
  - VN update: Process the incoming messages according to 3.9.

- **Early stopping criteria** :
  - Compute the symbols according to 3.11 and express $\tilde{\boldsymbol{E}}$ in binary form ($\tilde{\boldsymbol{e}}$).
  - if $H\tilde{\boldsymbol{e}}^T = \boldsymbol{S}$ then the decoding algorithm halts and $\tilde{\boldsymbol{e}}$ is considered a valid decoding result:
$$\hat{\boldsymbol{e}} = \tilde{\boldsymbol{e}}$$
  - Else, the algorithm starts the "Order-ℓ Reprocessing".

- **Information Set decoding and Bit-Flipping**: Performed in $\mathbb{F}_2$ according to Section 2.2.
  - Rearrange the bits of $\tilde{\boldsymbol{e}}$ according to $\mu_1(.)$ evaluated at the beginning 3.24, in order to full-fill the equation:
$$H' * \boldsymbol{e}' = H' * \mu_1(\tilde{\boldsymbol{e}}) = \boldsymbol{S}' \tag{3.25}$$
  - For $0 \leq i \leq \ell$ make all possible $\binom{n-h}{i}$ changes of i bits in the MRP and for each change, determine the corresponding vector $\boldsymbol{e}_i$ according to 2.23.

- **Decision**:
  - Among the candidates selects the one with minimum Hamming weight ($\boldsymbol{e}^*$).
  - if $\boldsymbol{e}^* = \hat{\boldsymbol{e}}$ the test is satisfied and ${\mu_1}^{-1}(\hat{\boldsymbol{e}})$ is select as the decoded vector.
  - Else, if the Hamming weight of $\boldsymbol{e}^*$ is lower than $\hat{\boldsymbol{e}}$ set the first as the decision at previous step

$$\text{If } w_H(\boldsymbol{e}^*) \leq w_H(\hat{\boldsymbol{e}}) \text{ then } \hat{\boldsymbol{e}} = \boldsymbol{e}^* \tag{3.26}$$

  Start the next iteration with BP decoding.

This implementation saves in complexity, but as it will be shown in the next chapter, not taking into account the statistics of the symbol will negatively affect the performance with respect to the algorithm presented in the previous section.

3.4 Correction of Y-Errors

In every combined decoder presented in this thesis the algorithm in charge of the reprocessing of the output of the BP acts in a binary fashion. In fact, the reprocessing, hence the creation of the list of candidates, is applied over the PCM and the sequence, respectively, in the form

$$\begin{array}{rcccc} H & = & [H_X & | & H_Z] \\ \tilde{\boldsymbol{e}} & = & [\tilde{\boldsymbol{e}}_Z & | & \tilde{\boldsymbol{e}}_X] \end{array} \tag{3.27}$$

Unfortunately, this method does not consider the correlation between the X and Z components, or more precisely does not support a quaternary symbol-flipping. In this section, different approaches are presented with the aim of counteracting this loss by exploiting the correlation between the X and the Z components also in the reprocessing. To better explain this effect in the decoder an example of the order-1 reprocessing of a given sequence is reported. In Section 2.2 the reprocessing has been defined as the flip of a single bit in the information set. Consequently, when both elements of the of $e_{i,Z}$ and $e_{i,X}$ are in the information a single bit-flip only corrects X or Z errors, but not Y errors which would require two flips. Consider the PCM in (2.11) in identity form with column permutation given by μ_1,

$$H' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.28}$$

$$\mu_1 = \begin{bmatrix} 1 & 2 & 3 & 9 & 5 & 10 & 11 & 12 & 4 & 6 & 7 & 8 & 13 & 14 \end{bmatrix}$$

And consider the error pattern generated by the channel:

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & Y & 0 \end{bmatrix} \tag{3.29}$$

Which may be express with with $e_i \in \mathbb{F}_2$ as:

$$\boldsymbol{e} = \left[\begin{array}{ccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \right]. \tag{3.30}$$

The syndrome before and after the GE is

$$\boldsymbol{S} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad \boldsymbol{S}' = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.31}$$

. Assume that the error pattern received from the BP is the all-zero sequence $\tilde{E}_{\text{zero}}$. In this case the syndrome would not match the one detected at the receiver and the decoder would move to the "order-ℓ reprocessing". Then the bit flipping algorithm performed according to Section 2.2 would produce the following list of candidates:

$$e_{\text{cand}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.32}$$

In this scenario, the decision based on the Hamming weights of the sequences would select randomly between the two sequences highlighted in red. It is obvious that none of them is the transmitted sequence, even if they are not permuted back, since both of the sequences have four ones instead of two. In this case, the bit-flipping in fact acted on the couple Z_6 X_6 by flipping either Z_6 or X_6. This means that the BP decoded a sequence presented $\tilde{E}_6 = 0$ and the bit flipping investigated only the cases: $\tilde{E}_6 = X$ and $\tilde{E}_6 = Z$ not taking into account Y and leading to a wrong decision. Of course the reprocessing has to achieve the same result twice before making any decision but not considering one of the three possible scenarios may lead to a degradation in performance. In the following, different solutions are proposed.

3.4.1 Symbol Flipping

The first solution proposed in this section is to emulate a symbol-flipping while remaining in $\mathbb{F}_2$. While GE is performed identically to the operation performed in $\mathbb{F}_2$, in this scenario

the reprocessing, in addition to the candidates evaluated in order-ℓ reprocessing, also adds the candidates which account for the errors of type-Y. The PCM after the GE is given in the form (2.16) in which is not ensured that the columns referring to the left side of the original matrix ($\boldsymbol{h}_{x_i}$) are exclusively on the left and the same hold for the right-part. The new positions of the original columns are in given in (2.20). For order-ℓ reprocessing let

$$\boldsymbol{e}_1 = \begin{pmatrix} e_1 & e_2 & \cdots & e_h & 0 & \cdots & 0 \end{pmatrix} \tag{3.33}$$

be the starting point of the reprocessing and create the first

$$\sum_{i=0}^{\ell} \binom{n-h}{i} \tag{3.34}$$

candidates according to 2.2.2. Due to the fact that we are working with a binary PCM, half of the columns are associated to X-components and the other half are Z-components. Given $\mu_1(.)$ the permutation among the column, the algorithm saves the positions of the elements which constitute a pair $X - Z$ in the unmodified matrix. Those pairs are stored in a matrix named P_Y, which appears as:

$$P_Y = \begin{pmatrix} p_{x1} & p_{x2} & \cdots & p_{xm'} \\ p_{z1} & p_{z2} & \cdots & p_{zm'} \end{pmatrix} \tag{3.35}$$

With m' the number of pairs whose elements lay both in the $n-h$ known positions of the matrix, hence the positions eligible for a bit-flip. For $1 \leq i \leq \ell$ in addition to the changes already introduced, the reprocessing function makes all possible changes in the positions described by the columns of 3.35. This increase the number of candidates to

$$\sum_{i=0}^{\ell} \binom{n-h+m'}{i} \tag{3.36}$$

and enable the chances to account for the Y-errors in the list of candidates.
An Example is proposed to show the behaviour of this particular reprocessing. Consider the matrix H' in 2.20, and the permutation μ_1 introduced by the GE, which stores the new positions of the bits. In this case, $N = 7$ is the size of H_X and H_Z. Two elements of μ_1, μ_{1_i} and μ_{1_j} constitute a pair only if $\mu_{1_i} = \mu_{1_j} + N = \mu_{1_j} + 6$. Hence the positions of these elements are stored in P_Y according to:

$$[i,j]^T \in P_Y \iff i,j \in B_{n-h} \tag{3.37}$$

Where $I(B_{n-h})$ is the set of elements belonging to the information set. Below elements of the considered $\mu_1(.)$ which satisfies 3.37 are highlighted.

$$\mu_1 = \begin{bmatrix} 1 & 2 & 3 & 9 & 5 & 10 & 11 & 12 & 4 & 6 & 7 & 8 & 13 & 14 \end{bmatrix} \tag{3.38}$$

In this case, the matrix accounting XZ pairs presents two columns representing these positions and the total number of positions to be considered in the $order-1$ reprocessing

increases by two. Consequently, the encoding is performed, according to 2.23, to the two vectors:

$$\boldsymbol{e}^8 = \left[\begin{array}{cccccccccccccc} e_1^8 & e_2^8 & e_3^8 & e_4^8 & e_5^8 & e_6^8 & e_7^8 & e_8^8 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}\right]$$

$$\boldsymbol{e}^9 = \left[\begin{array}{cccccccccccccc} e_1^9 & e_2^9 & e_3^9 & e_4^9 & e_5^9 & e_6^9 & e_7^9 & e_8^9 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

Assuming the error pattern introduced by the channel in 3.29, the $order-1$ reprocessing accounting also for those two new positions would lead to two further candidates. Hence the list of candidates would become:

$$e_{\mathsf{cand}} = \left[\begin{array}{cccccccccccccc} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \tag{3.39}$$

Differently from 3.32 in this set, there is only one sequence with minimum Hamming weight and appears to be the sequence highlighted in blue. Consequently, the decoder decides for that and if the inverse of the permutation (μ_1^{-1}) is applied to the decision, the output of the decoder coincides with the Error pattern generated by the channel.

3.4.2 Modified Gaussian Elimination

In the previous section, a different type of reprocessing has been proposed to cope with the degradation introduced by the error of type Y. In this section, a different approach is presented. While before the idea was to increase the number of candidates with the aim of including also the Y-flips, in this section the construction of the parity-check matrix in standard form is investigated. The objective of this modified GE is to keep the elements of the same pair $X - Z$ separate: one in the information set and the other in the complementary set. This way the bit-flipping does not have to account for the flip over the couples cause there are none in the information set. Generally, the algorithm aims to achieve the following:

$$\begin{array}{lllll} \forall i \in B_{n-h} & \text{if} & \mu_i > N & \text{then} & j : \mu_j = \mu_i + N, \in \bar{B}_{n-h} \\ & \text{if} & \mu_i < N & \text{then} & j : \mu_j = \mu_i - N, \in \bar{B}_{n-h} \end{array} \tag{3.40}$$

The difference with 2.2.1 lies on the columns switch. In fact, when the i-th column, with $h_i \in H_X$, does not have ones, it was set to be exchanged with the column in position $(m+1)^{th}$. In the function proposed, the decision is to exchange it with the column in position $(N+i)^{th}$ column. Where $(N+i)^{th}$ is the position of the Z-element of the couple XZ. For big matrices, the proposed GE reduces the overall number of pairs in the rightmost positions of the PCM in standard form, leading to a small increase of performance.

Of course, this type of GE is more complex than the one proposed in the previous chapter, but for the algorithm presented in Section 3.3 it only has to be performed once, hence may be possible to see how this improve the performance. On the other hand, this type of GE may not be applied to the Ordered-Statistic decoder. The reason is that the ordering imposed by reliability that group the pairs together. Hence, applying this type of GE would vanish the gain introduced by the knowledge of statistics. The following is reported as an Example which shows how the seven-cubit CSS code would appear in standard form if this method is applied.

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{3.41}$$

In this case, the fourth column, which is the one that does not present ones below the index, is exchanged with the eleventh, both highlighted in 3.41. The resulting matrix and relative columns permutation are:

$$H' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.42}$$

$$\mu_1 = \begin{bmatrix} 1 & 2 & 3 & 11 & 5 & 13 & 14 & 9 & 8 & 10 & 4 & 12 & 6 & 7 \end{bmatrix}$$

In this case, there are no couples in the information set. The indices of the couples that were on the right-most side are one on two different sides of the matrix. This, unfortunately, does not work for every matrix. If the code considered during the simulations is adopted in their binary representation, the PCM is of dimensions mxn with $m \leq n/2$. Hence, this procedure would always lead to having some couple remaining in the set B_{n-h}. Consequently, this method succeeds in reducing the number of couples in the information set, but it does not completely eliminate them.

Chapter 4

Numerical Results

4.1 Toric Codes

Toric codes [24] are a class of topological stabilizer codes defined over a two dimensional lattice, that takes the shape of a torus. We consider a few short toric codes examples in order to get a first insight on the performance of the proposed decoding algorithms. A toric code can be described by the parameter L where the number of physical bits $N = L^2$ and $K = 1$.

Our first experiment is a comparison between the binary BP decoder and the modified non-binary BP decoder over a depolarizing channel. The non-binary decoder is expected to outperform binary BP in terms of error rate, since at the VNs 4-ary probabilities (or better LLRs) are processed. Hence, the correlation between X and Z errors is accounted by the non-binary decoder. For the experiment we consider a short toric code with $L = 4$. In Figure 4.1 the frame error rate (FER) versus the error probability p of the depolarizing channel is shown. It is visible that non-binary BP outperforms its binary version. This comes at a cost of enhanced processing at the VNs which mainly consists of a marginalization step in (3.10) and of addition of 4-ary vectors in (3.9). The latter one can be simplified, noting that each 4-ary vector (CN message) has only two non-zero elements and not four.

In the same figure, the performance of non-binary BP with ISD and OSD (with bit flipping) is also reported for different orders of reprocessing. The results show that both of them gain in performance with respect to the pure BP, where the gain increases with the order of reprocessing. This can be attributed to the fact that the former employs the statistics of the bits while the latter does not. Interestingly, for the considered order of reprocessing, ISD and OSD perform similarly. The additional cost of the ISD consists in the re-encoding operation of the candidate error patterns. The complexity of this operation is driven by a matrix-vector multiplication. The cost is in general quadratic in n. In case the encoding matrix is sparse or structured simplifications are possible. OSD comes with the additional complexity of GE, since the information set may change depending on the output of the BP decoder in every iteration. The cost here is cubic in n.

Overall it can be stated that (at least) for short codes non-binary BP with ISD is preferred to non-binary BP with OSD due to the complexity savings and similar performance.

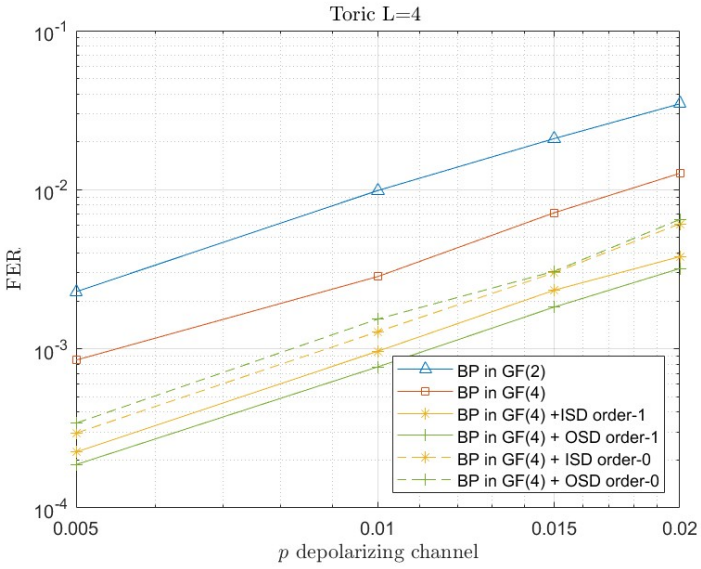


Figure 4.1: Comparison belief propagation in $\mathbb{F}_2$, pure belief propagation in $\mathbb{F}_4$ and BP in $\mathbb{F}_4$ with information-set decoding and with order-statistics decoding both with standard GE and BF

Figure 4.2 shows the FERs under non-binary BP and ISD for higher orders of reprocessing. In fact, for this toy example the algorithm reaches its best performance already with order-2. This is due to the fact that the code adopted is very short. Consequently, the correct codeword is already in the list when two bit-flips are performed. This last statement does not hold for longer codes, as it will be shown later, in which order-2 is not be enough to have the correct codeword in the list of candidates.

Next we investigate how the correlation between X and Z errors affects the performance of ISD only. To this end, we employ pairwise flips of those bits which jointly represent a 4-ary symbol as described in Section 3.4.1. We also refer to this as symbol flipping (SF). Since there is not the error pattern coming from the iteration of the BP the sequence of all-zero has been adopted as input of the ISD. For this study the channel has been modified in order to obtain only two errors of the same type. The outcome is reported in Figure 4.3. Two different decoders have been implemented both of them of order-2. Firstly, BF and secondly BF. The curve accounting for a SF decoder for the two Y errors is not reported because the two Y errors introduced by the channel are always corrected by the decoder. The same does not hold for the ISD with bit-flips and it is clear that in this case, the performance is worse than the channel which introduces two X or Z errors. For this toy example SF pays off in terms of performance. However

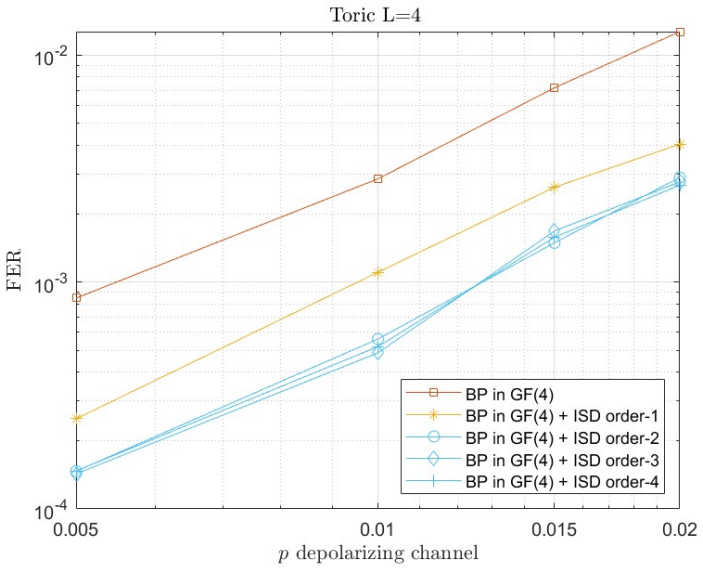


Figure 4.2: Comparison belief propagation in $\mathbb{F}_4$ with information-set decoding a nd BF at different orders

decoding complexity increases: now for order ℓ reprocessing all pairs of bits belonging to the same symbol (in the information set) need to be also flipped. For the ISD only there are $m' = 12$ couples in the information set, hence the number of candidates change from 596 to 1082 for order-2 according to eq.(3.36).

Figure 4.4 considers the performance of non-binary BP wit ISD and OSD. For the reprocessing we assume first BF and later SF for both the decoder. When SF is applied to the OSD the pairs are searched among the indices after the ordering introduced by the reliability and GE, those positions changes every iterations, but generally are more than the number of positions for ISD. In facr, for ISD the search for pair is performed over the indices reordered according to the permutation introduced by GE only, meaning that the positions of the pairs X and Z are always the same. The curves are very close to the ones employing reprocessing with BF. For this reason, SF in the reprocessing is not suggested for short-codes and the result also suggest that the X and Z parts may be treated independently. In the figure is also reported a curve for ISD employing the modified GE in this case the curve is lower than the one which employs standard GE, meaning that the ordering imposed by GE affect the performance more than the reprocessing with SF, for short codes.

Toric codes with larger dimensions have also been investigated. In Figure 4.5 the results corresponding to the toric code with $L = 7$ are shown. Similarly to the previous code, the curves relative to the combined algorithms show a gain with respect to pure BP. The adoption of statistics in the creation of the list of candidates (OSD) again outperforms the simpler algorithm based on ISD.

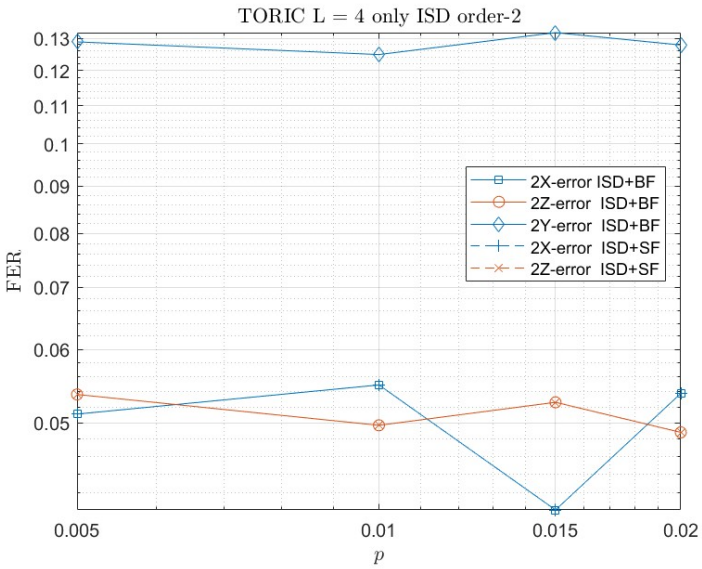


Figure 4.3: Information set decoding of order-2 on modified channel introducing exactly two 2 errors of the same type under BF and SF

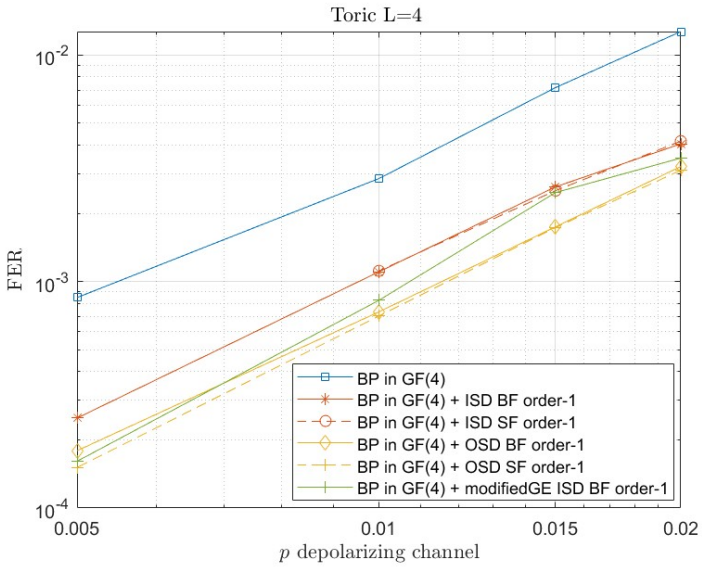


Figure 4.4: Comparison BP + ISD and BP + OSD with BF and SF reprocessing

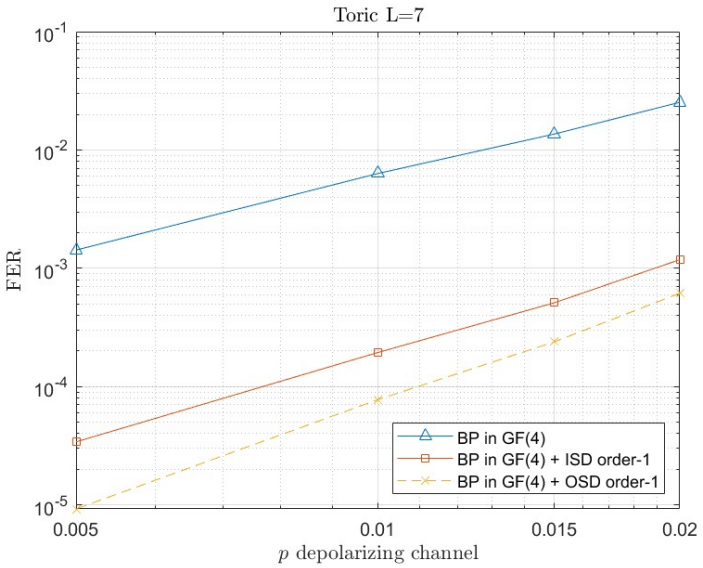


Figure 4.5: Comparison non-binary BP with information-set decoding and with order-statistics decoding both with BF reprocessing

Moreover, these results show that BP is less effective in this case compared to toric code with $L = 4$, and for this reason, the gain achieved thanks to further algorithms combined with it is larger. Overall, ISD and OSD perform close to each other also in this case.

4.2 Hyper-Graph Product Codes

We report results for some hyper-graph product codes (HPCs), a well-known family of quantum codes [25]. The code adopted for the simulations has dimensions $[550, 169]$ and has been constructed according to [22]. Its PCM is reported in the Appendix. In Figure 4.6 the results of pure BP and of the algorithms which adopt ISD and OSD to correct the guess of the BP are reported. Compared to the Toric code with $L = 7$, the gain over the pure BP is smaller. Moreover, the graph shows that for OSD with order-1 reprocessing the employment of the statistics of the symbols does not yield visible gains compared to ISD. For this reason, when the hyper-graph product code (HPC)s are adopted is suggested to use higher order of reprocessing and to stay on the ISD to correct the guess of the BP in order to save complexity.

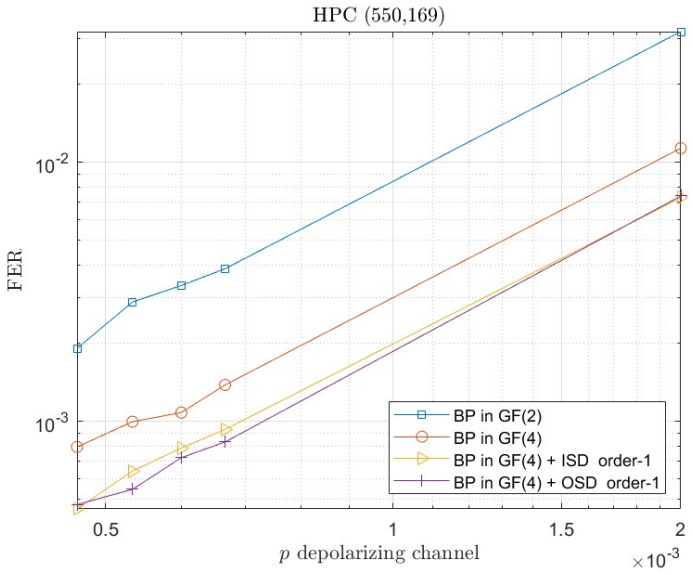


Figure 4.6: Hyper-graph Product Code $(550, 169)$ order-1 algorithms with BF reprocessing

4.3 Single Parity-Check Product Code

This section investigates the performance of quantum CSS single parity-check product codes (SPCPs). These codes were designed to achieve good performance with BP. We check whether the proposed combined algorithms can improve this result. The code adopted in the simulations is specified in [22].

The code adopted is a two-dimensional product code with parameters $[[32, 64, 4]]$ which is labelled by SPCP $D = 2$. Simulation results for non-binary BP with ISD and different orders of reprocessing (BF) are shown in Figure 4.7. When the reprocessing order is small, e.g., $\ell \in \{0, 1\}$, the gain is minor much with respect to pure BP.

Additionally, we introduce a benchmark to assess the performance of a genie-aided decoder. To this end, we put the actual error pattern (introduced by the channel) on list of candidate error patterns (if not there already). This would yield a lower bound on the performance of an ML decoder - in absence of degeneracy. We claim that for low error probabilities this benchmark converges to a lower bound. We argue as follows: when p is low, the channel error pattern will have low weight. It is improbable, that we find another error pattern in the same coset with lower weight. Assume that lowest weight error pattern of this coset (to which also the channel error pattern belongs) is on the list of candidate error patterns. Since the list is small compared to the full list of an ML decoder (i.e., there are less competitors), an ML decoder will make more errors than our genie-aided decoder. Observe that when the list size grows, the lower bound will become tighter.

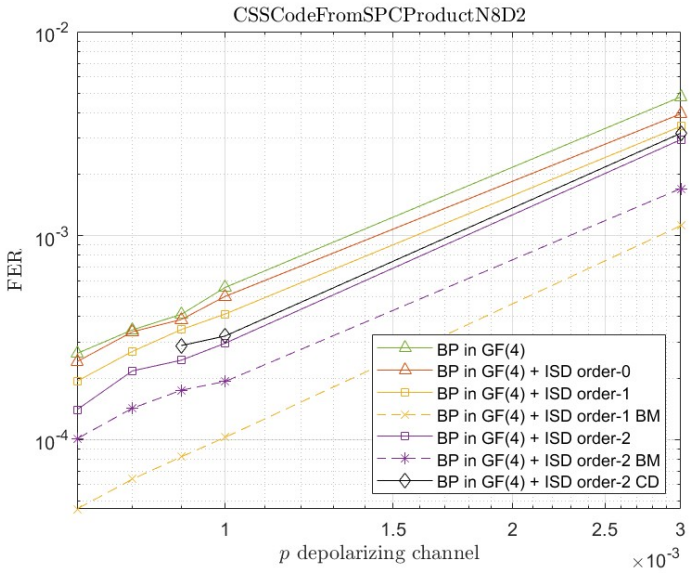


Figure 4.7: CSS code and D=2 with benchmark (BM) and coset decoding (CD) rule over ISD + BF reprocessing

For $L = 1$ the benchmark in Figure 4.7 is far from the actual decoding performance. For $L = 2$ the gap is small, suggesting that - at least for small error probabilities for which the benchmark will correspond to a lower bound - we operate close to the maximum-likelihood performance. Due to degeneracy, the maximum-likelihood decoding rule is not optimal, though. To this end, we consider a coset decoder described in Section 1.9. For this all candidate error patterns on the list are grouped into cosets whose probability is determined. Note that the list of candidate error patterns is small compared to all 2^n error patterns. Hence, the optimal coset decoding rule cannot be implemented.

Simulation results with the proposed coset decoding rule show a tiny loss in performance w.r.t. the maximum-likelihood decoding rule that is adopted to select the error pattern from the list. The reason behind is that the number of candidates in the list is not big enough.

Conclusions

In this thesis, multiple decoding schemes for quantum low-density parity-check (QLDPC) codes have been presented. Firstly, the binary version of BP has been investigated because of the gain in performance iterative decoding has shown through the years in the decoding process of sparse matrices. The study of QEC, on the other hand, imposes stringent constraints on the construction and on the decoding. Constraints which guided this thesis towards algorithms which combine the iterative approach with the reprocessing based on the statistics of the bits. To further embody the characteristics of quantum communication, the search for decoders has been shifted towards non-binary implementations. It has been shown how going from $\mathbb{F}_2$ to $\mathbb{F}_4$ presents some difficulties in handling decoding schemes which employ the syndrome, which for quantum codes is necessarily binary. In order to cope with the complexity introduced by the reprocessing performed with OSD we proposed another algorithm based on ISD. Finally, to further explore the correlation among the Pauli operators a function aiming to reprocess a binary codeword over the symbols has been presented.

The results showed that for short-codes, e.g., the toric L=4, a combination of the belief propagation in $\mathbb{F}_4$ and information-set decoding with bit-flipping of order-2 lead to a significant gain in performance with respect to the pure BP and saves in complexity with respect to the use of BP combined with OSD. The same holds for larger toric codes. Moreover, the SF reprocessing of a list of binary codewords does not show any improvement in performance with respect to the reprocessing over the bits. For hyper-graph product codes it is suggested to adopt ISD with a number $\ell > 1$ of bit-flips to correct the guess of the BP because the use of statistics of the symbols do not guarantee better performance, but an increase in complexity. Lastly, for the CSS code based on single parity-check product (SPCP) codes, we confirmed that the iterative approach has good performance on this type of code. However, it may be suggested to adopt a reprocessing of order ℓ greater than one to meaningfully lower the curves of error. Generally, it can be extracted that the optimal decoding rule based on cosets should be applied only when there are enough samples to properly weight the accumulated probabilities and order-1 does not guarantee that.

From this thesis, we concluded that exploiting the construction of a quantum code improves the performance of a decoding scheme, however, if the decision is based on a list of candidates, in order to apply some constrain over this list, it should be ensured that the list is big enough to be a faithful representation of how the cosets appear. Further

improvement may be explored by changing the construction of the list, or by changing the decision taking into account the role degeneracy plays inside the decoder. The field of quantum error correction is far to be standardized and much research may be further implemented with the goal of making this technology accessible in the near future.

Appendix

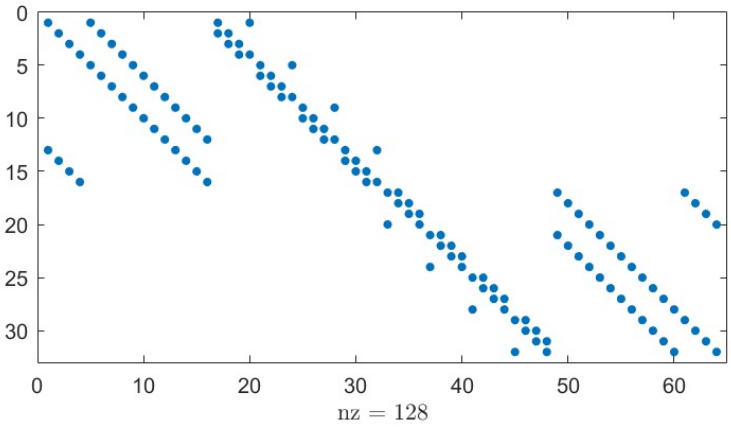


Figure 4.8: parity-check matrix of toric $L = 4$ code

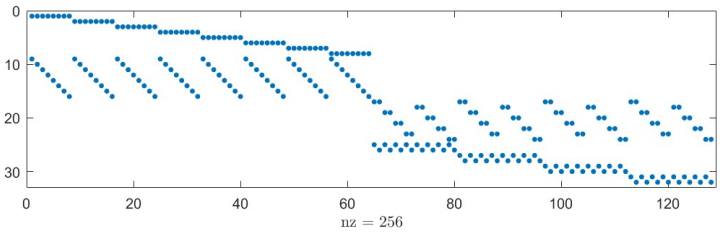


Figure 4.9: parity-check matrix of CSS code from SPCP code $[[32, 64, 4]]$ with $D = 2$

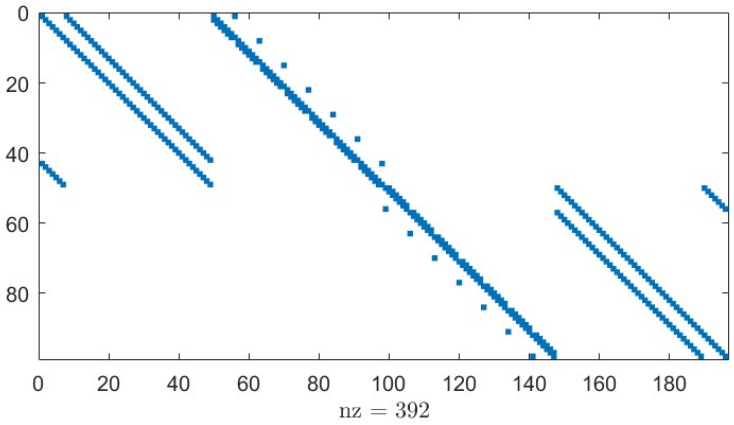


Figure 4.10: parity-check matrix of toric $L = 7$ code

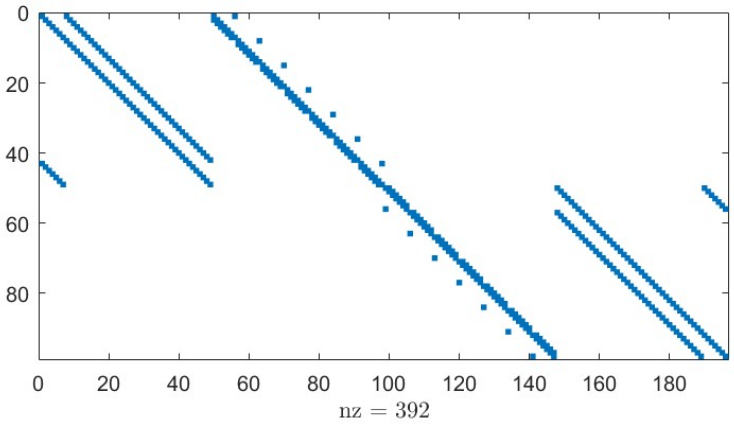


Figure 4.11: parity-check matrix of toric of Hyper-graph Product Code $(550, 169)$

Bibliography

[1] A. Beniza, "Industry applications of quantum computing." https://www.computer.org/publications/tech-news/research/industry-applications-of-quantum-computing?source=cedge&vgo_ee=1OXwrBnzQGtw2Nx1eEEqxETK%2Fa2e%2FoVZdGq4rYLmMaA%3D, 27/01/2022. [Online; accessed 18-November-2022].

[2] M. J. Biercuk and T. M. Stace, "Quantum error correction: Time to make it work." https://spectrum.ieee.org/quantum-error-correction, 26/06/2022. [Online; accessed 18-November-2022].

[3] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, pp. R2493–R2496, Oct 1995.

[4] H. Lo, T. Spiller, and S. Popescu, *Introduction to Quantum Computation and Information*. World Scientific, 1998.

[5] J. Preskill, "Lecture notes in physics 229, chapter 7." http://theory.caltech.edu/~preskill/ph229/, 1999. [Online; accessed 14-November-2022].

[6] D. Gottesman, "Stabilizer codes and quantum error correction," 1997.

[7] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, pp. 1098–1105, aug 1996.

[8] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009.

[9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[10] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum ldpc coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.

[11] D. MacKay, G. Mitchison, and P. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Transactions on Information Theory*, vol. 50, pp. 2315–2330, oct 2004.

[12] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo, and J. Garcia-Frías, "Degeneracy and its impact on the decoding of sparse quantum codes," *IEEE Access*, vol. 9, pp. 89093–89119, 2021.

[13] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[14] D. MacKay, G. Mitchison, and P. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2315–2330, 2004.

[15] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[16] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962.

[17] B. Dorsch, "A decoding algorithm for binary block codes andj-ary output channels (corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 391–394, 1974.

[18] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, 1995.

[19] M. Fossorier, S. Lin, and J. Snyders, "Reliability-based syndrome decoding of linear block codes," *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 388–398, 1998.

[20] M. Fossorier, "Iterative reliability-based decoding of low-density parity check codes," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 908–917, 2001.

[21] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," *Quantum*, vol. 5, p. 585, nov 2021.

[22] D. Ostrev, D. Orsucci, F. Lázaro, and B. Matuz, "Classical product code constructions for quantum calderbank-shor-steane codes," 2022.

[23] R. W. Farebrother, *Linear least squares computations*. Routledge, 2018.

[24] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, no. 1, pp. 2–30, 2003.

[25] J.-P. Tillich and G. Zémor, "Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 1193–1202, 2013.