Modified Belief Propagation decoders applied to non-CSS QLDGM codes

PROYECTO

presentado para optar

al Título de Grado en Ingeniería en Sistemas de Telecomunicación por

Borja Aizpurua Altuna

bajo la supervisión de

Pedro Crespo Bofill

Donostia-San Sebastián, junio 2021

tecnun
Universidad de Navarra

tecnun Universidad de Navarra

End of Grade Project

ENGINEERING IN TELECOMMUNICATION SYSTEMS

Modified belief propagation decoders applied to non-CSS QLDGM codes

Borja Aizpurua Altuna

Donostia-San Sebastián, June de 2021

Pº Manuel Lardizabal, 13. 20018 Donostia-San Sebastián, Gipuzkoa
Tel. 943 219 877 · Fax 943 311 442 · www.tecnun.es

Abstract

Quantum technology is becoming increasingly popular, and big companies are starting to invest huge amounts of money to ensure they do not get left behind in this technological race. Presently, qubits and operational quantum channels may be thought of as far-fetched ideas, but in the future, quantum computing will be of critical importance.

In this project, it is provided a concise overview of the basics of coding theory and how they can be used in the design of quantum computers. Specifically, Low Density Parity Check (LDPC) codes are focused, as they can be integrated within the stabilizer construction to build effective quantum codes. Following this, it is introduced the specifics of the quantum paradigm and present the most common family of quantum codes: stabilizer codes.

Finally, it is explained the codes that have been used in this project, discussing what type of code they are and how they are designed. In this last section, it is also presented the ultimate goal of the project: using modified belief propagation decoders that had previously been tested for QLDPCs, for the proposed non-CSS QLDGM codes of this project.

Acknowledgements

I remember myself one or two years ago when I didn't know what type of end of grade project I could do, and I remember that I knew that I wanted to do something related with the systems branch rather than the electronics, but I had no ideas. Fortunately, I play ice hockey with an incredible colleague called Patricio Fuentes (everyone calls him Patrick), who was starting his PhD in quantum computing and he started to tell me once a week or once every two weeks how he was evolving on his research and how interesting it was.

Thus, I started to feel some interest in quantum computing and started looking at some papers. Soon, I realized it could be a difficult but challenging possible end of grade project, and I love challenges, so I started to take it seriously. Without Patrick I would probably have done something related to communications but nor by far as interesting as this has been. He started recommending me more papers to look at, and some videos I could watch, and I started in summer to get the necessary backup to understand what he was doing. Through the year I have been all the time in touch and I want to thank him for spending time resolving my doubts and helping me understand, as it would have been impossible to make such a project without his help.

Apart from the incredible help from Patrick, I want to thank my teacher Pedro Crespo who introduced me to coding theory in class and then encouraged me to start the project with Patrick. Finally, I would like to thank Josu Etxezarreta and Imanol Granada who are working on their PhD with Patrick, and many of their papers have helped me to develop this project.

Index

Figure Index

List of Tables

Abbreviations

QEC	Quantum Error Correction
QSC	Quantum Stabilizer Codes
QLDPC	Quantum Low Density Parity Check
QLDGM	Quantum Low Density Generator Matrix
PCM	Parity Check Matrix
BP	Belief Propagation
SPA	Sum Product Algorithm
CSS	Calderbank-Shor-Steane
GF	Galois Field
QBER	Quantum Bit Error Rate
FER	Frame Error Rate
MAP	Maximum A Posteriori
APP	A Posteriori Probability
ML	Maximum Likelihood
DCM	Discrete Memoryless Channel
BSC	Binary Symmetric Channel
AWGN	Additive White Gaussian Noise
EFB	Enhanced Feedback
OSD	Ordered Statistics Decoder

1. INTRODUCTION

Since 1980 when physicist Paul Benioff proposed a quantum mechanical model[1] of the Turing machine [1] and then Richard P. Feynman proposed the idea of a Quantum Computer in his paper in 1982 [2], the field of quantum technology has become increasingly popular among researchers. One of the main disadvantages classical computing has is that discrete algorithms calculation as well as large numbers factorization into prime numbers are computationally intractable. Thus, RSA or Diffie-Hellman public-key cryptography are secure against a classical computer, whereas a quantum computer with sufficient qubits would be enough to break them.

In recent years, giant companies like Google, IBM and government agencies like NASA or the European Union have invested more than 1 billion euros trying to develop fault-tolerant quantum computers. One of the main problems researchers have faced is errors in qubits due to interaction with their environment, the unavoidable phenomenon known as quantum decoherence. The imperfect implementation of quantum gates, which are responsible of the corresponding quantum logical operations, also cause errors, and that's why Quantum Error Correction (QEC) has become a key subject of research in the quantum computing paradigm.

Fault-tolerant quantum computing refers to the idea of quantum devices working effectively even if their elementary components are imperfect. This, along with error correction must be the base of quantum computing if reliable communications are desired, as quantum channels are noisy by nature. Error correction in quantum codes differs in three ways to its analog in classical paradigm. Firstly, as a result of the no-cloning theorem, it is impossible to make copies of a quantum state; secondly, the code can't be measured directly, as it would cause the superposition state of the qubits to collapse. Finally, quantum errors are continuous, whereas classical errors are discrete.

One of the most important achievements has been the discovery of Quantum Stabilizer Codes (QSCs) in Gottesmans Ph.D. thesis [3], who showed that the development of quantum coding schemes from existing classical strategies was effective. This has led to the development of many types of codes from which Sparse quantum codes or Quantum Low Density Parity Check (QLDPC) are going to be highlighted. The "low density" property helps in the reduction of the necessary quantum interactions per qubit in error correction procedure. Moreover, QLDPC codes are based on existing classical LDPC[2], that can be defined as stabilizer codes with sparse generators.

The root LDPC codes guarantee their low density by imposing that each equation involves small number of bits as well as each particular bit being involved in a small number of equations. This is seen with the Parity Check Matrix (PCM) where each row represents a parity check equation and each column a coded bit. PCMs can also be represented as factor graphs, where which variables are arguments of which local functions is shown. In this case, there are two types of nodes; variable nodes (error nodes) and parity check nodes.

[1] Quantum mechanics is the field of physics that describes the behaviour of matter on an atomic scale.
[2] LDPC codes protect information against noise effectively and their sparse structure permits iterative belief propagation decoder of low complexity.

Decoding is performed with the Belief Propagation (BP) algorithm, which is limited as it attempts to get the single most likely error instead of considering the degenerate nature of quantum errors. Moreover, unavoidable 4-cycles appear in the factor graph corresponding to the GF(4) parity-check matrix due to the requirement that all stabilizer generators must commute. The BP algorithm can be classified within the framework of the Sum Product Algorithm (SPA), a generic message passing algorithm that computes various marginal functions associated with a global function. The low-density property ensures a small number of loops, giving therefore a good performance when decoded based on the SPA.

This led to the proposal of Calderbank-Shor-Steane (CSS) codes (subset of Stabilizer codes), as they reduce the number of 4-cycles present by representing generators as elements of $GF(2)^{2n}$. These codes allow the error to be divided into X and Z component, and each is decoded individually using two binary GF(2) decoders. However, it must be noted that this counterpart of the depolarizing channel ignores the correlations between both components. CSS codes have been used with numerous types of LDPC codes, having achieved their greatest performance when used with LDGM codes, which are a specific type of LDPC code. These Low Density Generator Matrix codes have shown great improvements, especially a parallel concatenation of two regular LDGM codes, and they are deeply analyzed in [27], [28], [29] and [30].

Based on the aforementioned concepts, the idea for this project arises: incorporating decoding concepts to reconsider the correlations between the X and Z components of the errors. CSS codes present a limit called CSS lower bound [3] and many researchers have tried to deal with this problem at the expense of greater decoding complexity. However, a non-CSS code based on LDGM will be analyzed (it is proposed in [18] and [19]) and similar performance while higher rate or better performance at same rate will be demonstrated.

The proposed non-CSS QLDGM code is designed from a CSS QLDGM code, by performing specific row operations on their quantum parity check matrices to modify the associated decoding graphs. This code outperforms the best QLDPC codes that have appeared to this day. In this project, different modified belief propagation non-CSS QLDGM decoders will be tested and compared to the performance of the standard version and those from the QLDPC version. Augmented, adjusted, enhanced and combined [16] decoders will be explained and analyzed.

2. CODING THEORY

2.1 Introduction to Error Control Coding

2.1.1 Preliminaries

Communications have 3 basic steps: Encoding a message at its source, transmitting that message through a communication medium (channel) and decoding the message at its destination.

In 1948, Claude Elwood Shannon published a state-of-the-art paper [4] in which he studied the limits of communications over noisy channels, and which, ultimately, gave birth to the field of Telecommunications Engineering. The most important discovery was the one about the channel capacity, which was defined as the maximum rate at which we can communicate with a reduced number of errors. Shannon proved in his noisy channel coding theorem that a channel with arbitrarily low error probability can be achieved if the information can be transmitted at a rate less than the channel capacity. Shannon did not specify how to design these codes that achieve the limit, a task that coding theory attempts to tackle.

Codes are designed by properly adding redundant bits (parity bits) to the source bits, which facilitate proper detection and correction of transmission errors (noise, fading, interferences). For example, a rate ½ Repetition code will detect single errors, but not double errors, and it won't be able to correct single errors. However, a rate 1/3 Repetition code, will detect single and double errors, and will be able to correct single errors, but not double errors. In addition, it must be noted that the code rate is not the sole design concern, given that there is no point in designing capacity achieving codes if they can't be used for actual data transmission.

2.1.2 Digital Communication Coding System Model

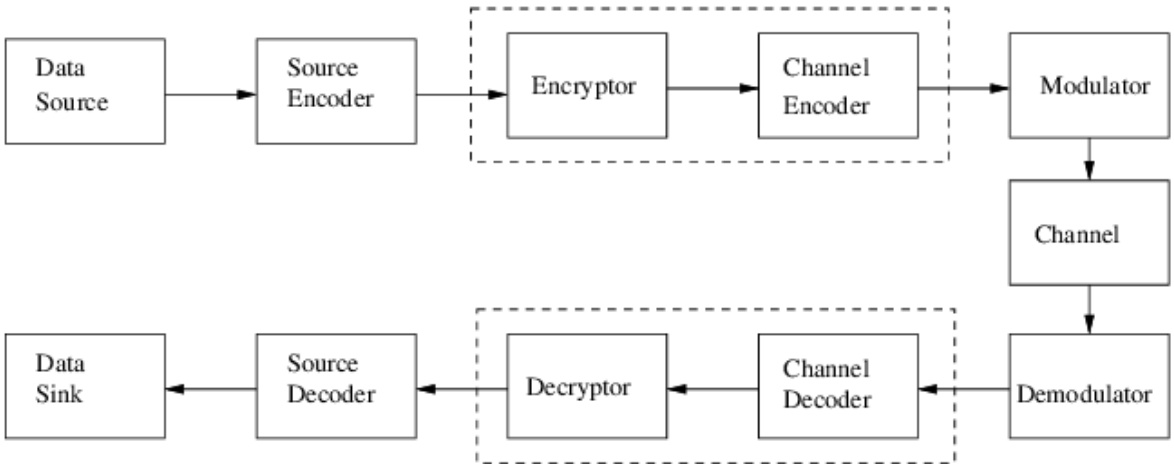

Figure 1:Block diagram of a digital communication coding system model

Figure 1 portrays a common block diagram of a digital communication coding system model, and principal function of each is now described. The source encoding is responsible for the data compression, to minimize the number of bits per time unit. Encryption makes source bits secure, so that unwanted users should not be able to see what's transmitted. Channel coding corrects transmission errors introduced by the channel by means of adding redundant bits in the source, and it is separated from the source encoder block in order to have control over those redundant bits that are inherent in a source, but added carefully in the channel encoder.

Modulation maps the codewords into waveforms which are then transmitted over the physical medium known as the channel (PSK, QAM). The channel is the physical medium (free space, wire, powerline...), and corrupts transmitted waveforms with multipath transmission, noise, fading or interference. Demodulation converts the received noisy waveform into a sequence of bits (estimation of transmitted information), and can be hard (just two quantization levels) or soft (unquantized or with more than 2 quantization levels).

Channel decoding estimates the information bits and corrects transmission errors. The performance of an error correcting scheme over a specific channel is usually measured by the BER or FER, which are typically plotted on graphs.

- BER (Bit Error Rate): Expected number of information bit decoding errors per decoded information bit.
- FER(Frame Error Rate): Percentage of frames in error[3].

Finally, Decryption recovers the plain text from the cipher text with the help of the key and Source Decoding reconstructs the original source bits from the decoded information sequence.

2.1.3 Types of Channel Codes

- Block Codes:

The information sequence is partitioned into message blocks of k information bits, $u = (u_0, u_1, \dots, u_{k-1})$ and the encoder maps each block of k-information bits u to an n-bit codeword $v = (v_0, v_1, \dots, v_{n-1})$. One of the key differences between this type of code and the others is that the encoder is memoryless, the output depends only on that current block of k-bits.

The ratio k/n is known as the code rate, denoted by R, and n-k is the number of redundant bits (also known as parity bits) added to protect from errors. Finally, the set of 2^kcodewords of length n is called a binary (n,k) block code.

- Convolutional Codes:

The information sequence is processed in a continuous fashion, and the n-bit encoder has memory of order m, implying that each output at a particular time depends not only on the k-bit information

[3] A frame error occurs if any information bit in that data frame is in error.

sequence, but also on m previous input blocks. It is called a (n,k,m) convolutional code and n,k values are usually much smaller for convolutional codes than for block codes.

2.1.4 Decoding Strategies

The channel decoder produces an estimate of the information sequence, $\hat{v}$, from the received sequence r, and the source decoder uses inverse encoder mapping to find the information sequence $\hat{u}$ corresponding to $\hat{v}$.

The average probability of error is given by:

$$P(E) = P(\hat{v} \neq v) = \sum_{r} P(E|r) * P(r) = \sum_{r} P(\hat{v} \neq v|r) * P(r)$$

To reduce error probability, $P(\hat{v} \neq v|r)$ is minimized, what equivalently means maximizing $P(\hat{v} = v|r)$, which indeed is:

$$P(v|r) = \frac{P(r|v) * P(v)}{P(r)}$$

for each r, and every v, choosing v that maximizes $P(v|r)$. This means that maximizing $P(v|r)$ is the same as maximizing $P(r|v) * P(v)$ since P(r) doesn't depend on v. A Maximum A-posteriori Probability (MAP) decoder chooses $\hat{v}$ such that $P(v|r)$ is maximized. A Maximum Likelihood (ML) decoder chooses $\hat{v}$ such that $P(r|v)$ is maximized.

For a Discrete Memoryless Channel (DMC) where each received symbol depends only on the corresponding transmitted symbol:

$$P(r|v) = \prod_{i} P(r_i|v_i)$$

And since log x is a monotone increasing function of x, maximizing $P(r|v)$ is equivalent to maximizing $\log(P(r|v))$.

$$\log\big(P(r|v)\big) = \sum_{i} \log\left(P(r_i|v_i)\right)$$

2.1.5 Error Control Strategies

- Forward Error Correction (FEC):

These error correcting codes work in one-way systems, where transmission takes place only in one direction, from transmitter to receiver. Therefore, the code allows to correct detected errors without retransmitting the information again.

- Automatic Repeat request (ARQ):

This type of code works perfectly if the employed link is very good, as it detects errors with parity bits and resends information when necessary, but it is unable to correct errors. It is usual in two-way systems, where there exists a feedback path from the receiver to the transmitter.

- Hybrid Automatic Repeat request (HARQ):

This is a combination of FEC and ARQ, as receiver sends a negative acknowledgment (NACK) when an error is detected and it is unable to correct it, so that transmitter resends information. As a result, this code works better on poor signal conditions, in which ARQ would perform negatively.

2.2 Linear Block Codes, Generator Matrix and Parity Check Matrix

2.2.1 Linear Block Codes & Generator Matrix

A (n,k) binary linear block code can be defined by a k x n generator matrix G, which is used to map information words to codewords. Each entry can be either 0 or 1, so there is a set of 2^k codewords.

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix}$$

Each information sequence $u = (u_0, u_1, \dots, u_{k-1})$ is mapped into:

$$v = (v_0, v_1, \dots, v_{n-1}) = u * G = (u_0 * g_0 + u_1 * g_1 + \cdots + u_{k-1} * g_{k-1})$$

There are two conditions any linear block code must respect: The sum of any two codewords in a linear code is also a codeword and the all zero vector is a codeword in every linear code.

2.2.2 Parity Check Matrix

The (n,k) linear block code can be defined in its systematic form:

$$G = [P: I_k] = \left(\begin{array}{cccc|cccc} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0 & 0 & \dots & 1 \end{array} \right)$$

P represents the encoding operation, and I_k is the message part of the systematic linear block code. The codeword will be made up of k unaltered message bits and n-k encoded bits.

A linear (n,k) block code can also be specified by a (n-k) x n parity check matrix $H = [I_{n-k}: P^T]$, and a binary n-tuple v is a codeword if and only if $v * H^T = (0,0, \dots ,0)$.

2.3 Syndrome, error detection and error correction

2.3.1 Syndrome and error detection

Let $v = (v_0, v_1, \dots, v_{n-1})$ be a codeword from a binary (n,k) linear block code with generator matrix G and parity check matrix H. Then, if v is transmitted over a Binary Symmetric Channel (BSC), the received sequence is:

$$r = (r_0, r_1, \dots, r_{n-1}) = v + e = (v_0 + e_0, v_1 + e_1, \dots, v_{n-1} + e_{n-1})$$

where the binary vector e is the error pattern. The "1's" in e mean transmission errors, as e_i means that i^{th} position in r has an error.

$$e_i = \begin{cases} 1 \; if \; r_i \neq v_i \\ 0 \; if \; r_i = v_i \end{cases}$$

Once r is received, the decoder must detect errors and correct them. Error detection is achieved by computing the (n-k) tuple known as the syndrome:

$$s = (s_0, s_1, \dots, s_{n-k-1}) = r * H^T$$

The received sequence r will be a codeword if and only if $s = r * H^T = 0$. If it satisfies this condition, then r is a codeword and no errors will have taken place. However, if $s = r * H^T \neq 0$, then transmission errors have occurred. Finally, it could be the case in which $s = r * H^T = 0$, but r is different to the codeword send, this is called undetected error and can occur when e is a non-zero codeword.

An interesting fact is that the syndrome s actually depends only on the error pattern e and not on the transmitted codeword, as is shown below:

$$s = r * H^T = (v + e) * H^T = v * H^T + e * H^T = e * H^T$$

2.3.2 Syndrome and error correction

For an error pattern e and a PCM H given by:

$$H = [I_{n-k} : P^T] = \left(\begin{array}{cccc|ccccc} 1 & 0 & \dots & 0 & p_{0,0} & p_{1,0} & & \cdots & p_{k-1,0} \\ 0 & 1 & \dots & 0 & p_{0,1} & p_{1,1} & & \cdots & p_{k-1,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & & p_{k-1,n-k-1} \end{array}\right)$$

The syndrome equations can be rewritten as:

$$s_j = e_j + e_{n-k} * p_{0,j} + e_{n-k+1} * p_{1,j} + \cdots + e_{n-1} * p_{k-1,j}, 0 \leq j \leq n-k$$

This is a set of n-k equations and n unknowns $(e_0, e_1, \dots, e_{n-1})$. The decoder must solve these equations for an estimated error pattern $\hat{e}$. Therefore, estimated codeword is $\hat{v} = r + \hat{e}$.

There are 2^k possible solutions to the syndrome equations and only one is correct. To minimize the probability of a decoding error, the most probable error pattern that satisfies the above set of equations is chosen as the true error vector.

For the previous BSC example, the maximum likelihood decoder chooses $\hat{v}$ as the codeword that minimizes Hamming weight of the error pattern e.

3. LDPC CODES

3.1 Definition

The density of a source of random bits is basically the expected number of 1s in the source. A source is sparse if its density is less than 0.5 and a vector v is very sparse if its density vanishes as its length increases[4]. The overlap between two vectors is the number of 1's in common between them.

Low-density parity-check (LDPC) codes are codes specified by a parity check matrix H containing mostly 0's and only a small number of 1's.

A regular (n, w_c, w_r) LDPC code is a code of block length n with a m x n parity check matrix where each column contains a small fixed number, $w_c \geq 3$, of 1's and each row contains a small fixed number, $w_r \geq w_c$, of 1's. Therefore, the number of 1's in each row and column of the parity check matrix is fixed. In other words,

- There are w_c 1s in each column, what means that each bit appears in w_c parity check equations. Therefore, each parity check constraint involves w_r codebits, and each codebit is involved in w_c constraints.

- Low-density implies that $w_c \ll m$ and $w_r \ll n$.

- Number of ones in the parity check matrix $H = w_c * n = w_r * m$.

- $m \geq n - k \rightarrow R = \frac{k}{n} \geq 1 - \left(\frac{w_c}{w_r}\right)$, and thus $w_c < w_r$.

3.2 Tanner Graphs

A Tanner graph for an LDPC code is a bipartite[5] graph in which there are two types of nodes: variable nodes (corresponding to n bits in the codeword) and check nodes (corresponding to m parity check equations).

An edge connects a variable node to a check node if and only if that particular bit is included in the parity check equation. A cycle of length l in a Tanner graph is a path comprised of l edges from a node back to the same node.

[4] The number of 1s is fixed even if we increase the length of the vector, therefore density vanishes as length increases.

[5] A bipartite graph is one in which the nodes can be partitioned into two classes, and no edge can connect nodes from the same class.

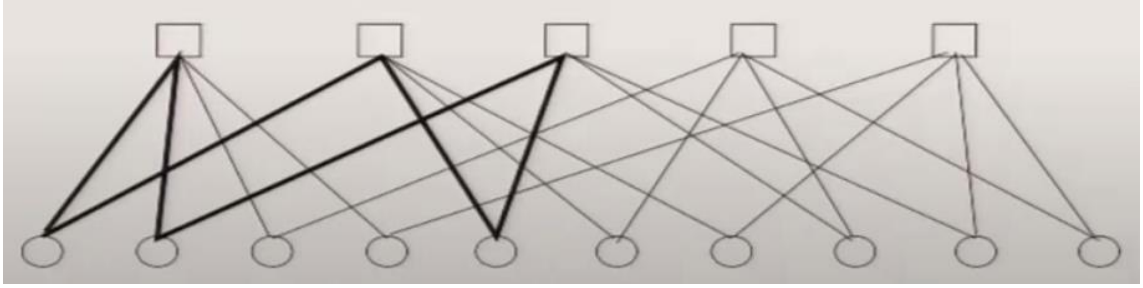

Figure 2:Bipartite graph with cycle of length six

The length of the smallest cycle in the graph is known as its girth. When decoding LDPC codes using the sum-product algorithm (which will be explained in another section below), the number of independent iterations of the algorithm is proportional to the girth of the Tanner graph.

3.3 Gallager's and MacKay's construction

- Gallager's construction

Construct a m x n[6] matrix with w_c 1's per column and w_r 1's per row (A (n, w_c, w_r) code). Divide a m x n matrix into w_c, $\frac{m}{w_c} x\ n$ sub-matrices, each containing a single 1 in each column. The first of these sub-matrices contains all 1' in descending order (i^{th} row contains 1's in columns $(i-1) * w_r + 1$ to $i * w_r$). The other sub-matrices are merely column permutations of the first sub-matrix.

- MacKay's construction [5]

An m by n matrix (m rows, n columns) is created at random with weight per column w_c, and weight per row w_r, and overlap between any two columns no greater than 1. Another way of constructing regular LDPC codes is to build the parity check matrix from non-overlapping random permutation matrices.

3.4 Irregular LDPC codes

An irregular low-density code is a code of block-length N with a sparse parity check matrix where column distribution $\lambda(x)$ and row distribution $\rho(x)$ is respectively given by

$$\lambda(x) = \sum_{i>1} \lambda_i * x^{i-1}$$

[6] N transmitted block-length of an information sequence of length k and m is the number of charity check equations.

$$\rho(x) = \sum_{i>1} \rho_i * x^{i-1}$$

Where λ_i and ρ_i denote the fraction of edges incident to variable and check nodes with degree i, respectively.

3.5 Decoding of Low Density Parity Check Codes: Belief Propagation Algorithm

Decoding of LDPC codes is based on the sum-product algorithm, which is often also referred to as the belief propagation algorithm. In this sub-section, the belief propagation algorithm is briefly introduced, and it's relationship to the SPA will be discussed at length in sections 3.5.1 and 3.5.2. The belief propagation algorithm is a specific instance of the SPA, which is why we analyze the more general algorithm (the SPA) later on in this work. Belief propagation decoding operates as follows:

1. In the first step, algorithm is initialized, with received values from the channel. Based on that, initial p's and q's[7] are computed, and these are assigned to check nodes.
2. In the second step, check nodes do local computation on the probability of a check node being satisfied given a particular bit c that is 0 or 1. Then, this information is passed over the edges back to bit nodes.
3. Now, these bit nodes are getting information from other check nodes as well because each bit node is connected to multiple parity check equations. So it takes those inputs into account to update its q's and these are passed back from bit nodes to check nodes.
4. Back in the check nodes, A Posteriori Probability (APP)[8] ratios for each bit position i are computed and the process is repeated iteratively until all the parity check constraints are satisfied.
5. Finally, hard decisions are computed and the stop-continue criterion is checked. If the calculated APP is bigger than 0.5, then the estimated codeword in that position i will be decided as a 1. Otherwise, that position will be defined as a 0.

3.5.1 SPA Algorithm

As was mentioned previously, the SPA is the best-known algorithm to decode many channel codes. It operates based on a divide-and-conquer approach, as it transforms a problem involving complex global functions of many variables into a problem in which the complex global functions are factorized into simpler functions with less variables.

[7] q's are the messages passed from bit nodes to check nodes, whereas the p's are messages from check nodes to bit nodes

[8] The input bit probabilities are known as "a priori probabilities", as the probabilities are known before initiating the LDPC decoding process. On the other side, the bit probabilities returned by the decoder are known as "a posteriori probabilities".

Knowing that LDPC codes can be represented using factor graphs, and given that these graphs represent global functions of many variables, it is logical to think that the SPA will serve to decode these codes. A deeper explication on how the SPA works is presented in [6], but summarizing, it is based on message passing between the edges of the factor graph. The message passing is done between two levels of the factor graph, and they are usually named variable nodes and function/check nodes. They are usually presented as a circle and a square respectively, but this depends on the developer's notation; and more importantly, the messages generated from one level to the other and vice versa are always different.

Generally, the exchange of messages will last until the algorithm finishes on its own, except for the cases in which the factor graph has cycles. In such instances, there will be 'pending' messages on the cycle edges and no exact solution will be reached (instead an approximation will be produced). In the case of LDPC codes (recall that LDGM codes are a subset of LDPC codes), there are usually cycles and therefore the message passing needs to be stopped. In [7], it is well explained how these approximations tend to be near the exact solutions when the algorithm runs the cycles several times.

If we consider a scenario in which codeword x has been sent and noisy vector y has been received, where x has been encoded with an LDPC generator matrix G and parity check matrix H known by the receiver. Then, if the receiver knows the noise variance, it can compute the log-likelihood ratio for every component of the received noisy vector. The receiver constructs a factor graph based on H and associates each computed log-likelihood to a unique variable node in the factor graph. At this point, the decoding process can begin.

In the general case, nodes can be divided into variable nodes and check nodes, where they correspond to the columns of H and rows of H respectively. It needs to be clarified that each time the cycle restarts, the a posteriori probabilities of the values of each variable node are recalculated and this process continues until a fixed number of iterations is reached. Each iteration, each complete cycle, the variable nodes first send a message to the check nodes they are connected to and this message depends on the log-likelihood calculated previously. Then, each check node responds to those variable nodes that send them something with a new message depending on the information they just received. As said before, this process will take place various times, and that's why it is often referred as iterative decoding. Interestingly, it has been shown that after a certain amount of iterations, performance no longer improves, hence the procedure should be halted.

A cycle free code will finish by itself and the remaining a posteriori probabilities will be correct. However, in codes like the ones used in this work where there are cycles, there will be 'pending' messages and therefore, the decoding procedure will have to be stopped manually. This is why the solutions obtained from the iterative algorithm are actually approximations of the real solution.

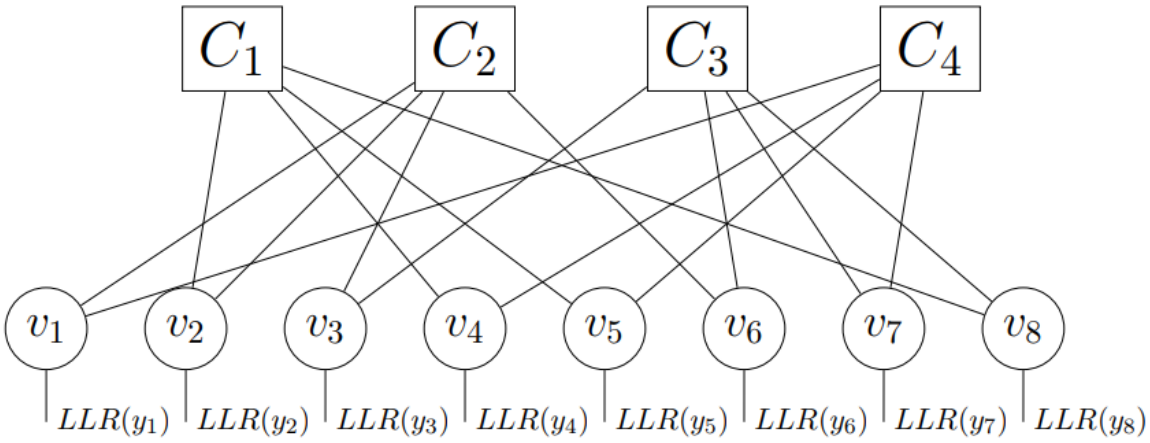

Figure 3:SPA explanation's general factor graph

3.5.2 SPA step by step

In this section we provide a step-by-step analysis of the SPA algorithm.

1. Initialization

In this first step, a log-likelihood ratio is computed for each of the components of the received y. The value of this operation is then assigned to the corresponding variable node, which at this step has the value of:

$$v_i = LLR(Y_i) = \frac{2}{\sigma^2} * y_i$$

where σ is the variance of the Gaussian variable (this expression is derived for the Additive White Gaussian Noise (AWGN) channel in [7]).

2. Messaging the Check Nodes

In this second step, the variable nodes will compute their corresponding probability of being in a specific state (either -1 or +1 in BPSK for example) taking into account the log-likelihood value used to firstly initialize each variable node, and the sum of all the messages received from the check nodes except for the one to which this message is going to be sent. Obviously in the first iteration this second term will be null, as the variable nodes start receiving messages from the check nodes after the third step, and therefore the sent message will be the channel log-likelihood ratio itself.

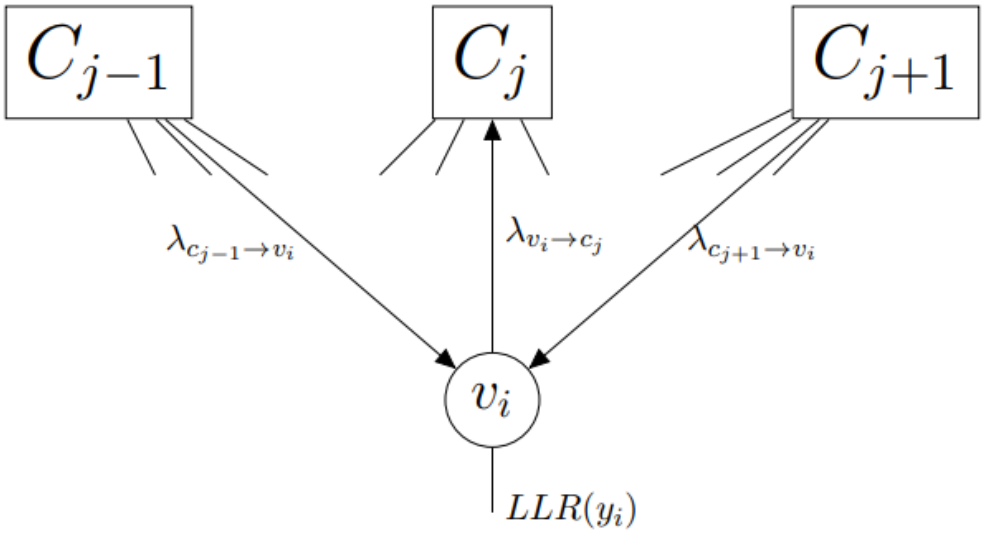

Figure 4: Second step of SPA

As it can be seen in figure 4, the variable node v_i receives as inputs the log-likelihood ratio, and the messages from all the check nodes except for the one to which the new message will be sent. The $\lambda_{c_j\to v_i}$ is the message sent from check node c_j to variable node v_i.

3. Messaging the variable nodes

In this third stage, the check nodes receive many messages from different variable nodes, and they respond to each of them with a specific message. This message could be explained as the certainty the check node has about the variable node to which it is communicating, considering the value it indicated in its previous message. The exact message and how it is derived is well explained in [8].

As in the previous step, the check node will take into consideration all received messages except for the one coming from the variable node to which it will send the new message. This can be perfectly seen on the next figure.

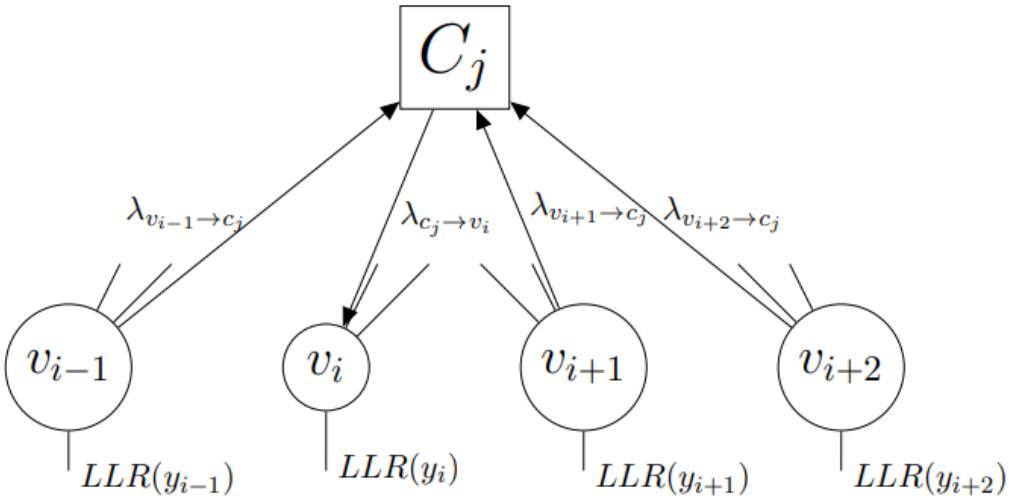

Figure 5: Third step of SPA

4. Stop-Continue

At this point, the algorithm will see if the number of iterations is sufficient to get a good approximation or if it is necessary to execute subsequent decoding iterations, returning to step 2 and repeating the process. Each iteration will improve the approximation, but scientists usually define a limit of maximum iterations as there is one point at which the improvement becomes negligible.

5. Message Prediction/Estimation

Finally, once all the iterations have finished, the decoder must perform hard decision decoding on the a posteriori values of each variable node. Therefore, taking into account the proposed BPSK encoding in which [0,1] are encoded into [-1,1] and the final a posteriori values obtained in the last step of the last iteration, the codeword bits x_i are estimated:

$$\begin{cases} v_i > 0 \to x_i = 0 \\ v_i < 0 \to x_i = 1 \end{cases}$$

The primary drawback of the SPA is the vast number of messages that are exchanged over the factor graph. In the past, this led to it being discarded in many instances because the computational power to run the algorithm successfully was not available. Even presently, many computers might struggle if the algorithm is not perfectly programmed or if the LDPC code matrices are not sufficiently sparse.

4. QUAMTUM ERROR CORRECTION

Up to this point we have discussed error correction in the framework of classical communications. In this section, we will focus on how error correction is performed in the quantum paradigm. Quantum error correction tries to detect error events in qubits[9], when they are sent through quantum channels and then attempts to correct these error events with a high success probability.

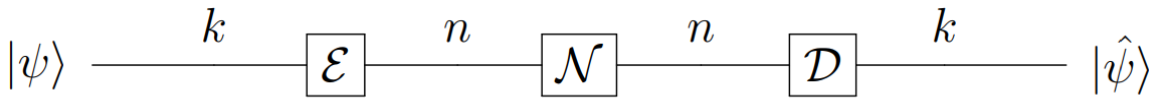

Figure 6:Quantum Scenario Block Diagram

As can be seen in the above figure, the quantum scenario is similar to the classical one, as k target qubits are encoded into n code qubits so that they are not corrupted when travelling through the quantum channel. N refers to the noise introduced by the quantum channel, and D is the decoding operation from which the final estimation is calculated, which might not be correct.

4.1 Effects and Problems for QEC

Quantum Error Correction (QEC) is necessary because quantum information suffers from deleterious effects. The phenomenon responsible for most of these effects is known as decoherence. This mechanism makes the stored qubits suffer from temporal errors, which need to be corrected if quantum information is to be transmitted successfully. On the other hand, the imperfect construction of quantum gates used in quantum algorithms can cause errors locally in the same computer. This leads to the concept of fault-tolerant quantum computing, by which quantum devices work effectively even if their elementary components are imperfect.

In what follows we discuss the two most important issues that QEC has to face. The first one is that measuring a superposition quantum state does destroy such superposition state, and therefore, quantum error correction must be implemented without actually measuring the received quantum states. Fortunately, a solution can be found based on the previously introduced concept of error syndromes, as they can be used to estimate the error that has taken place without destroying the superposition quantum state and therefore correct the transmitted qubits.

The second problem arises as a consequence of the no-cloning theorem which doesn't allow the use of repetitive codes to protect quantum information as it is impossible to recreate an arbitrary quantum state. However, this problem can be dealt with by introducing redundant qubits, redundancy is indeed the base of quantum error correction.

Correction of different errors is possible and as the discretization of errors theorem shows, both continuous and decoherence errors can be corrected. On the one side, continuous errors, which are linear combinations of errors that are associated to the same error syndrome, can be corrected

[9] Basic unit of quantum information, quantum version of classical binary bit (two-state)

by applying the correction related to such syndrome. On the other side, decoherence errors, which are the errors that occur due to the interaction of qubits with the environment, can also be corrected, as any linear combination will be corrected by the discretization of errors theorem.

Finally, throughout this section, many of the error correction problems are analyzed and overcome, and therefore, it can be concluded that quantum error correction is possible. All the above mentioned solutions are what form Quantum Error Correction Codes (QECC).

4.2 Stabilizer Codes

Prior to discussing the specific stabilizer codes that we will use in this work, two types of codes need to be highlighted. On the one side, non-degenerate codes distinguish each error, what means that each error will have a unique error syndrome. On the other side, degenerate codes don't distinguish between some errors as they might have the same error syndrome. Degenerate codes can transmit more information as they can correct errors without actually distinguishing them, although classical decoding algorithms cannot exploit this property appropriately.

The stabilizer construction can be used to construct quantum error correction codes based on classical codes. Most of the QECCs nowadays are stabilizer codes, and that's why they will be further analyzed in this sub-section.

4.2.1 Definition:

As it is shown in [11], let $\varepsilon_n^+ = \{E_n \otimes \dots \otimes E_1 | E_i = I, X, Y, Z\}$ and let $Q \leq C^{2^n}$ be a quantum error code. Then, the stabilizer of Q is defined to be the set:

$$\mathrm{S} = \{\mathrm{M} \in \varepsilon_n^+ \mid \mathrm{M\,v} = \mathrm{v\ for\ all\ v} \in \mathrm{Q}\}$$

S is a group, necessarily abelian[10] if $Q \neq \{0\}$.

Now, let Q be a quantum error correcting code and let S be the stabilizer of Q. Then, the code Q is called a stabilizer code if and only if the condition M*v = v for all $M \in S$ implies that $v \in Q$. Q is the joint +1-eigenspace of the operators in S.

The +1-eigenspace exists because the selected group S is abelian, and therefore all its $M \in S$ elements must commute, that is $[M_i, M_j] = 0, \forall i, j$. In consequence, the simultaneous diagonalization theorem[11] shows that all those elements share a common eigenspace. Finally, the exclusion of $-I_n$ from the stabilizer implies that all the elements of S are Hermitian, and hence have eigenvalues +1.

[10] Also known as a commutative group, is a group in which all its elements commute or in other words, has a symmetric multiplication table.

[11] [A,B]=0 if and only if there exists an orthonormal basis such that both A and B are diagonal with respect to that basis. This allows to determine if two operators share common eigenspace or not.

A quantum error correcting code that encodes k qubits into n qubits, has a code space C(S) of dimension 2^k, and the stabilizer subgroup S will have 2^{n-k} elements. One interesting concept to highlight is the centralizer Z(S) of a stabilizer S, which is the set of Pauli elements that commute with all the elements[12] $M \in S$.

$$Z(S) = \{Z \in G_n : [Z, M] = 0, \forall\, M \in S\}$$

This concept is interesting because Z(S)-S contains the elements that commute with all the elements of S but do not actually belong to S. Finally, to end with this sub-section, it is important to mention that the set of correctable errors will consist of elements in the Pauli group that anticommute with at least one and maximum all the generators of the stabilizer.

4.2.2 Encoding stabilizer codes

Encoding in the quantum paradigm consists in adding n-k ancilla qubits to the k starting qubits in order to generate the n-qubit quantum system. This is implemented by means of an encoding unitary, which generates the encoded quantum state that can be transmitted through the channel.

The Gottesman-Knill theorem is of critical importance for this purpose (see [12,13] for further discussion) as it states: Supposing a quantum computation that involves only state preparations in the computational basis, Hadamard gates, phase gates, controlled-NOT gates, Pauli gates, and measurements of observables in the Pauli group, together with the possibility of classical control conditioned on the outcome of such measurements is performed. Then, such computation can be efficiently simulated on a classical computer.

This is a key theorem, as quantum computers nowadays are still not fully developed (there are some that work with less than 100 qubits[13], but this is still far from the capacity of simulation in classical computers). Thus, most computations related to stabilizer coding are actually run and simulated on classical computers.

[12] In other words, commutes with all the generators of S.

[13] IBM and Google are investing huge amounts of money in a race to reach a meaningful quantum computer.

5. NON-CSS QLDGM CODES

Earlier we mentioned how the sparse nature of LDPC codes reduces the number of quantum interactions per qubit in the error correction procedure, which in turn reduces quantum gate errors and facilitates fault-tolerant decoding.

Low-density generator matrix (LDGM) codes are a subset of LDPC codes, that provide a seamless way to design quantum codes while minimizing the encoding complexity. However, LDGM codes are known to exhibit errors that do not decrease with the block length ([27], [28], [29] and [30]).

This issue was resolved by Lou et al. in [14] and [15] by constructing a design in which two regular LDGM codes were concatenated and shown to outperform the rest of designs. LDGM codes can be cast in the framework of stabilizer codes by using the CSS construction, which can lead to CSS codes (a subset of stabilizer codes that have many improvements but that increase their decoding complexity). In [9] a strategy that exploited the advantages of LDGM codes and avoided some of the complications of CSS constructions was proposed. Given its good performance and simplicity, these codes are used in this project.

This new design is based in a modification of the existing CSS QLDGM codes that reaches similar performance for higher quantum rates, or much better performance for the same rate. The results in [9] are obtained for the independent depolarizing channel ([18], [19], [22], [23] and [26]), in which the individual depolarizing probabilities are all equal, $p_x = p_y = p_z = p/3$. In this case, the channel is fully characterized by the depolarizing probability p, and the probability to leave the qubit unchanged is (1-p).

5.1 QLDGM CSS codes

As mentioned previously, CSS codes are a specific subset of stabilizer codes. For this reason, their parity check matrix can be written as $H_Q = (H_Z|H_X)$, where row i of matrix H_Q is the symplectic representation of the stabilizer generator S_i. In order to construct a stabilizer code from two classical codes, their PCMs must fulfill:

$$(H_1 H_2^T + H_2 H_1^T) mod 2 = 0$$

The above expression is known as the symplectic criterion.

Finding two codes that fulfill the above condition is quite difficult, and it is for this reason that CSS codes are useful, as they introduce a simple way to construct stabilizer codes from classical codes:

$$H_Q = (H_Z|H_X) = \begin{pmatrix} H_Z' & 0 \\ 0 & H_X' \end{pmatrix}$$

where H_Z' and H_X' are the parity check matrices of two classical LDPC codes, and each matrix is used to correct phase flips or bit flips respectively.

As in the classical paradigm, CSS codes can be represented by means of Tanner graphs (which are a specific type of factor graph). Based on this, these codes can be decoded using generic message passing algorithms like the sum-product algorithm (SPA) that works by the factor graph and computes various marginal functions associated with the global function.

As can be seen in figure 7, the CSS codes have at the top of the factor graph the syndrome nodes s, and these can only contain information of either X or Z operators, leading to d_x and d_z nodes respectively.

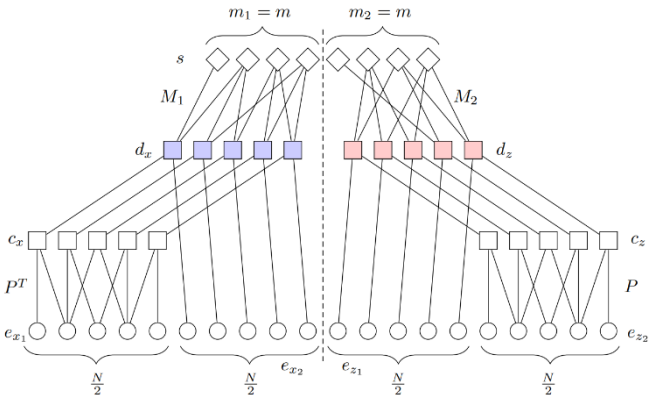

Figure 7:General Factor Graph of a CSS QLDGM code

The PCM of classic LDGM codes with rate ½ can't be directly applied to quantum codes as they result in a 0 quantum rate. This is why the number of rows must be reduced and new PCMs are constructed, while CSS condition is kept. The new PCM of an LDGM-based CSS code is defined in [9]:

$$H_Q = (H_Z|H_X) = \begin{pmatrix} H & 0 \\ 0 & G \end{pmatrix} = \begin{pmatrix} M_1\tilde{H} & 0 \\ 0 & M_2\tilde{G} \end{pmatrix}$$

where M_1 and M_2 are low-density full-rank binary matrices whose numbers of rows satisfy $m_1 < n_1$ and $m_2 < n_2$. $N = n_1 + n_2$ and the final rate is:

$$R_Q = \frac{N - m_1 - m_2}{N}$$

The c and d nodes in figure 7 represent the matrix multiplications used to perform these linear row operations on the parity check matrix and generator matrix.

5.2 Design of LDGM-based non-CSS codes

As mentioned above, CSS codes are decoded separately on a two-sided graph in which X and Z errors act separately on each graph ([22]). However, the non-CSS design of [9] attempts to decode X and Z errors simultaneously by allowing s nodes to connect to both to d_x and d_z nodes, and better performance is shown.

Thus, it is of great importance to design the upper layer of the factor graph correctly in order to get a good non-CSS QLDGM code. The proposed design is based on a CSS quantum code that is designed using classical LDGM codes. There are two efficient methods in order to construct the upper layer efficiently and are: Syndrome node combination, and Syndrome node combination + removal of s_A nodes. These methods are perfectly explained in [9] and as a brief summary, the connections are not made randomly as this would lead to numerous decoding problems.

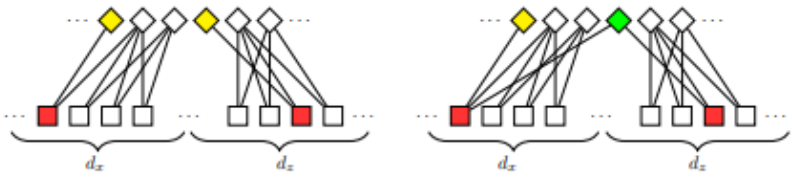

Figure 8:Generation of s_C node with method 1

.

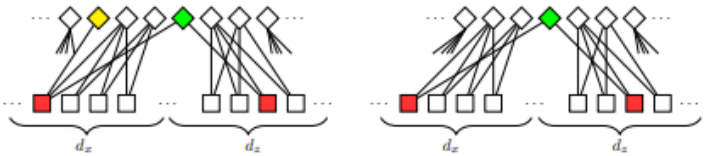

Figure 9:Generation of s_C node with method 2

Ultimately, both of the construction
methods result in a non-CSS QLDGM code in which there is communication between both sides of the factor graph and that outperforms other CSS codes. The decoding of these codes is done using the previously introduced sum-product algorithm, which runs over the factor graph and allows interaction from both sides of the graph during the decoding function.

To end with this section, some interesting simulation results that from [9] will be presented, in order to show that indeed non-CSS codes outperform CSS codes.

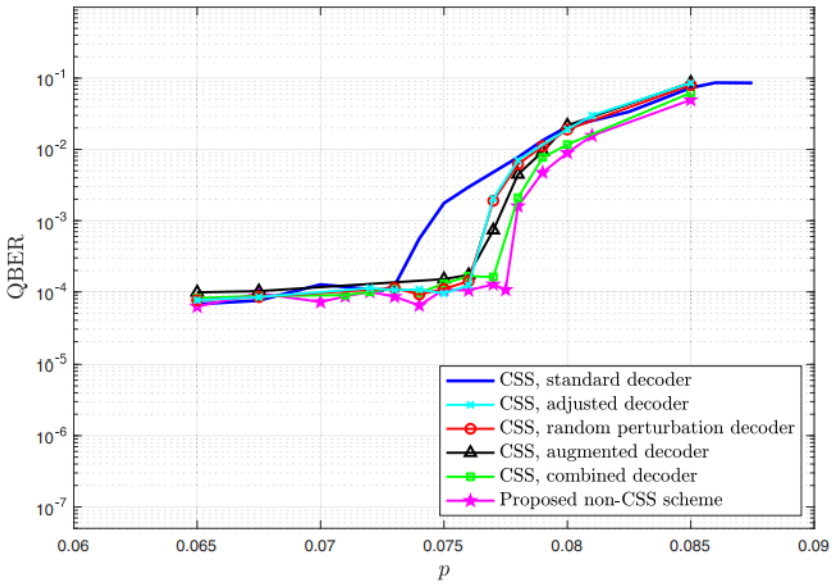

Figure 10: Simulated QBER CSS vs non-CSS over the depolarizing channel

As it can be seen in figure 10, the QBER performance of the proposed non-CSS decoder outperforms the rest of CSS decoders, even if as it can be seen, different types of modified CSS BP decoders have been simulated. This links with the next section because the final goal of this project is to try the modified BP types on the proposed non-CSS decoder and see if there is an improvement or not.

6. MODIFIED BELIEF PROPAGATION DECODERS FOR NON-CSS QLDGM CODES

6.1 Modified belief propagation decoders for QLDPC

In [16], modified belief propagation decoders for quantum low-density parity-check (QLDPC) codes are presented, and based on this paper, similar decoders are implemented for non-CSS QLDGM codes.

In the aforementioned paper, 3 existing decoders are first explained and then 3 new decoders are proposed.

- The first introduced decoder is the random perturbation decoder. The idea of this decoder is to iteratively reattempt decoding with randomly modified channel error probabilities to break decoding symmetries.

- Next, the enhanced feedback (EF) decoder is introduced. In this case, the behavior is very similar to the random perturbation decoder but with the difference that the modification of channel error probabilities is controlled or based on the decoder's output.

- To end with existing decoders, the supernode decoder combines pairs of check nodes in the factor graph to form supernodes. Thus, the decoding complexity is reduced and the number of cycles too, which can lead to considerable performance improvements.

- The first of the new decoders that is proposed in [16] is the augmented decoder. This decoder has previously been applied to classical LDPC codes but has not been used in the quantum paradigm. It operates by re-attempting decoding with a randomly selected subset of check nodes duplicated.

- Another of the decoding strategies proposed in [16] is the adjusted decoder, which tries to reintroduce correlations between the X and Z components. General CSS codes are decoded using two binary BP decoders, one for each component. This ignores the correlations between the X and Z components of the error. The adjusted decoder attempts to reintroduce correlations by checking if one of constituent decoders has failed while the other has not, in which case it modifies the error probabilities of the failed decoder based on those of the successful decoder.

- Finally, a new decoder is presented that combines both augmented and adjusted decoders.

As is shown in Figure 11, the decoders proposed in [16] outperform standard decoding techniques.

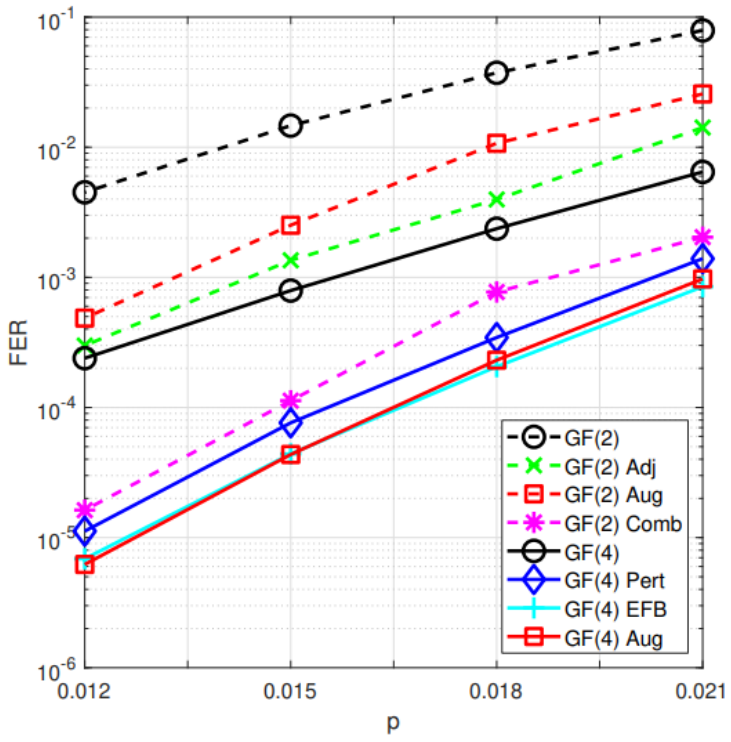

Figure 11: Simulation results from [16]

It is clear that the standard binary and quaternary decoders are outperformed in terms of Frame Error Rate (FER) by their corresponding modified belief propagation decoders for QLDPC codes. From the newly presented decoders, the performance of the combined decoder needs to be highlighted as it shows an appreciable difference versus the augmented and adjusted decoders.

6.2 Modified belief propagation decoders for non-CSS QLDGM

After reviewing coding theory concepts, LDPC codes, quantum coding, non-CSS QLDGM codes, and modified BP decoders, we are in a position to tackle the simulation part of this project. The ultimate goal of this project was to implement and try these modified belief propagation decoders that have successfully worked for QLDPC codes with non-CSS QLDGM codes.

From the previously mentioned decoders, we have implemented and simulated the following in Matlab: enhanced feedback, adjusted, augmented and combined decoders.

6.2.1 Enhanced feedback

The enhanced feedback decoder is designed for the depolarizing channel, and as explained in the above introduction, it's similar to the random perturbation in the way that it reattempts iterations modifying the channel probabilities. In this case, the first iteration is with a standard GF(4) decoder, and if the estimated syndrome is equal to the transmitted syndrome, then the decoding is complete and transmission has been successful. On the other part, if the estimated syndrome is different to the transmitted syndrome $\hat{z} \neq z$, then an erroneous check node is selected with an involved qubit j ∈ $M(i)$:

- If $z_i = 1\ but\ \hat{z}_i = 0$ the adjustment is

$$P(E_j = \sigma) \rightarrow \begin{cases} \frac{p}{2}\ if\ \sigma = I\ or\ M_i^{(j)} \\ \frac{1-p}{2}\ otherwise \end{cases}$$

  Where $M_i^{(j)}$ is the j-th component of the generator M_i.

- If $z_i = 0\ but\ \hat{z}_i = 1$ then the adjustment is

$$P(E_j = \sigma) \rightarrow \begin{cases} \frac{1-p}{2}\ if\ \sigma = I\ or\ M_i^{(j)} \\ \frac{p}{2}\ otherwise \end{cases}$$

  Where $M_i^{(j)}$ is the j-th component of the generator M_i.

The decoder now reattempts the process with these new probabilities and if it fails again, it will choose another qubit k ∈ $M(i)$. The decoder will continue reattempting with new qubits until all qubits are tried, then it will change the selected check node and will start again. Given how long this process can go on for, researchers usually set a maximum number of attempts N.

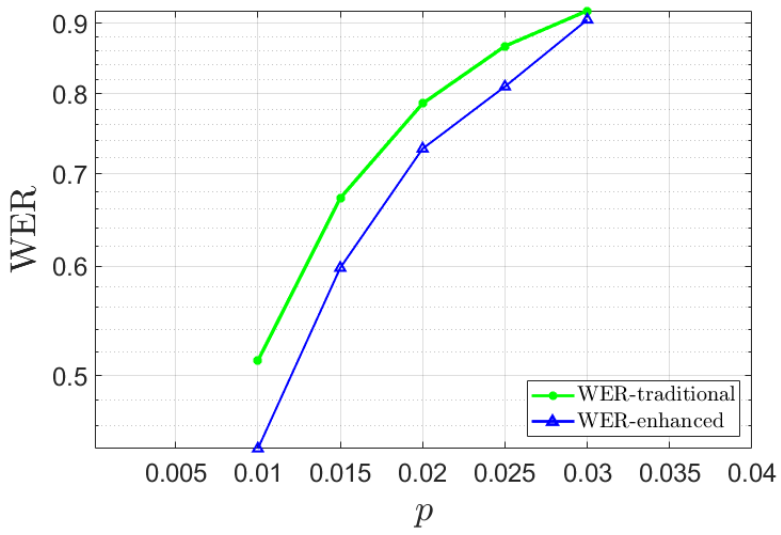

Figure 12: Word Error Rate for Enhanced Feedback decoder with QLDGM codes of length 10.

As can be seen in figure 12, the proposed decoder outperforms traditional decoding in terms of the word error rate. As it shows, the WER difference is reduced with the incrementation of the depolaizing channel probability, but there is still an appreciable difference in the performance. Therefore, for a blocklength of N=10, the proposed decoder provides a substantial performance increase.

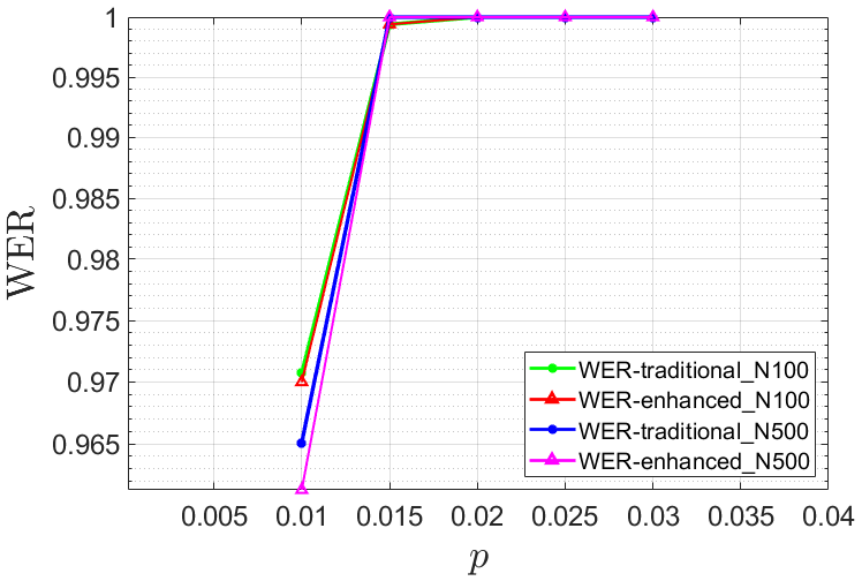

Figure 13: Word Error Rate for Enhanced Feedback decoder with QLDGM codes of length 100 and 500.

However, as figure 13 displays, the performance gain provided by the enhanced decoder is reduced for larger blocklengths. For the depolarizing channel probability of 0.01 there is a little difference and the proposed enhanced feedback decoder shows a better performance. On the other side, when incrementing the depolarizing channel probability from 0.01 upwards, the performance of the proposed decoder and the traditional one is the same. The results showed for the code of length

500 are the ones for a rate of 0.1, another code with rate 0.2 was tested but no errors were corrected.

6.2.2 Adjusted

The adjusted decoder is a GF(2) based but with the difference that it reintroduces the correlation between the X and Z component that the standard version ignores. In this case, the first attempt is done with the GF(2) standard version. If decoding is successful or if both $H_Z\hat{e}_X = \hat{z}_X \neq z_X$ and $H_X\hat{e}_Z = \hat{z}_Z \neq z_Z$ (both sides are unsuccessful), then the decoder stops. However, if one of them is successful and the other is not, the incorrect component will be reattempted using the next channel probabilities:

- If $\hat{z}_X = z_X$ but $\hat{z}_Z \neq z_Z$ then

$$P(e_Z^{(j)} = 1) \rightarrow \begin{cases} \dfrac{p_Y}{p_Y + p_X} \; if \; \hat{e}_X^{(j)} = 1 \\ \dfrac{p_Z}{1 - (p_X + p_Y)} \; if \; \hat{e}_X^{(j)} = 0 \end{cases}$$

- If $\hat{z}_X \neq z_X$ but $\hat{z}_Z = z_Z$ then

$$P(e_X^{(j)} = 1) \rightarrow \begin{cases} \dfrac{p_Y}{p_Y + p_Z} \; if \; \hat{e}_Z^{(j)} = 1 \\ \dfrac{p_X}{1 - (p_Z + p_Y)} \; if \; \hat{e}_Z^{(j)} = 0 \end{cases}$$

In the next figure the performance of the adjusted decoder is shown.

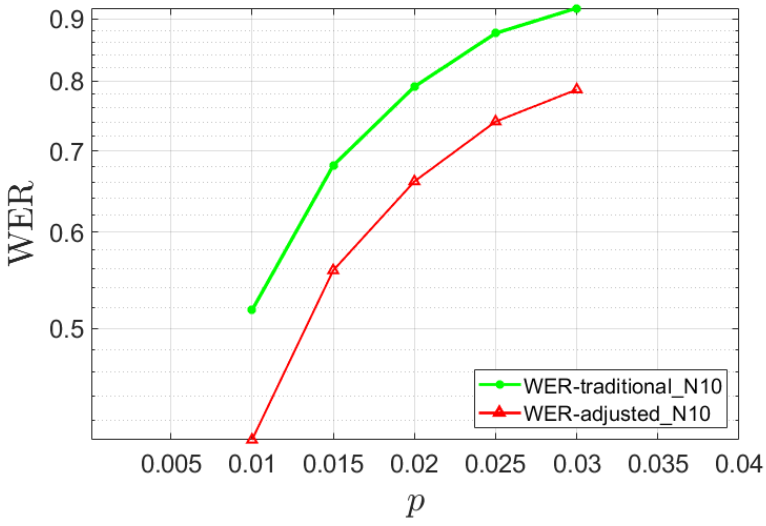

Figure 14 : Word Error Rate for Adjusted decoder with QLDGM codes of length 10.

As can be seen in Figure 14, performance is better than with a standard decoder, even more so than for the EFB decoder. The bigger difference with respect to the enhanced feedback is seen in the depolarizing channel probability of 0.025, as the WER is 0.12 smaller. However, in the same way as with the enhanced feedback decoder, this proposed decoder yields decreasing performance gains as the blocklength increases, as is shown in the following figure.

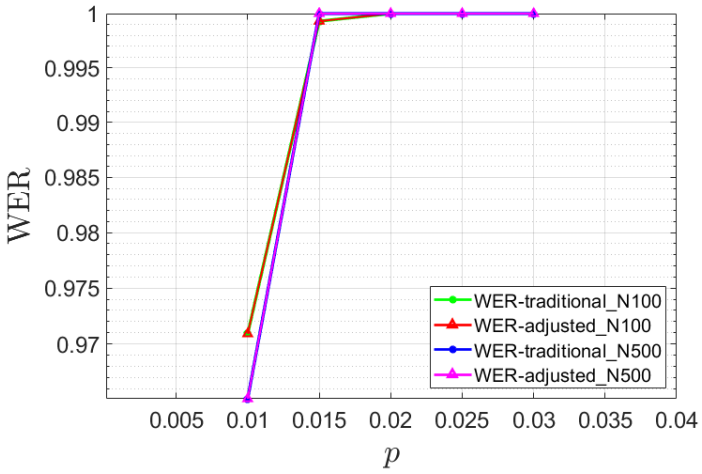

Figure 15: Word Error Rate for Adjusted decoder with QLDGM codes of length 100 and 500.

Differently to what happens in enhanced feedback, in this proposed adjusted decoder, when incrementing the code length to 100 or 500 there are no errors corrected. Therefore, it may be possible that this decoder is only useful for codes with small blocklengths. The performance for both rates in the case of length 500 is the same.

6.2.3 Augmented

The augmented decoder operates as follows: During the first iteration, the standard version of an SPA decoder is run. If the decoder is successful, then the decoder stops. If the decoder fails, the decoding is reattempted but using a randomly generated augmented parity check matrix:

$$H_A = \begin{pmatrix} H \\ H_\delta \end{pmatrix}$$

Where H_δ is just a subset of randomly chosen rows from H that are going to be repeated in the new parity check matrix and δ is the augmentation density, the fraction of rows that is going to be selected. Therefore, the new syndrome will be:

$$z_A = \begin{pmatrix} z \\ z_\delta \end{pmatrix}$$

Where z_δ is the subset of syndromes of the corresponding H_δ rows. The decoding process is iteratively repeated generating each time different random augmented parity check matrices until either the decoding process is successful or the maximum number of attempts N has been reached.

In the case of this project, the start point of this decoder will be a GF(2) standard version, and therefore it will be the same as with the GF(4) but with two augmented decoders, one for the X component and another for the Z component. For example, the augmented parity check matrix and augmented syndrome for the X decoder are:

$$H_A = \begin{pmatrix} \widetilde{H_Z} \\ \widetilde{H}_{Z\delta} \end{pmatrix} \text{ and } z_A = \begin{pmatrix} z_X \\ z_{X\delta} \end{pmatrix}$$

This duplication in the number of rows obviously affects the factor graph, as the number of check nodes will increase respectively. This is of great interest, as it has been proved that repeating one of the check nodes increases its influence in estimating the error. Moreover, before, each variable node was sending a message to a check node taking into account all the received information except for the one coming from the check node to which it as directed. In this case, as there are check nodes duplicated, a variable node might receive the information of the check node to which it is directed from the duplicated check node. With this new information, the decoder can make a different estimation of the marginal probability and hopefully correct an increased number of bit or phase flips.

The previous presented decoders have had problems with bigger codes and this proposed decoder presents a difference, as it can be seen in the next figure.

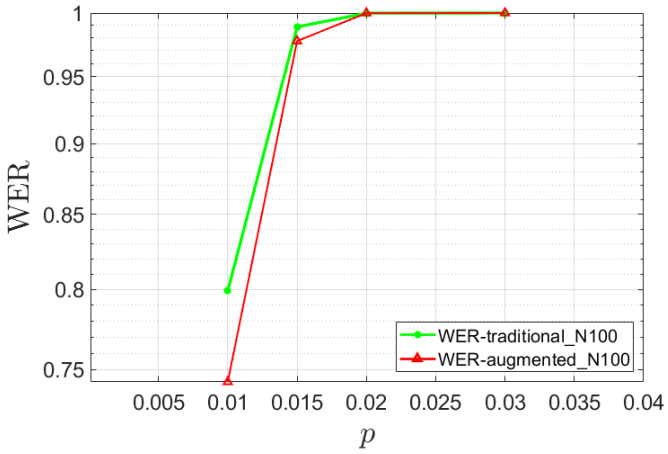

Figure 16: Word Error Rate for Augmented decoder with QLDGM codes of length 100.

In contrast to the performance shown by the previous decoders, this decoder corrects many errors with a bigger code length. As it displays, it's true that when incrementing the probability the performance lowers, but it still outperforms the traditional decoders. However, as it can be seen in the next final figure, the performance for even bigger codes like those with length 500 is the same as the other, it does not correct errors. As in the previous case, the rate difference in the case of length 500 makes no difference in the decoder's performance.

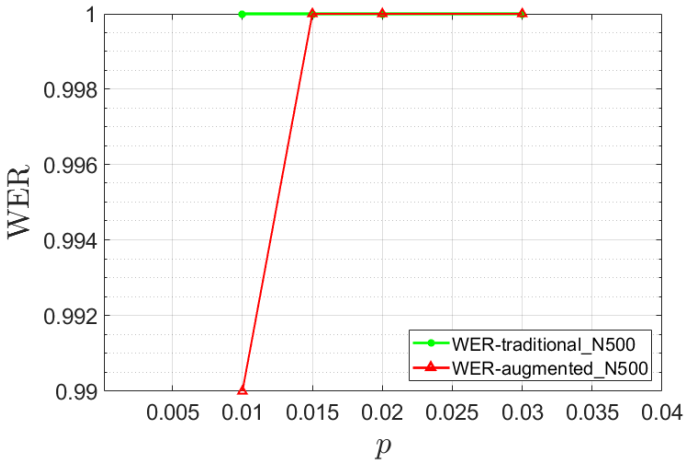

Figure 17: Word Error Rate for Augmented decoder with QLDGM codes of length 500.

6.2.4 Combined

Finally, the combined decoder combines both the adjusted and augmented decoders. In this case, if $\hat{z}_X \neq z_X$ and $\hat{z}_Z \neq z_Z$ then the augmented process begins for the X component first (with a limited number of iterations N). If N iterations have been reached and the decoder has been unsuccessful, then the process is repeated for the Z component and if the decoding is still unsuccessful then the process stops.

On the other side, if one of $\hat{z}_X = z_X$ or $\hat{z}_Z = z_Z$ then the adjusted channel error probabilities are applied to the incorrect component. If still is not successful, then decoding starts with the augmented parity check matrices but the same adjusted probabilities, until the limit of N iterations is reached.

In this final case, this decoder has been studied and analyzed but it has not been tested, that's why it is mentioned in the conclusion and future work section. Looking at the performance of both decoders, it can be predicted that it would have the best performance of all the proposed decoders, as it would combine the efficiency of the adjusted decoder with small codes with the efficiency of the augmented decoder with bigger codes. However, the performance of the decoder with even bigger codes like the one with length 500 could be improved.

7. CONCLUSION AND FUTURE WORK

As a conclusion, all the studied decoders have outperformed the traditional decoder in some way. On the one side, the adjusted decoder has shown the best performance with small codes (code length N10). On the other side, the enhanced feedback decoder is a little bit worse with small codes than the adjusted decoder but it corrects some errors with bigger codes, something the adjusted decoder is incapable of doing.

On the other side, the augmented decoder has been a satisfactory surprise, as in the initial test with code lengths of 100 was the only one to perform acceptably, while it has been necessary to test the others with smaller codes. It is necessary to say that the augmented decoder was slightly tested with small codes and it performed really well, and that is why the research was directed to bigger codes. It would have been great to be able to compare the performance of adjusted and augmented decoders with small codes to see which one yields the best performance for small codes. As for the codes of length 100, the augmented decoder showed a surprisingly good performance.

Apart from the above mentioned comparison, the combined decoder test is also postponed for future work, as it really looks like it could be the best decoder of all the proposed ones. Finally, another type of decoder that has been looked but has not been implemented or tested is the Ordered Statistics Decoder (OSD).

This new proposed decoder has shown significant performance improvements with QLDPC codes and it would be of great interest to try to apply it to the non-CSS QLDGM codes studied in this work. Moreover, this OSD decoder is a post-processing algorithm, which means that it does not replace the mentioned modified belief propagation decoders (it compliments them). Whenever the modified BP algorithm fails to converge into a final result, this post-processing algorithm can be applied and the performance has been seen to improve several orders magnitude in comparison to the BP decoder alone.

As a summary, there is one main algorithm called OSD-0 whose steps can be briefly mentioned as: firstly, it uses the BP soft decision vector to obtain a ranked list of bit-indices ordered from most-to-least likely of being flipped. Secondly, the columns of the PCM are ordered according to the previously mentioned ranking. Afterwards, from the reordered PCM, the first RANK(H) linearly independent columns are selected as the most-probable basis-set. Next, the OSD-0 solution is calculated based on the basis-bits by matrix inversion. Finally, taken into account that the OSD-0 solution will always satisfy the syndrome equation, it is mapped to the original bit ordering.

There are other algorithms like the Higher order OSD that are being researched, but they still need some work. In the next figure (taken from [32]), it can be seen the comparison between this new proposed BP+OSD decoder and the rest of the modified belief propagation decoders mentioned throughout this project. It would be interesting to replicate these experiments for the non-CSS QLDGM codes and assess whether the differences are as drastic as those shown in Figure 18.

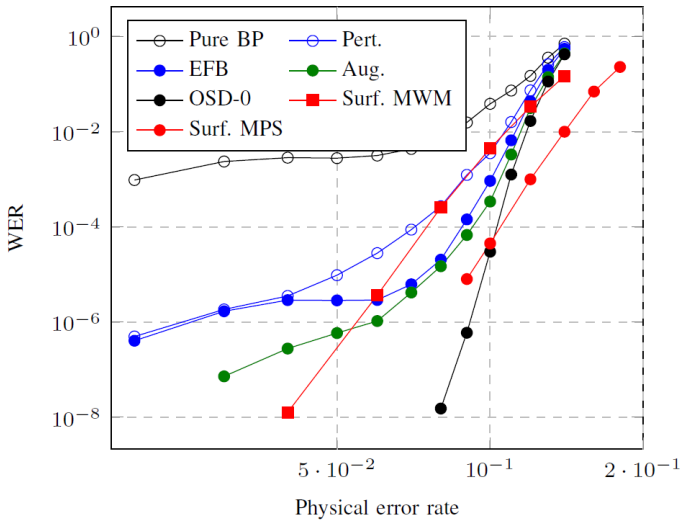

Figure 18: OSD-0 compared to modified belief propagation decoders over QLDPC codes.

As can be seen in the figure, the OSD-0 post-processing algorithm outperforms the rest of the modified decoders that do not use it. In conclusion, it seems that the OSD algorithm would result in further performance improvements and could possibly lead to a near-optimal decoder for QLDPC codes. Implementing this algorithm would be suitable for a future Master Thesis.

8. BUDGET

In this section, the budget of the project will be analyzed; which could be divided into 4 main parts:

- The first part would be the fungible resources, which in this case are the materials that have been used and consumed for the realization of this project.

Quantity	Reference	Description	Price (€)	
			Unitary	Total
100	-	Din A4 pages	0,01	1
Fungible total				1

Table 1: Fungible budget

- Secondly, the material resources, or the used equipment is taken into account, what includes the cost of the use of any machine (amortization and use time will be considered).

Equipment	Acquisition price(€)	Amortization time (years)	Amortization monthly (€)	Use time (months)	Amortization (€)
Computer	1200	4	25	7	175
Total					175

Table 2: Material budget

- Thirdly, the budget of the needed software is studied, where the amortization of the software licenses used will be taken into consideration.

Equipment	Acquisition price(€)	Amortization time (years)	Amortization monthly (€)	Use time (months)	Amortization (€)
Windows 10	259	1	21,6	7	151,2
Matlab	800	1	66,7	5	333,5
Word	135	1	11,25	3	33,75
Total					518,45

Table 3: Software budget

- Finally, the human resources, which considers the cost of the human beings involved in the development of the project.

Task	Duration (hours)	Price (€)	
		Unitary	Total
Project developer – Engineer student	300	25	7.500
Total			7.5000

Table 4: Human resources budget

As a summary, all the parts of the budget are added and the corresponding taxes and costs are taken into account, to give a final budget of 10.906,81€:

Heading	Amount (€)
Fungible	1
Equipment	175
Software	518,45
fHuman resources	7.500
Direct costs (10%)	819,45
Total without VAT	9.013,9
Total with VAT (21%)	10.906,81

Table 5: Total budget

9. REFERENCES

[1] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines," *Journal of Statistical Physics,* vol. 22, no. 5, pp. 563-591, 1980.

[2] R. Feynman, «Simulating physics with computers,» International Journal of Theoretical Physics, vol. 21, pp. 467-488, June 1982.

[3] D. Gottesman, *Stabilizer codes and quantum error correction,* Pasadena,CA,USA: Ph.D. Dissertation, California Inst. Tech, 1997.

[4] C. E. Shanon, «A mathematical theory of communication,» *Bell System Technical Journal,* p. 27:379{423, 1948.

[5] D. J. MacKay, «Good Error-Correcting Codes Based on Very Sparse Matrices,» *IEEE Transactions on Information Theory,* pp. 399-431, Mar. 1999.

[6] B. J. F. F. R. K. a. H.-A. Loeliger, «Factor Graphs and the Sum-Product Algorithm,» *IEEE Trans. Inform. Theory,* vol. 47(2), p. 498–519, 2001.

[7] P. F. Ugartemendia, «BER predictions for LDPC codes by means of EXIT Charts,» A thesis submitted in partial fulfillment for the Graduate degree in Telecommunications Systems Engineering, April 2017.

[8] A. Shokrollahi, «LDPC Codes: An Introduction,» Digital Fountain, Inc., Fremont,CA, April, 2003.

[9] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo, and J. Garcia-Frias, ``Approach for the construction of non-Calderbank-Steane-Shor low-density-generator-matrix based quantum codes,'' *Phys. Rev. A*, vol. 102, pp. 012423, 2020. doi:10.1103/PhysRevA.102.012423.

[10] J. E. Martínez, «Quantum Error Correction: stabilizer coding and beyond,» Tecnun - University of Navarra, Donostia-San Sebastián, July 2018.

[11] A. Klappenecker, «A Short Introduction to Stabilizer Codes,» Department of Computer Science, Texas A&M University.

[12] D. G. a. R. Cleve, «Efficient computations of encodings for quantum error correction,» *Physical Review A,* vol. 56, pp. 76-82, July 1997.

[13] S. A. a. D. Gottesman, «Improved simulation of stabilizer circuits,» *Physical Review A,* vol. 70, p. 052328, November 2004.

[14] H. L. a. J. Garcia-Frias, «Signal Processing Advances in Wireless Communications,» IEEE 6th Workshop, Piscataway,NJ, 2005.

[15] H. L. a. J. Garcia-Frias, «Turbo Codes and Related Topics,» IEEE 4th International Symposium, Piscataway, NJ, 2006.

[16] ∗. J. O. a. P. J. Alex Rigby, «Modified belief propagation decoders for quantum low-density parity-check codes,» College of Sciences and Engineering, University of Tasmania, Hobart, Tasmania 7005, Australia, May 2019.

[17] E. B. Marchante, «Quantum error correction – decoders,» MASTER THESIS WORK, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, September 2019.

[18] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo and J. Garcia-Frias, «A survey on degeneracy and its impact on the decoding of sparse quantum codes,» IEEE Communications Surveys and Tutorials, *submitted to IEEE Access*, on December 2020.

[19] J. Etxezarreta Martinez, P. Fuentes, P. M. Crespo, and J. Garcia-Frias, ``Approximating Decoherence Processes for the Design and Simulation of Quantum Error Correction Codes in Classical Computers,'' *IEEE Access,* vol. 8, pp. 172623-172643, 2020. doi: 10.1109/ACCESS.2020.3025619.

[20] M. A. Nielsen, and I. Chuang, ``Quantum Computation and Quantum Information: 10th Anniversary Edition,'' *Cambridge: Cambridge University Press*, 2011. doi:10.1017/CBO9780511976667.

[21] Z. Babar, D. Chandra, H. V. Nguyen, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, ``Duality of Quantum and Classical Error Correction Codes: Design Principles and Examples,'' *IEEE Communications Surveys Tutorials*, vol. 29, no. 1, pp. 970--1010, 2019. doi:10.1109/COMST.2018.2861361.

[22] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo, and J. Garcia-Frias, ``Design of LDGM-based quantum codes for asymmetric quantum channels,'' *Phys. Rev. A*, vol. 103, pp. 022617, 2021. doi: 10.1103/PhysRevA.103.022617.

[23] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo, and J. Garcia-Frias, ``Performance of non-CSS LDGM-based quantum codes over the Misidentified Depolarizing Channel,'' *IEEE International Conference on Quantum Computing and Engineering (QCE20),* 2020. doi:10.1109/QCE49297.2020.00022.

[24] J Etxezarreta Martinez, PM Crespo, J Garcia-Frías, “On the performance of interleavers for quantum turbo codes”, *Entropy*, vol. 21, no.7, pp. 633, 2019.

[25] J Etxezarreta Martinez, PM Crespo, J Garcia-Frías, “Depolarizing channel mismatch and estimation protocols for quantum turbo codes”, *Entropy*, vol. 21, no. 12, pp. 1133, 2019.

[26] J. E. Martinez, P. Fuentes, P. M. Crespo and J. Garcia-Frías, "Pauli Channel Online Estimation Protocol for Quantum Turbo Codes," *2020 IEEE International Conference on Quantum Computing and Engineering (QCE),* 2020, pp. 102-108, doi: 10.1109/QCE49297.2020.00023.

[27] I Granada, PM Crespo, and J Garcia-Frías, “Asymptotic BER EXIT chart analysis for high rate codes based on the parallel concatenation of analog RCM and digital LDGM codes, *EURASIP Journal on Wireless Communications and Networking,* vol. 1, pp. 1-14, 2019.

[28] I Granada, PM Crespo, and J Garcia-Frías , “Combining the Burrows-Wheeler Transform and RCM-LDGM Codes for the Transmission of Sources with Memory at High Spectral Efficiencies”*, Entropy*, vol. 21, no.4, 2019.

[29] I. Granada, P. M. Crespo and J. Garcia-Frías, "Rate Compatible Modulation for Non-Orthogonal Multiple Access," in *IEEE Access*, vol. 8, pp. 224246-224259, 2020, doi: 10.1109/ACCESS.2020.3043529.

[30] I. Granada, P. M. Crespo, M. E. Burich and J. Garcia-Frías, "Rate Compatible Modulation for Correlated Information Sources," in *IEEE Access*, vol. 9, pp. 65449-65465, 2021, doi: 10.1109/ACCESS.2021.3073972.

[31] M. P. C. Fossorier and Shu Lin, "Soft-decision decoding of linear block codes based on ordered statistics," in *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379-1396, Sept. 1995, doi: 10.1109/18.412683.

[32] P. Panteleev and G. Kalachev, "Degenerate Quantum LDPC Codes With Good Finite Length Performance",Quantum Physics-Cornell University, April 2019, arXiv:1904.02703.

[33] J. Roffe, D.R. White, S. Burton and E. Campbell, "Decoding Across the Quantum LDPC Code Landscape", University of Sheffield, May 2019, doi:10.1103/PhysRevResearch.2.043423, arXiv:2005.07016.