

Simulation of a five qubits convolutional code

¹Aziz Mouzali, ²Fatiha Merazka

¹Departement of physics, ENPEI, Rouiba, Algeria.

² Computer Engineering Faculty, USTHB, Algeria.

Abstract

We describe in this work a five-qubit quantum convolutional error correcting code and its implementation on a classical computer. The encoding and decoding circuits and an error correction procedure are presented. We will verify that if any X, Y, Z error or any product of them occurs on one or two qubit, this correction always allows to recover the useful information or to obtain a list of possible errors. The originality in this correction is the winning time obtained by measuring only the required syndromes, thus avoiding the decoherence phenomenon. Also, we give the average fidelity for double errors recovered as single errors having same syndrome.

Keywords: Quantum correction, convolutional code, implementation.

PACS numbers : 03.67.Acn, 03.67.Hk, 03.67.Lx

1 Introduction

The classical information processing is performed with physical supports, which are governed by the classical physics laws. The quantum information processing uses particles, which obeys to quantum mechanics (electrons, ions, photons...). An electron can carry information : according to its spin, it represents the bit 0 or the bit 1. The quantum information support called qubit is a physical system, which can be in two quantum states representing the bits 0 or 1. The advantage of quantum information processing is the possibility for the qubit to be in a two states superposition, which is forbidden for the classical bit. However, any environment-qubit interaction will break this superposition at one of its two possible states. Another essential tool is the qubits entanglement, which is used in quantum computing. An n-qubits system entangled state is not the n-tensor product of every qubit state, as it is the case for a separable state. The particularity of the entanglement is that any evolution of one qubit state will affect the other qubits states in the entangled system. This specific quantum behavior and the superposition states gives to quantum computing a potential superior to classical computing.

Unfortunately, the entanglement is very sensitive to the environment noise. The superposition and entanglement loss is called decoherence and introduces errors in computation, which requires quantum error correcting code. In fact, while the classical computation error probability is negligible, the error probability is very high in quantum computation. The difference comes from the physical nature of the information support : The electrons number in a capacitor can vary without changing the bit, but if an electron spin changes then the bit value will change. As the experimental protection against decoherence is very hard, several quantum codes have been built to correct computation errors. A quantum error correcting code aims to identify the qubits states alteration caused by the noises and then to recover the correct states. A quantum code is similar to a classical code, but have one particularity: We cannot copy the qubit state to protect it because it is impossible to clone the quantum information. In fact, the codedwords are build by intrication between the superposed qubit state carrying the useful information and a number of additional qubits states called ancillas. If an errors affects the qubits, a correcting procedure allows getting a system separable state, containing the recovered useful information. For further reading, quantum information and computation is treated in details in reference [1]. A good introduction to quantum error correction is given in [2] and quantum computing for non-physicists is described in [3].

The convolutional codes, among the quantum codes, aim to protect a continuous information flow transmitted by a sender through quantum channels and processed by a receiver. The reference [4] describes in details the theory of quantum convolutional codes. We propose in this paper the simulation on Maple of a five qubit quantum convolutional error correcting code using the Feynman program. The work is organized as follows: we begin by giving in section 2 some basics of quantum computing and compare in section 3 the convolutional and the block error correcting codes. The quantum convolutional code are introduced in section 4 and section 5 describes the particular five-qubits convolutional code and its simulation on a classical computer. Section 6 concludes the paper, section 7 gives the references and the code simulation by Feynman program is depicted in section 8 (appendix).

2 Basics gates for quantum computing

Consider an n-qubits system in interaction with the environment. We represent an error as unitary transformation U overall the "n-qubits+environment" state. It can be a bit flip X, a phase flip Z and both of them $Y=iXZ$. We give below, the matricial representation of these errors and their effect on a superposed state $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$. The Kets $|0\rangle$ and $|1\rangle$ correspond, for example, to respectively the spin $+1/2$ and $-1/2$ of an electron, or the fundamental and excited states of an atom. The factors α and β with $(|\alpha|^2 + |\beta|^2 = 1)$ are in general complex number and give respectively the probability $|\alpha|^2$ and $|\beta|^2$ that the qubit is in the states $|0\rangle$ and $|1\rangle$.

If none error occurs, we use the identity matrix I : $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

The matricial representations of a single qubit states are:

$$|Psi\rangle = \alpha |0\rangle + \beta |1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} ; |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} ; |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$I|0\rangle = |0\rangle ; I|1\rangle = |1\rangle ; I|+\rangle = |+\rangle ; I|-\rangle = |-\rangle$$

with $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$; $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ then

$$I(\alpha |0\rangle + \beta |1\rangle) = \alpha |0\rangle + \beta |1\rangle$$

The Bit flip error is represented by the X Pauli matrix : $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$X|0\rangle = |1\rangle ; X|1\rangle = |0\rangle ; X|+\rangle = |+\rangle ; X|-\rangle = -|+\rangle$; then

$$X(\alpha |0\rangle + \beta |1\rangle) = \beta |0\rangle + \alpha |1\rangle$$

The phase flip error is represented by the Z Pauli matrix : $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

$Z|0\rangle = |0\rangle ; Z|1\rangle = -|1\rangle ; Z|+\rangle = |-\rangle ; Z|-\rangle = |+\rangle$; then

$$Z(\alpha |0\rangle + \beta |1\rangle) = \alpha |0\rangle - \beta |1\rangle$$

The Bit and phase flip error is represented by the Y Pauli matrix : $Y = i \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

$Y|0\rangle = i|1\rangle ; Y|1\rangle = -i|0\rangle ; Y|+\rangle = -i|-\rangle ; Y|-\rangle = i|+\rangle$; then

$$Y(\alpha |0\rangle + \beta |1\rangle) = -i(\beta |0\rangle - \alpha |1\rangle) = -iXZ(\alpha |0\rangle + \beta |1\rangle)$$

We also use in quantum error correcting code the Hadamard (H), controlled-not (CNOT or CN), controlled-Z (CZ) and Toffoli (T) gates. These gates act respectively on one, two and three qubits. We give below their matricial representation and effect on some states.

$$\text{Hadamard gate: } H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

This gate transform a simple state in a superposed state and inversely:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle ; H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle ;$$

$$H|+\rangle = |0\rangle ; H|-\rangle = |1\rangle$$

Controlled-not gate (CNOT): $CN = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

The matricial representations of a two-qubit system states are:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} ; \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} ; \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} ; \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

This gate acts on two qubits and flips the second one (target) if the first one (controller) is equal to 1:

$$CN|00\rangle = |00\rangle \quad ; \quad CN|01\rangle = |01\rangle \quad ; \quad CN|10\rangle = |11\rangle \quad ; \quad CN|11\rangle = |10\rangle$$

Controlled-Z (CZ): $CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$

This gate realises the transformation $CZ|11\rangle = -|11\rangle$ and leaves the other Kets unchanged.

Toffoli gate : $T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

This gate acts on three qubit and flips the third one (target) only if the two others (controllers) are equal to 1:

$$T|110\rangle = |111\rangle ; \quad T|111\rangle = |110\rangle \quad \text{with} \quad |110\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |111\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

3 Quantum error correcting code

There are two classes of error correcting codes: block codes used for burst errors (like in secondary memories) and convolutional code aimed at protecting a continuous information flow transmitted over long distance by noisy channel. In block code, after waiting that all the bits arrive at the encoder, we divide them into a finite number of equal blocks which are encoded separately, such that each block is independent of the other blocks during all the encoding procedure. Then, finite number of syndromes extraction and recovery operations are separately realized on each block after waiting that all the bits arrive on the receiver. In convolutional code, the information flow is converted in a single codeword whatever is its length by encoding online unequal successive groups of qubits. The encoding of any qubits frame depends on the preceding and the following frames. Unlike as in blocks code, the encoding starts as soon as the first qubits group arrives at the encoder without waiting the other ones. The most important attribute of the convolutional codes is their maximum likelihood error estimation algorithm for all memoryless channels with a complexity growing linearly with the encoded qubits number (exponentially for the block codes). The syndromes extraction and recovery operations are realized on each encoded qubits frame as soon as it arrives on the receiver. This online processing of the information that characterizes the convolutional codes allows to efficiently counter the decoherence phenomenon better than the block codes. Table 1 summarizes the comparison between block and convolutional codes.

	Block code	Convolutional code
Application	Channel with burst errors (secondary memories,...)	Errors in a long distance communication channel
Coding	Equal blocks separately encoded after the arrival of all the bits to the encoder	Online encoding of unequal qubits frames into a single codeword
Error detection	Error estimation algorithm with exponential complexity starting after the arrival of all the bits to the receiver	Online error estimation algorithm with linear complexity starting at the arrival of the first frame
Decoding	Blocks separately decoded after the arrival of all the bits to the receiver	Online decoding of each qubits frame as soon as it is corrected

Table 1: Block codes versus convolutional codes.

4 Convolutional codes

A quantum convolutional code can be described by using the stabilizer code. We encode k to be protected qubits in a frame of n qubits containing $(n-k)$ ancillas (rate= k/n). In general, we define for an (n,k) qubits code a stabilizer group $G_k = \{M_i, i = 1 \dots (n-k)\}$ with $k < n$. It contains $(n-k)$ generators M_i which are a tensor product of the Pauli operators I, X, Y and Z : $M_i = \{\pm 1, \pm i\} \times \{I, X, Y, Z\}^n$. We define the generator weight as the number of its Pauli operators X, Y and Z . The code subspace C associated to the stabilizer group (n,k) is the whole of the eigenvectors $\{|\Psi\rangle\}$ of the generators with the eigenvalue $+1$: $M_i |\Psi\rangle = |\Psi\rangle, \forall M_i \in G_k$ and $|\Psi\rangle \in C$ [4].

If an error E affects the state $|\Psi\rangle$, the generator action will give $M_i E |\Psi\rangle = \pm E M_i |\Psi\rangle = \pm |\Psi\rangle$ because M_i commutes or anticommutes with E . In fact, we apply the generators on the affected coded state $|\Psi'\rangle = E |\Psi\rangle$ to measure the eigenvalues. The eigenvalues whole of the $(n-k)$ generators M_i corresponding to one error E is called the error syndrome. As we can know all the possible errors syndromes, an error list corresponding to the measured syndromes is deduced. Then, we choose in the list the most probable error and recover the correct coded state.

The stabilizer group of a quantum convolutional codes has a convolutional structure, which gives them two specific properties. First a maximum likelihood error estimation algorithm for all memoryless channels with a complexity growing linearly with the encoded qubits number (exponentially for the block codes). Secondly an on-line qubits encoding : a qubit coded state is transmitted through the communication channel, without waiting the arrival to the encoder of the other qubits [4].

The general procedure of a convolutional code is described in reference [5]. The sender performs the encoding circuit by applying successively on the qubits stream, the same group U of unitaries gates. One particularity of convolutional coding is that two successive groups of gates U_i and U_{i+1} overlap a number m of same qubits. A frame of qubit is sent through the channel as soon as the gates finished acting on it, without waiting the other qubits processing. The encoding circuit is then considered "online" as it acts on a few qubits frame at a time. This rapid processing is necessary because of the decoherence phenomenon. As the noisy channel will probably corrupts the qubits, the receiver performs syndromes measurements to diagnose errors. These measurements require that the used generators should have a finite weight, because the sent qubits stream is composed of a finite qubits number. Also, the generators should be divided in equal groups acting on an equal qubits sets to get an online measurement.

The measured syndromes are processed by an error estimation algorithm (like Viterbi algorithm) to determine the most likely error in a qubits frame. The suitable gates are applied to recover the correct sent coded state. Finally, the received encoded qubits are decoded online (a qubits frame decoding starts just after its correction has

finished) to recover the original qubits stream holding the useful information. Each procedure step can begin even if the previous one is not finished. The qubits are processed frame by frame to obtain at the end an error free quantum data ready for quantum computation. In conclusion, all the operations (encoding, measurement, correction and decoding) in a convolutional code are online, so that the sender and receiver do not have to process sequentially [5].

5 The five-qubits quantum convolutional code

We present below the five-qubits quantum convolutional code described in references [6][7] and our simulation of this code with Feynman Program. .

5.1 Generators

This code uses five qubits ($n=5$) to protect the information contained in one ($k=1$) of them (rate= $k/n=1/5$). We define for this code a stabilizer group $G_1 = \{M_i, i = 1 \dots 4\}$ which contains the four generators M_i :

$$\begin{aligned} M_1 &= ZXXZIIIIII & M_2 &= IZXXZIIIIII & M_3 &= IIZXXZIIIIII \\ M_4 &= IIIZXXZIIIIII \end{aligned}$$

The other generators are given by :

$$M_{4i+j} = I^{\otimes 5i} \otimes M_j \text{ with } i > 0 \text{ and } 1 \leq j \leq 4 ; M_\infty = \dots IIIIZX.$$

All the code generators have the same weight equal to four. We note that the position of every Pauli operator in a generator M_i ($1 \leq i \leq 4$) is five time shifted to the right in the generator M_{i+4} . We also remark that the generators are independants as they commute with each other because we have for every qubit i [6][7] :

$$[X_i, Y_i] = [X_i, Z_i] = [Y_i, Z_i] \neq 0 \quad \text{and} \quad [X_i, X_i] = [Z_i, Z_i] = [Y_i, Y_i] = 0$$

5.2 Encoding circuit

The reference [4] gives the coding circuit of a five-qubits quantum convolutional code (figure 11, page 22). The circuit is executed from the left to the right and each horizontal line represents one qubit on which is successively applied different gates. The ancillas are initially in the $|0\rangle$ state and the qubits holding the useful information in the superposed states $|Q_i\rangle = \alpha_i |0\rangle + \beta_i |1\rangle$. A gate on a qubit is represented by a box placed on it. When a box is related to another qubit, it indicates a conditionally application of the corresponding gate with this qubit as controller. Because of their specificity, convolutional codes propagate the information carried by a qubit to its successors. This information propagation can be a problem in the decoding circuit : an error on a finite number of qubits in the noisy channel, propagates to an infinite number of qubits. To avoid such catastrophic error, the

gates in encoding and decoding circuit are arranged in a finite number of layers in which they commute with each other. This layer structure is the necessary and sufficient condition for noncatastrophicity [6][7] .

5.3 Error detection and correction

To correct a single or double error from qubit 2 to qubit 12 [4] (figure 11, page 22), the syndromes are measured from generators M_1 to M_{10} :

$$\begin{aligned} M_1 &= ZXXZIIIIIII & M_2 &= IZXXZIIIIIII & M_3 &= IIZXXZIIIIIII \\ M_4 &= IIIZXZXZIIIII & M_5 &= IIIIZZXZXZIII & M_6 &= IIIIIIZZXZXZII \\ M_7 &= IIIIIIZZXZXZI & M_8 &= IIIIIIZZXZXZ & M_9 &= IIIIIIIIZZXZXZ \\ M_{10} &= IIIIIIIIIIZZXZXZ & M_{11} &= IIIIIIIIIIZZXZXZ \\ M_{12} &= IIIIIIIIIIIIZZXZXZ \end{aligned}$$

We note that the generators M_1 to M_4 act from the first to the seventh qubits. While the generators M_5 to M_8 act from the sixth to the twelfth qubits. Then the two groups of generators apply on seven qubits with an overlapping of two qubits (the sixth and the seventh). The generators M_9 to M_{12} act from the eleventh to the seventieth qubits with an overlapping of two qubits (eleven and twelve) with the precedent group (M_5 to M_8). It is a general property of a convolutional code that the generators are divided in groups all acting on a same number of qubits with an overlapping of m qubits between two successive groups. For the five-qubits convolutional code we have $m = 2$ and we call it a $(5, 1, 2)$ code or in general a (n, k, m) code [6][7]. The stabilizer group S of an (n, k, m) convolutional code is given by : $S = sp \{ M_{j,i} = I^{\otimes jxn} \otimes M_{0,j} , 1 \leq i \leq n - k , 0 \leq j \}$ where $M_{0,j} \in G_{n+m}$ and $M_{j,i}$ are independent and commute with each other [4].

5.4 Syndromes extraction

We explain, below, the error detection made by syndrome extraction [7]. This is a non-destructive measurement procedure as it does not disturb the system coded state. We use the same symbols as those of the simulation to facilitate its understanding. We begin by adding to the to be measured state $|Q27\rangle$ a new qubit initially in the simple state $|0\rangle$ and then transformed in a superposed state by applying the gate H . The obtained state $|Q28\rangle$ is the tensor product of $|Q27\rangle$ with the added qubit state :

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \quad (1)$$

$$|Q28\rangle = |+\rangle |Q27\rangle = \frac{1}{\sqrt{2}}(|0\rangle |Q27\rangle + |1\rangle |Q27\rangle) \quad (2)$$

The generator M_i is conditionally applied on $|Q28\rangle$ (if the first qubit state is $|1\rangle$):

$$|Q281b\rangle = M_i |Q28\rangle = \frac{1}{\sqrt{2}} (|0\rangle |Q27\rangle + M_i |1\rangle |Q27\rangle) = \frac{1}{\sqrt{2}} (|0\rangle |Q27\rangle \pm |1\rangle |Q27\rangle)$$

$$\text{as } M_i |Q27\rangle = \pm |Q27\rangle \quad (3)$$

The generator M_i eigenvalue is +1 or -1 for a given error E affecting the coded state $|Q25\rangle$ sent by the encoder, depending if M_i commutes or anticommutes with E :

$$M_i |Q27\rangle = M_i E |Q25\rangle = \pm E M_i |Q25\rangle = \pm E |Q25\rangle \quad \text{as } M_i |Q25\rangle = |Q25\rangle \text{ and } |Q27\rangle = E |Q25\rangle \quad \text{then } M_i |Q27\rangle = \pm |Q27\rangle \quad (4)$$

If the eigenvalue is +1 then M_i and E commute (value 0 in the syndrome). If it is -1, then M_i and E anticommute (value 1 in the syndrome). Then we obtain :

$$|Q281b\rangle = \frac{1}{\sqrt{2}} (|0\rangle \pm |1\rangle) |Q27\rangle = |\pm\rangle |Q27\rangle \quad (5)$$

We apply the gate H_1 (the suffix indicates the first qubit) and obtain a separable system state containing the unchanged coded state $|Q27\rangle$:

$$|Q281c\rangle = H_1 |Q281b\rangle = |0\rangle |Q27\rangle \text{ or } |1\rangle |Q27\rangle \quad \text{as } H_1 |\pm\rangle = |0\rangle \text{ or } |1\rangle \quad (6)$$

We can identify the first qubit state by applying on it the Z_1 gate :

$$|Q281\rangle = Z_1 |Q281c\rangle = Z_1 |0\rangle |Q27\rangle \text{ or } Z_1 |1\rangle |Q27\rangle \quad (7)$$

as $Z_1 |0\rangle = |0\rangle$ and $Z_1 |1\rangle = -|1\rangle$ we finally obtain:

$$|Q281\rangle = |Q281c\rangle \quad (\text{syndrome } 0) \text{ or } |Q281\rangle = -|Q281c\rangle \quad (\text{syndrome } 1) \quad (8)$$

5.5 Syndromes computation

The syndromes are obtained using the commutators of the X_i, Y_i and Z_i gates on qubit "i" : $[X_i, Y_i] = [X_i, Z_i] = [Y_i, Z_i] \neq 0$; $[X_i, X_i] = [Z_i, Z_i] = [Y_i, Y_i] = 0$.

We know for example, that an error X_i anticommutes with a generator M if it contains a Z or Y at its ieme position. An error $X_i Z_j$ anticommutes with a generator M if it contains Z or Y at the position i , or X or Y at the position j . The generators M_{11} and M_{12} are not concerned as they act from qubits 13 to qubits 17. Each row gives the syndrome for the error E contained in the first column. The rows order is aimed at leading the error estimation algorithm whose beginning is presented in appendix 8.4. The values in boldface indicates the generators to be measured for detecting the error E . We don't need to measure all the generators because in most

cases only some of them are sufficient to identify error. In fact, the aim is avoiding the decoherence phenomenon by winning time in measurement.

The tables 2, 3 and 4 below, show the syndromes for all the possible errors (253 errors) on one or two qubits in the frame from qubits 1 to qubits 11 (in fact qubits 2 to qubits 12 if we take in count the first not sent qubit shown in figure 2).

E	Syndromes	E	Syndromes
$Y_3, Z_2 Z_4$	1111000000	$X_3 Z_6$	1001100000
$Z_2 X_5$	1110100000	$X_3 Z_8$	1001011000
$Y_2, Z_1 Z_3$	1110000000	$X_3 X_9$	1001010010
$Z_2 X_6$	1101010000	$Z_1 X_6$	1001010000
$Z_2 Z_5, X_1 X_3, X_3 X_4$	1101000000	$X_3 Z_9$	1001001100
$Z_2 Z_7$	1100110000	$X_3 X_{10}$	1001001010
$Z_2 X_8$	1100100100	$X_3 X_7$	1001001000
$Z_2 Z_6$	1100100000	$X_3 X_{11}$	1001000101
$Z_2 Z_8$	1100011000	$X_3 Z_{10}$	1001000100
$Z_2 X_9$	1100010000	$X_3 Z_{11}$	1001000010
$Z_2 Z_9$	1100001100	$X_3, Z_1 Z_5$	1001000000
$Z_2 X_{10}$	1100001010	$Z_1 Z_7$	1000110000
$Z_2 X_7$	1100001000	$Z_1 X_8$	1000100100
$Z_2 X_{11}$	1100000101	$Z_1 Z_6$	1000100000
$Z_2 Z_{10}$	1100000100	$Z_1 Z_8$	1000011000
$Z_2 Z_{11}$	1100000010	$Z_1 X_9$	1000010010
$Z_2 Z_{11} \quad ?$	1100000010	$X_3 X_6$	1000010000
$Y_1, Z_2, Z_1 X_4$	1100000000	$Z_1 Z_9$	1000001100
$Z_1 Z_4, X_2 X_3$	1011000000	$Z_1 X_{10}$	1000001010
$X_3 X_5$	1011100000	$Z_1 X_7$	1000001000
$Z_1 X_5$	1010100000	$Z_1 Z_{10}$	1000000110
$Z_2 Z_3, X_3 Z_4, Z_1 X_2$	1010000000	$Z_1 X_{11}$	1000000100
$X_3 Z_7$	1001110000	$Z_1 Z_{11}$	1000000010
$X_3 X_8$	1001100100	$Z_1, X_1 Z_2, X_3 Z_5, Z_2 X_4$	1000000000

Table 2: Error syndromes for bit flip X and phase flip Z on one or two qubits. The values in boldface indicates the generators to be measured for detecting the error E.

E	Syndromes	E	Syndromes
Z_3X_6	0111010000	Z_4Z_{10}	0011000100
Y_4, X_1Z_4, Z_3Z_5	0111000000	Z_4Z_{11}	0011000010
Z_3Z_7	0110110000	Z_4, X_2Z_5	0011000000
Z_3X_8	0110100100	X_5Z_8	0010111000
X_1X_5, X_4X_5, Z_3Z_6	0110100000	X_5X_9	0010110010
Z_3Z_8	0110011000	X_2Z_7	0010110000
Z_3X_9	0110010000	X_5Z_9	0010101100
Z_3Z_9	0110001100	X_5X_{10}	0010101010
Z_3X_{10}	0110001010	X_5X_7	0010101000
Z_3X_7	0110001000	X_5Z_{10}	0010100100
Z_3X_{11}	0110000101	X_5X_{11}	0010100101
Z_3Z_{10}	0110000100	X_2X_8	0010100100
Z_3Z_{11}	0110000010	X_5Z_{11}	0010100010
Z_3, X_1X_2, X_2X_4	0110000000	X_5, X_2Z_6	0010100000
X_1X_6, X_4X_6	0101010000	X_2Z_8	0010011000
$X_1Z_5, X_4Z_5, Z_2X_3, Z_3Z_4$	0101000000	X_2X_9	0010010010
X_1Z_7, X_4Z_7	0100110000	X_5Z_7, Z_4X_6	0010010000
X_1X_8, X_4X_8	0100100100	X_2Z_9	0010001100
X_1Z_6, X_4Z_6, Z_3X_5	0100100000	X_2X_{10}	0010001010
X_1Z_8, X_4Z_8	0100011000	X_2X_7	0010001000
X_1X_9, X_4X_9	0100010010	X_2Z_{10}	0010000100
X_1Z_9, X_4Z_9	0100001100	X_2X_{11}	0010000101
X_1X_{10}, X_4X_{10}	0100001010	X_5X_8	0010000100
X_1X_7, X_4X_7	0100001000	X_2Z_{11}	0010000010
X_1Z_{10}, X_4Z_{10}	0100000100	$X_2, X_1Z_3, X_5Z_6, Z_3X_4, Z_4Z_5$	0010000000
X_1X_{11}, X_4X_{11}	0100000101	X_6X_8	0001110100
X_1Z_{11}, X_4Z_{11}	0100000010	Y_6, Z_5Z_7	0001110000
X_4, X_1, X_2Z_3, Z_1Z_2	0100000000	Z_5X_8	0001100100
X_5X_6, Z_4Z_7	0011110000	X_6Z_7, Z_4X_5, Z_5Z_6	0001100000
Z_4X_8	0011100100	X_6Z_9	0001011100
Y_5, Z_4Z_6	0011100000	X_6X_{10}	0001011010
Z_4Z_8	0011011000	X_6X_7, Z_5Z_8	0001011000
X_2X_6, Z_4X_9	0011010000	X_6X_{11}	0001010101
Z_4Z_9	0011001100	X_6Z_{10}	0001010100
Z_4X_{10}	0011001010	X_6Z_{11}	0001010010
Z_4X_7	0011001000	X_6, Z_5X_9	0001010000
Z_4X_{11}	0011000101	Z_5Z_9	0001001100

Table 3: Error syndromes for bit flip X and phase flip Z on one or two qubits. The values in boldface indicates the generators to be measured for detecting the error E.

E	Syndromes	E	Syndromes
Z_5X_{10}	0001001010	Z_8X_{11}	0000011101
X_6Z_8, Z_5X_7	0001001000	Y_9, Z_8Z_{10}	0000011100
Z_5X_{11}	0001000101	X_9X_{10}, X_7X_9	0000011010
Z_5Z_{10}	0001000100	Z_8, Z_8Z_{11}	0000011010
Z_5Z_{11}	0001000010	X_9X_{11}	0000010101
$Z_5, X_6X_9, X_2Z_4, Z_1X_3$	0001000000	$X_9Z_{10}, Z_7X_8, Z_8Z_9$	0000010100
Y_8, Z_7Z_9	0000111100	$X_9, X_9Z_{11}, Z_8X_{10}$	0000010010
Z_7X_{10}	0000111010	X_7Z_8, Z_5X_6, Z_6Z_7	0000010000
Y_7, Z_6Z_8	0000111000	$X_{10}X_{11}$	0000001111
Z_7X_{11}	0000110101	Y_{10}, Z_9Z_{11}	0000001110
X_8X_9, Z_7Z_{10}	0000110100	X_7X_{11}	0000001101
Z_7Z_{11}	0000110010	Z_9, X_7Z_{10}	0000001100
Z_7, Z_6X_9	0000110000	X_{10}, X_7Z_{11}	0000001010
X_8X_{10}	0000101110	Z_9X_{11}	0000001001
X_7X_8, Z_6Z_9	0000101100	$X_7, Z_8X_9, X_{10}Z_{11}, Z_9Z_{10}$	0000001000
Z_6X_{10}	0000101010	Y_{11}	0000000111
X_8Z_9, Z_6X_7, Z_7Z_8	0000101000	$Z_9X_{10}, Z_{10}Z_{11}$	0000000110
X_8Z_{11}	0000100110	X_{11}	0000000101
Z_6X_{11}	0000100101	Z_{10}, X_7Z_9, Z_6X_8	0000000100
X_8, Z_6Z_{10}	0000100100	Z_{11}, X_7X_{10}	0000000010
Z_6Z_{11}	0000100010	$Z_{10}X_{11}$	0000000001
X_8X_{11}	0000100001	X_1X_4	0000000000
$Z_6, X_2X_5, Z_7X_9, X_8Z_{10}$	0000100000	No error	0000000000

Table 4: Error syndromes for bit flip X and phase flip Z on one or two qubits. The values in boldface indicates the generators to be measured for detecting the error E.

5.6 Simulation on Maple with Feynman Program

The Feynman program described in [8][9][10][11] and obtainable from [12] is a set of procedures supporting the definition and manipulation of an n-qubits system and the unitary gates acting on them. Section 8 presents the five-qubits convolutional code simulation, made with Feynman program version 4, 2008.

The appendix 8.1 shows the encoding simulation made with Feynman program. We note that the first qubit will not be sent through the channel, because no gate is applied on it [4] (figure 11, page 22). We also remark that until the state $|Q8\rangle$ the encoding procedure concerns only five qubit (qubit 2 to qubit 6). The encoding from the states $|Q9\rangle$ to $|Q20\rangle$ concerns six qubits. The state $|Q21\rangle$ which is the tensor product of the five-qubits and six-qubits states $|Q8\rangle$ and $|Q20\rangle$ respectively is not yet the coded state. We derive the coded state $|Q25\rangle$ by applying the remaining CZ gates shared by the two former qubits frames. This procedure reduces the running time, because the product of several big matrices drill a too long running time. The final coded state $|Q26\rangle$ corresponds to the system from qubit 2 to qubit 17. To verify

that the coding circuit gives the correct coded state, we measure in appendix 8.2 the generators eigenvalues for the state $|Q25\rangle$ ($M_i|Q25\rangle = |Q25\rangle, 1 \leq i \leq 8$). The message "M_i true" proves that M_i stabilizes the coded state $|Q25\rangle$. As we obtained this message for all the generators, we conclude that our simulated coding circuit is correct. The syndromes extraction explained in section 4.3 is simulated in appendix 8.3 from generator M_1 to generator M_8 . We give in appendix 8.4 the correction for a single and double error in the frame from qubit 2 to qubit 12. The syndromes are listed in the tables 2,3 and 4 in order to make easier the error detection. For each error, we search in the tables, other error having similar syndromes, then we only measure the eigenvalues which allow identifying this error. Making the lowest number of measures is required by the decoherence phenomenon. At the end, the coded state is recovered from qubit 2 to qubit 12. Then, another procedure using generators M_9 to M_{16} starts for detecting error from qubit 13 to qubit 24. Once this detection is beginning, we start decoding the previous corrected qubits frame. The decoding circuit simulated in appendix 8.5 is the coding circuit of figure 2, but inversely executed. For more understanding, we divided the simulation by layers of gates commuting with each other. To reduce the running time, only one gate (cn, cz, Z or H) is applied at a time on a single qubit of the eleven-qubits system. In fact, the running time of the subroutine called **"Feynman_quantum_operator"** increases with the qubits number.

5.7 Results

The table 5 shows the fidelity F calculated for errors with same syndromes and table 6 some different errors with same fidelity. This fidelity is the overlap $F = \langle \Psi_t | \rho_m | \Psi_t \rangle$ where $|\Psi_t\rangle = |Q_1\rangle \otimes |Q_2\rangle$ is the correct transmitted useful qubits state and $\rho_m = |\Psi_m\rangle \langle \Psi_m|$ the density matrix for the measured state $|\Psi_m\rangle = |Q_{1m}\rangle \otimes |Q_{2m}\rangle$. When a double error happens, the state $|\Psi_m\rangle$ is obtained by applying (after syndrome measurement and before decoding) the operator corresponding to a single error (more likely) having similar syndrome. For example, if the coded state is affected by the double error $E = X_5Z_6$, we recover it by applying the operator X_2 , then obtain the fidelity $F = |2\alpha_2^2 - 1|^2 < 1$, and if $E = X_1Z_3$ we obtain $F = 1$. When some double errors have no similar syndrome with single errors, we consider them with equal probability. For example, the errors (X_1X_6, X_4X_6) with same syndrome are recovered by applying the operator X_4X_6 when X_1X_6 error is introduced or inversely. The factors $\alpha_i = \cos(\theta_i/2)$ and $\beta_i = e^{i\phi_i} \sin(\theta_i/2)$ are illustrated in the block sphere by the spherical angles (θ_i, ϕ_i) , then we can express the fidelity $F(\theta_i, \phi_i)$ in term of these angles. The table 7 resume the fidelity $F(\theta_i, \phi_i)$ and average fidelity F_a for all the possible errors on the two useful qubits ($|Q_1\rangle, |Q_2\rangle$). We note that if an error X_5X_6 occurred and is recovered by applying the operator Z_4Z_7 (same syndrome), or

inversely, then both the two useful qubits are affected after decoding. In this case the average fidelity is $F_a = \frac{1}{9}$, but it could be $F = 1$ if the applied operator corresponds to the actually occurred error. The average fidelity is calculated by integrating over the angles (θ_i, ϕ_i) :

$$F_{a_i} = (1/4\pi) \iint F(\theta_i, \phi_i) \sin(\theta_i) d\theta_i d\phi_i ; 0 < \theta_i < \pi \text{ and } 0 < \phi_i < 2\pi \quad (9)$$

E	$F(\alpha_1, \beta_1, \alpha_2, \beta_2)$
$x_1, x_4, x_2 z_3, z_1 z_2$	$1, 1, 1 \text{ ou } 2\alpha_1^2 - 1 ^2, \alpha_1 \beta_1^* - \beta_1 \alpha_1^* ^2 \text{ ou } \alpha_1 \beta_1^* + \beta_1 \alpha_1^* ^2$
$x_2, x_1 z_3, z_3 x_4, x_5 z_6, z_4 z_5$	$1, 1, 1, 2\alpha_1^2 - 1 ^2, \alpha_1 \beta_1^* + \beta_1 \alpha_1^* ^2$
$x_3, z_1 z_5$	$1, \alpha_1 \beta_1^* + \beta_1 \alpha_1^* ^2$
$x_5, x_2 z_6$	$1, 2\alpha_1^2 - 1 ^2$
$x_6, z_5 x_9$	$1, 2\alpha_2^2 - 1 ^2$
$x_7, z_8 x_9, x_{10} z_{11}, z_9 z_{10}$	$1, 2\alpha_2^2 - 1 ^2, 2\alpha_2^2 - 1 ^2, \alpha_2 \beta_2^* + \beta_2 \alpha_2^* ^2$
$x_8, z_6 z_{10}$	$1, \alpha_2 \beta_2^* + \beta_2 \alpha_2^* ^2$
$x_9, x_9 z_{11}$	$1, 2\alpha_2^2 - 1 ^2$
$x_{10}, x_7 z_{11}$	$1, 2\alpha_2^2 - 1 ^2$
x_{11}	1
$z_1, x_1 z_2, x_3 z_5, z_2 x_4$	$1, \alpha_1 \beta_1^* - \beta_1 \alpha_1^* ^2$
$z_3, x_1 x_2, x_2 x_4$	$1, 1, 2\alpha_1^2 - 1 ^2$
$y_2, z_1 z_3$	$1, \alpha_1 \beta_1^* - \beta_1 \alpha_1^* ^2$
$x_1 x_5, x_4 x_5, z_3 z_6$	$1, 2\alpha_1^2 - 1 ^2, 2\alpha_1^2 - 1 ^2$

Table 5: Fidelity for errors with similar syndromes.

E	$F(\alpha_1, \beta_1, \alpha_2, \beta_2)$
$x_1 z_3, z_3 x_4$	1
$x_2 z_6, x_5 z_6, x_1 x_2, x_2 x_4$	$ 2\alpha_1^2 - 1 ^2$
$z_5 x_9, z_8 x_9, x_{10} z_{11}, x_7 z_{11}$	$ 2\alpha_2^2 - 1 ^2$
$z_1 z_5, z_4 z_5$	$ \alpha_1 \beta_1^* + \beta_1 \alpha_1^* ^2$
$z_9 z_{10}, z_6 z_{10}$	$ \alpha_2 \beta_2^* + \beta_2 \alpha_2^* ^2$
$x_1 z_2, z_1 z_3$	$ \alpha_1 \beta_1^* - \beta_1 \alpha_1^* ^2$
$x_5 x_6, z_4 z_7$	$1 \text{ ou } \alpha_1 \beta_1^* - \beta_1 \alpha_1^* ^2 \alpha_2 \beta_2^* - \beta_2 \alpha_2^* ^2$

Table 6: Some errors with similar fidelity.

E	$ Q_1\rangle \otimes Q_2\rangle$	$F(\theta, \phi)$	F_a
No error	$\langle \alpha_1, \beta_1 \langle \alpha_2, \beta_2 \rangle$	1	1
X_1	$\langle \beta_1, \alpha_1 \langle \alpha_2, \beta_2 \rangle$	$ \alpha_1 \beta_1^* + \beta_1 \alpha_1^* ^2 = \sin \theta_1 \cos \phi_1 ^2$	1/3
Z_1	$\langle \alpha_1, \beta_1 \langle \alpha_2, \beta_2 \rangle$	$ 2\alpha_1^2 - 1 ^2 = \cos^2 \theta_1$	1/3
Y_1	$\langle \beta_1, \alpha_1 \langle \alpha_2, \beta_2 \rangle$	$ \alpha_1 \beta_1^* - \beta_1 \alpha_1^* ^2 = \sin \theta_1 \sin \phi_1 ^2$	1/3
X_2	$\langle \alpha_1, \beta_1 \langle \beta_2, \alpha_2 \rangle$	$ \alpha_2 \beta_2^* + \beta_2 \alpha_2^* ^2 = \sin \theta_2 \cos \phi_2 ^2$	1/3
Z_2	$\langle \alpha_1, \beta_1 \langle \alpha_2, -\beta_2 \rangle$	$ 2\alpha_2^2 - 1 ^2 = \cos^2 \theta_2$	1/3
Y_2	$\langle \alpha_1, \beta_1 \langle \beta_2, \alpha_2 \rangle$	$ \alpha_2 \beta_2^* - \beta_2 \alpha_2^* ^2 = \sin \theta_2 \sin \phi_2 ^2$	1/3
$X_1 X_2$	$\langle \beta_1, \alpha_1 \langle \beta_2, \alpha_2 \rangle$	$ \sin \theta_1 \cos \phi_1 ^2 \sin \theta_2 \cos \phi_2 ^2$	1/9
$X_1 Z_2$	$\langle \beta_1, \alpha_1 \langle \alpha_2, -\beta_2 \rangle$	$ \sin \theta_1 \cos \phi_1 ^2 \cos^2 \theta_2$	1/9
$X_1 Y_2$	$\langle \beta_1, \alpha_1 \langle -\beta_2, \alpha_2 \rangle$	$ \sin \theta_1 \cos \phi_1 ^2 \sin \theta_2 \cos \phi_2 ^2$	1/9
$Z_1 X_2$	$\langle \alpha_1, \beta_1 \langle \beta_2, \alpha_2 \rangle$	$\cos^2 \theta_1 \sin \theta_2 \cos \phi_2 ^2$	1/9
$Z_1 Z_2$	$\langle \alpha_1, \beta_1 \langle \alpha_2, -\beta_2 \rangle$	$\cos^2 \theta_1 \cos^2 \theta_2$	1/9
$Z_1 Y_2$	$\langle \alpha_1, \beta_1 \langle \beta_2, \alpha_2 \rangle$	$\cos^2 \theta_1 \sin \theta_2 \sin \phi_2 ^2$	1/9
$Y_1 X_2$	$\langle \beta_1, \alpha_1 \langle \beta_2, \alpha_2 \rangle$	$ \sin \theta_1 \sin \phi_1 ^2 \sin \theta_2 \cos \phi_2 ^2$	1/9
$Y_1 Z_2$	$\langle \beta_1, \alpha_1 \langle \alpha_2, -\beta_2 \rangle$	$ \sin \theta_1 \sin \phi_1 ^2 \cos^2 \theta_2$	1/9
$Y_1 Y_2$	$\langle \beta_1, \alpha_1 \langle \beta_2, \alpha_2 \rangle$	$ \sin \theta_1 \sin \phi_1 ^2 \sin \theta_2 \sin \phi_2 ^2$	1/9

Table 7 : The fidelity $F(\theta, \phi)$ and average fidelity F_a for different possible errors on the useful qubits.

5.8 Generalisation

We can generalize the results obtained for the first processed qubits frame (q_2 to q_{12}) to all the frames because of the convolutional property of the code. In fact, when the first frame has been processed by the receiver (syndrome measurement and recovery), he can decode it and access the useful qubits states ($|Q_1\rangle, |Q_2\rangle$). Then, he starts the same processing for the next frame (q_{12} to q_{22}) to access the qubits states ($|Q_3\rangle, |Q_4\rangle$) and so on. We note that the receiver should not decode qubit q_{12} before he finished processing the second frame where it is used as controller and target. For k eleven-qubits frames containing $n = 2k$ useful qubits, each qubit $q_{2(5m+1)}$ ($m = 1, 2, \dots, k$) is common to frames m and $m+1$, but should be decoded with frame $m+1$. Table 8 gives for any qubits frame (q_l to q_{l+10} , $l = 2, \dots, 5n-8$) the fidelity obtained by the procedure described below.

E	$F_i(\alpha_i, \beta_i)$	F_{a_i}
$(X_j, Z_j, Y_j), (2 \leq j \leq 12, l \leq j \leq l+10, l = 12, 22, \dots)$	1	1
$(X_j X_k, Z_j Z_k, X_j Z_k), (2 \leq j, k \leq 12, l \leq j, k \leq l+10, l = 12, 22, \dots)$	1 ou $ 2\alpha_i^2 - 1 ^2$ ou $ \alpha_i \beta_i^* \pm \beta_i \alpha_i^* ^2$ $i = 1/2, 3/4, \dots, (n-1)/n$	1 ou $\frac{1}{3}$
$X_{m+6} X_{m+7}, Z_{m+5} Z_{m+8}, m = -1, 5, 10, \dots, 5(n-1), n \geq 2$	1 ou $ \alpha_i \beta_i^* - \beta_i \alpha_i^* ^2 \alpha_{i+1} \beta_{i+1}^* - \beta_{i+1} \alpha_{i+1}^* ^2$	1 ou $\frac{1}{9}$

Table 8 : Fidelity for double error in one frame containing two useful qubits.

We have considered only single or double error in one frame, triple error or more being very unlikely. We conclude from table 5 that for each frame containing two useful qubits, one of them is recovered in the majority of cases. The two decoded useful qubits (Q_i, Q_{i+1}) are both affected after decoding only if an error $X_{m+6}X_{m+7}$ or $Z_{m+5}Z_{m+8}$, $m = -1, 5, 10, \dots, 5(n-1)$ occurs in the channel. The fidelity for two useful qubits is given by :

$$F_i = \langle \Psi_t | \Psi_m \rangle \langle \Psi_m | \Psi_t \rangle = |\langle Q_{t_i} | Q_{m_i} \rangle|^2 |\langle Q_{t_{i+1}} | Q_{m_{(i+1)}} \rangle|^2 \quad (10)$$

For n useful qubits transmitted, the fidelity is given by the expression :

$$F = \Pi_i F_i = \Pi_i |\langle Q_{t_i} | Q_{m_i} \rangle|^2 |\langle Q_{t_{i+1}} | Q_{m_{(i+1)}} \rangle|^2, \quad i = 1, 3, \dots, n-1 \quad (11)$$

The total average fidelity is:

$$F_a = \Pi_i F_{a_i}, \quad F_{a_i} = 1 \text{ or } \left(\frac{1}{3}\right) \text{ or } \left(\frac{1}{9}\right), \quad i = 1, 3, \dots, n-1 \quad (12)$$

Suppose that a number n_1 among the processed qubits frames are recovered with average fidelity $F_{a_i} = 1/3$, and that a number n_2 of the frames are recovered with average fidelity $F_{a_i} = 1/9$. Then the $(\frac{n}{2} - n_1 - n_2)$ remaining frames are recovered with average fidelity $F_{a_i} = 1$ and the total average fidelity is :

$$F_a = (1)^{\left(\frac{n}{2} - n_1 - n_2\right)} \left(\frac{1}{3}\right)^{n_1} \left(\frac{1}{9}\right)^{n_2} \quad (13)$$

We define the probability P_1 and P_2 corresponding respectively to average fidelity $F_{a_i} = 1/3$ and $F_{a_i} = 1/9$ as :

$$P_1 = \frac{n_1}{n/2} \quad \text{and} \quad P_2 = \frac{n_2}{n/2} \quad (14)$$

We obtain the average fidelity as function of P_1 and P_2 :

$$\begin{aligned} F_a(P_1, P_2) &= (1)^{\frac{n}{2}(1-P_1-P_2)} \left(\frac{1}{3}\right)^{P_1 n/2} \left(\frac{1}{9}\right)^{P_2 n/2} \\ F_a(P_1, P_2) &= \left(\frac{1}{3^n}\right)^{(P_1/2)+P_2} \end{aligned} \quad (15)$$

We can suppose $P_2 \ll P_1$ as the average fidelity $F_{a_i} = 1/9$ concerns only two double channel errors ($X_{m+6}X_{m+7}$ or $Z_{m+5}Z_{m+8}$, $m = -1, 5, 10, \dots, 5(n-1)$). The figure 1 shows the average fidelity for $(0 \leq P_1 = P \leq 1, P_2 \cong 0)$ for different number n of useful qubits. We see that the fidelity decreases rapidly when P increases and vanishes for lower value of P if n is higher. Finally, it is close to one for very low values of P ($\lesssim 10^{-2}$) and is better when n is lower for fixed value of P .

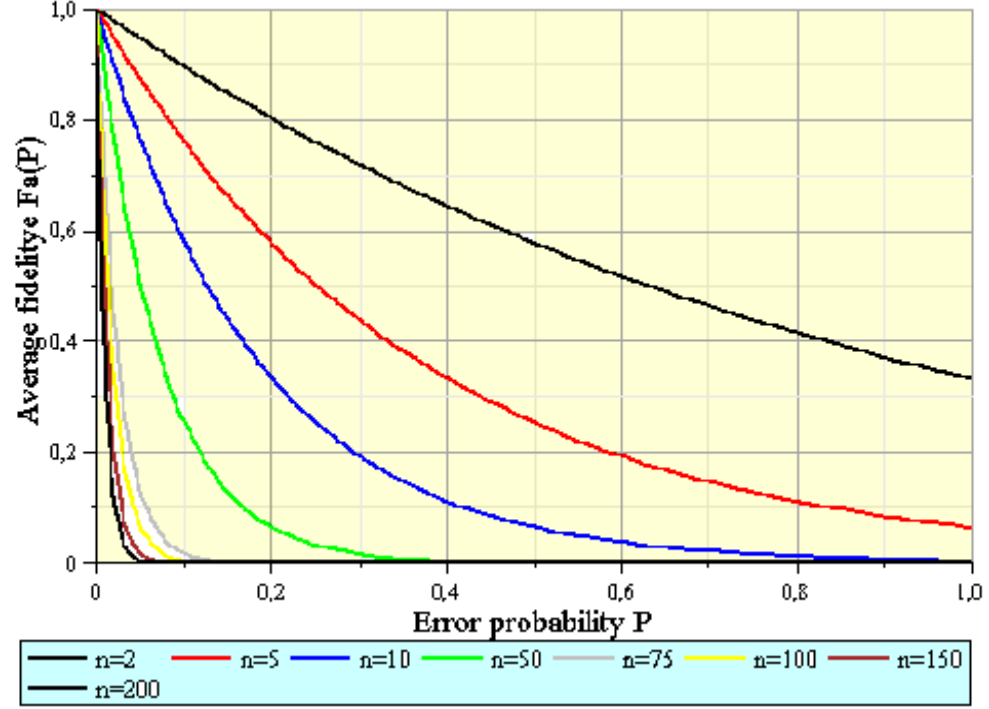


Figure 1: Evolution of the average fidelity versus the probability P of occurring a double channel error $X_j X_k$, $Z_j Z_k$ or $X_j Z_k$ for different number n of useful qubits.

6 Conclusion

We investigated in this paper a five-qubits quantum convolutional error correcting code. We described the encoding, error correction and decoding of the first sent qubits frame. The processing of the next qubits frame is similar because of the convolutional property of the code. The simulated coding circuit, gives in a short time the correct coded state. Our proposed correction procedure, performs the required measurement syndromes only, thus avoiding the decoherence phenomenon. Finally, we showed in this work that the fidelity in the case of double errors having same syndromes as single error is close to one only if the probability of these errors is very low.

Acknowledgement

We would like to thank very much Mrs S.Fritsch and T.Radtke for providing us with the version 4 (2008) of Feynman Program.

7 References

- [1] M.A. Nielsen, I.L. Chuang: "Quantum computation and quantum information", Cambridge University Press, UK, 2000.
- [2] Daniel Gottesman, "An Introduction to Quantum Error Correction", arXiv:0904.2557v1, [quant-ph], 16 Apr 2009.
- [3] Eleanor Rieffel and W.Polak, "An introduction to quantum computing for non-physicists", permissions@acm.org, 2000.
- [4] H.Olivier, J.P.Tillich, "Quantum Convolutional code: Fundamentals ", arXiv:quant-ph/0401134v11, 2004.
- [5] Mark McMahon Wilde, "Quantum Coding with Entanglement", arXiv:0806.4214v1, [quant-ph], 25 Jun 2008
- [6] H.Olivier, "Elements of quantum information theory, decoherence and error correcting codes", thesis, INRIA-CODES, 2004.
- [7] H.Olivier, J.P.Tillich, "Description of a quantum Convolutional code ", arXiv:quant-ph/0304189v2, 15 May 2003
- [8] T.Radtke, S.Fritzsche: 'Simulation of n-qubits quantum systems', I. Quantum gates and registers, Computer Physics Communications, 2005.
- [9] T.Radtke, S.Fritzsche: 'Simulation of n-qubits quantum systems', II. Quantum states, Computer Physics Communications, 2006.
- [10] T.Radtke, S.Fritzsche: 'Simulation of n-qubits quantum systems', III. Quantum operations, Computer Physics Communications, 2007.
- [11] T.Radtke, S.Fritzsche: 'Simulation of n-qubits quantum systems',IV. Parametrizations of quantum states, Computer Physics Communications, 2008.
- [12] CPC Program Library, Queen's University of Belfast, N.Ireland, Apr 2008.

8 Appendix: Simulation of the five qubits convolutional code with Feynman Program.

8.1 Coding circuit

The coding circuit is simulated in accordance with the diagram of figure 11, page 22 in reference [4]. The boxes H and Z represent the Hadamard and the Z gates respectively. The gate H transforms each simple state $|0\rangle$ in a superposed state $|+\rangle$ and the gate Z change $|+\rangle$ to $|-\rangle$. The states $|Q_1\rangle$, $|Q_2\rangle$ and $|Q_3\rangle$ are not concerned by the gate H as they are already in a superposed state $|\Psi_i\rangle = \alpha_i |0\rangle + \beta_i |1\rangle$. The simulation begins by calling Feynman program and Linear Algebra. The state Qo is the tensor product of five qubits state (qubit 2 to qubit 6) after applying H and Z on them ($\langle 1,0 \rangle$ is the state $|0\rangle$). The five qubits state Q8 is obtained after applying all the controlled gate X, Y and Z except the two last controlled Z (CZ) (layer 5 and 6) on qubit 6 as we have before a CZ gate (layer 4) applied on it with qubit 7 as controller. We note that Q2 and Q3 correspond to the controlled-Y gate applied on qubit 5 with qubit 2 as controller (we write [1,4] instead of [2,5] because the first qubit is not sent through the communication channel as no gate is applied on it). Coding from qubits 7 to qubit 12 begins by applying the H and Z gates (layer 1 and 2) to obtain the six qubits state Q9. The six qubits state Q20 is obtained after applying all the controlled gate X, Y and Z except the CZ gate (layer 5) on qubit 7 as we have before (layer 4) a CZ gate on qubit 6 with qubit 7 as controller. Applying the CZ gate will change its value and introduces an error. We note that in state Q10, for example, [5,4] indicates the position 5 and 4 in the six qubits system we are encoding, but corresponds to qubit 11 (controller) and qubit 10 (target) in the twelve qubits system. In coding from qubit 2 to qubit 12 we construct the tensor product Q21 of the states Q8 and Q20 and then apply on it the four remaining CZ gates ([6,5] in layer 4, [2,5] and [7,6] in layer 5 and [3,5] in layer 6) to obtain the completely coded eleven qubits state Q25. We proceed in the same manner for coding from qubit 13 to qubit 17 to obtain the seventeen qubits coded state Q26. Below, the coding circuit simulation with first some explained instructions:

Z:= Feynman_quantum_operator("Z"): Matrice for the phase flip gate.
H:=Feynman_quantum_operator("H"): Matrice for the Hadamard gate.
Qo:=Feynman_evaluate("Kronecker product",Z.H.<1,0>,Z.H.<1,0>,H.<1,0>,<a1,b1>,H.<1,0>): Tensor product of five qubits states ($\langle 1,0 \rangle = |0\rangle$).
Q1:=Feynman_quantum_operator(5,"cn",[5,4]).Qo: Conrolled-not gate on the five qubits system state Qo to obtain the state Q1. The fifth qubit is the controller and the fourth one the target.
Q2:=Feynman_quantum_operator(5,"cz",[1,4]).Q1: Conrolled-Z gate on the five qubits system Q1 to obtain the state Q2. The first qubit is the controller and the fourth one the target.:

```

> with(Feynman): with(LinearAlgebra): Digits:=20:
Z:=Feynman_quantum_operator("Z"): H:=Feynman_quantum_operator("H"):
# Coding from qubit 2 to qubit 6
Qo:=Feynman_evaluate("Kronecker product",Z.H.<1,0>,Z.H.<1,0>,H.<1,0>,
<a1,b1>,H.<1,0>):
Q1:=Feynman_quantum_operator(5,"cn",[5,4]).Qo:
Q2:=Feynman_quantum_operator(5,"cz",[1,4]).Q1:
Q31:=I.Feynman_quantum_operator(5,"cn",[1,4]).Q2:
Q3:=I.Feynman_quantum_operator(5,"cz",[1,5]).Q31:
Q4:=Feynman_quantum_operator(5,"cz",[2,1]).Q3:
Q5:=Feynman_quantum_operator(5,"cz",[2,4]).Q4:
Q6:=I.Feynman_quantum_operator(5,"cn",[2,4]).Q5:
Q7:=Feynman_quantum_operator(5,"cz",[3,2]).Q6:
Q8:=Feynman_quantum_operator(5,"cn",[3,4]).Q7:
# Coding from qubit 7 to qubit 12
Q9:=Feynman_evaluate("Kronecker product",Z.H.<1,0>,Z.H.<1,0>,H.<1,0>,
<a2,b2>,H.<1,0>,Z.H.<1,0>):
Q10:=Feynman_quantum_operator(6,"cn",[5,4]).Q9:
Q11:=Feynman_quantum_operator(6,"cz",[6,5]).Q10:
Q12:=Feynman_quantum_operator(6,"cz",[1,4]).Q11:
Q13:=I.Feynman_quantum_operator(6,"cn",[1,4]).Q12:
Q14:=Feynman_quantum_operator(6,"cz",[1,5]).Q13:
Q15:=Feynman_quantum_operator(6,"cz",[2,4]).Q14:
Q16:=I.Feynman_quantum_operator(6,"cn",[2,4]).Q15:
Q17:=Feynman_quantum_operator(6,"cz",[2,5]).Q16:
Q18:=Feynman_quantum_operator(6,"cz",[3,2]).Q17:
Q19:=Feynman_quantum_operator(6,"cn",[3,4]).Q18:
Q20:=Feynman_quantum_operator(6,"cz",[3,5]).Q19:
# Coding from qubit 1 to qubit 12
Q21:=Feynman_evaluate("Kronecker product",Q8,Q20):
Q22:=Feynman_quantum_operator(11,"cz",[6,5]).Q21:
Q23:=Feynman_quantum_operator(11,"cz",[2,5]).Q22:
Q24:=Feynman_quantum_operator(11,"cz",[7,6]).Q23:
Q25:=Feynman_quantum_operator(11,"cz",[3,5]).Q24:
# Coding from qubit 13 to qubit 17
Q251:=Feynman_evaluate("Kronecker product",Z.H.<1,0>,H.<1,0>,<a3,b3>,
H.<1,0>,Z.H.<1,0>):
Q252:=Feynman_quantum_operator(15,"cn",[15,14]).Q251:
# Coding from qubit 2 to qubit 16
Q253:=Feynman_evaluate("Kronecker product",Q25,Q252):
Q254:=Feynman_quantum_operator(15,"cz",[11,14]).Q253:
Q255:=I.Feynman_quantum_operator(15,"cn",[11,14]).Q254:
Q256:=Feynman_quantum_operator(15,"cz",[11,15]).Q255:
Q257:=Feynman_quantum_operator(15,"cz",[12,11]).Q256:
Q258:=Feynman_quantum_operator(15,"cz",[12,14]).Q257:
Q259:=I.Feynman_quantum_operator(15,"cn",[12,14]).Q258:

```

```

Q260:=Feynman_quantum_operator(15,"cz",[12,15]).Q259:
Q261:=Feynman_quantum_operator(15,"cz",[13,12]).Q260:
Q262:=Feynman_quantum_operator(15,"cn",[13,14]).Q261:
Q26:=Feynman_quantum_operator(15,"cz",[13,15]).Q262:

```

8.2 Stabilisation of the coded state $|Q25\rangle$

After constructing the eleven qubits coded state $|Q25\rangle$, we have to check if our coding procedure is correct. Then, we apply the eight generators M_1 to M_8 and verify that $M_i|Q25\rangle = |Q25\rangle$ for $1 \leq i \leq 8$. The generator $M_1=ZXXZ$ is applied in the following manner: the first Z gate is suppressed as the first qubit is not sent, the second qubit ([1] instead of [2]) is bit flipped to obtain the state S1a and the third and fourth ([2,3] instead of [3,4]) qubits are respectively bit and phase flipped to obtain the state S1. Dividing this generator in two parts allows reducing the running time. Finally, we verify the equality $|S1\rangle = |Q25\rangle$ using a message "M1 true". We process in the same manner for all the other generators and obtain the same message "M_i true", witch is the proof that the coding procedure is correct. Below, the simulation of the stabilisation with first some explained instructions:

S1a:=Feynman_quantum_operator(11,"X",[1]).Q25: Application of a bit flip on the first qubit state in an eleven qubit system. We note that this operator X is at the second position in the generator $M_1=ZXXZ$. It occupies the first position because the first qubit has been suppressed.

S1:=Feynman_quantum_operator(11,"XZ",[2,3]).S1a: Application of a bit and phase flip on respectively the second and third qubit . It is in fact the third and fourth operator of the generator M_1 . All the generators were divided in two parts to reduce the running time.

```

# Generator M1
S1a:=Feynman_quantum_operator(11,"X",[1]).Q25:
S1:=Feynman_quantum_operator(11,"XZ",[2,3]).S1a:
if evalb(Equal(S1,Q25))=true then print("M1 true") else print("M1 false");fi:
# Generator M2
S2a:=Feynman_quantum_operator(11,"ZX",[1,2]).Q25:
S2:=Feynman_quantum_operator(11,"XZ",[3,4]).S2a:
if evalb(Equal(S2,Q25))=true then print("M2 true") else print("M2 false");fi:
# Generator M3
S3a:=Feynman_quantum_operator(11,"ZX",[2,3]).Q25:
S3:=Feynman_quantum_operator(11,"XZ",[4,5]).S3a:
if evalb(Equal(S3,Q25))=true then print("M3 true") else print("M3 false");fi:
# Generator M4
S4a:=Feynman_quantum_operator(11,"ZX",[3,4]).Q25:
S4:=Feynman_quantum_operator(11,"XZ",[5,6]).S4a:
if evalb(Equal(S4,Q25))=true then print("M4 true") else print("M4 false");fi:
# Generator M5
S5a:=Feynman_quantum_operator(11,"ZX",[5,6]).Q25:

```

```

S5:=Feynman_quantum_operator(11,"XZ",[7,8]).S5a:
if evalb(Equal(S5,Q25))=true then print("M5 true") else print("M5 false");fi:
# Generator M6
S6a:=Feynman_quantum_operator(11,"ZX",[6,7]).Q25:
S6:=Feynman_quantum_operator(11,"XZ",[8,9]).S6a:
if evalb(Equal(S6,Q25))=true then print("M6 true") else print("M6 false");fi:
# Generator M7
S7a:=Feynman_quantum_operator(11,"ZX",[7,8]).Q25:
S7:=Feynman_quantum_operator(11,"XZ",[9,10]).S7a:
if evalb(Equal(S7,Q25))=true then print("M7 true") else print("M7 false");fi:
# Generator M8
S8a:=Feynman_quantum_operator(11,"ZX",[8,9]).Q25:
S8:=Feynman_quantum_operator(11,"XZ",[10,11]).S8a:
if evalb(Equal(S8,Q25))=true then print("M8 true") else print("M8 false");fi:

```

Below the given output : Welcome to Feynman (April 2008)

```

"M1 true"
"M2 true"
"M3 true"
"M4 true"
"M5 true"
"M6 true"
"M7 true"
"M8 true"

```

8.3 Syndromes extraction for generators M₁ to M₈

First, we introduce an error on the $|Q25\rangle$ state, for example a single bit flip on the first qubit ($[1]$) or a double bit flip on the first and the eight qubits ($[1,8]$). We add a superposed qubit state ($H.<1,0>=|+\rangle$) by constructing a tensor product with the affected state $|Q27\rangle$. This added qubit will allow the receiver to make sounding in the coded state and know if it is affected by an error without disturbing it (see section 4.3). The generator M₁ is conditionnally applied in two parts ("c" means conditionnal application of X or XZ on second $[1,2]$ or third and fourth $[1,3,4]$ qubits with the first added qubit as controller) to obtain a twelve qubits state $|Q281b\rangle$. The application of H on the first qubit ($12,"H",[1]$) will transform it in a simple state $|0\rangle$ or $|1\rangle$ in the twelve qubit state $|Q281c\rangle$. Finally, applying the Z gate on the added qubit will give $|0\rangle$ ($|Q281\rangle=|Q281c\rangle$) or $|1\rangle$ ($|Q281\rangle=-|Q281c\rangle$). Below, the simulation of this syndrome extraction procedure with first some explained instructions:

Q27:=Feynman_quantum_operator(11,"X",[1]).Q25: A bit flip occurs on the first qubit in the communication channel.

Q28:=Feynman_evaluate("Kronecker product",H.<1,0>,Q27): A superposed qubit state is added to the system.

Q281a:=Feynman_quantum_operator(12,"c","X",[1,2]).Q28:

Q281b:=Feynman_quantum_operator(12,"c","XZ",[1,3,4]).Q281a: Generator M_1 is conditionally applied on the affected coded state.

Q281c:=Feynman_quantum_operator(12,"H",[1]).Q281b: Application of H on the added qubit.

Q281:=Feynman_quantum_operator(12,"Z",[1]).Q281c: Application of the Z gate on the added qubit.

Q281a:=Feynman_quantum_operator(12,"c","X",[1,2]).Q27: X gate is conditionally applied ("c") on qubit 2 (if the first qubit is 1) in a twelve qubits system.

Q281b:=Feynman_quantum_operator(12,"c","XZ",[1,3,4]).Q281a : Conditional application of the gates \bar{X} and Z on respectively the third and fourth qubit. We note that we conditionally applied the generator M_1 .

Q281c:=Feynman_quantum_operator(12,"H",[1]).Q281b: We apply again H on the first qubit to obtain a simple state $H|+\rangle = |0\rangle$ or $H|-\rangle = |1\rangle$.

Q281:=Feynman_quantum_operator(12,"Z",[1]).Q281c: The application of the Z gate on the first qubit will give $Z|0\rangle = |0\rangle$ or $Z|1\rangle = -|1\rangle$.

```
# Single error between qubit 2 and qubit 12
Q27:=Feynman_quantum_operator(11,"X",[1]).Q25:
# Double error between qubit 2 and qubit 12
Q27:=Feynman_quantum_operator(11,"XX",[1,8]).Q25:
# Adding one superposed qubit state
Q28:=Feynman_evaluate("Kronecker product",H.<1,0>,Q27):
# Generator  $M_1$ 
# Conditional application of  $M_1$ 
Q281a:=Feynman_quantum_operator(12,"c","X",[1,2]).Q28:
Q281b:=Feynman_quantum_operator(12,"c","XZ",[1,3,4]).Q281a:
# Application of H and Z on added qubit
Q281c:=Feynman_quantum_operator(12,"H",[1]).Q281b:
Q281:=Feynman_quantum_operator(12,"Z",[1]).Q281c:
# Generator  $M_2$ 
Q282a:=Feynman_quantum_operator(12,"c","ZX",[1,2,3]).Q28:
Q282b:=Feynman_quantum_operator(12,"c","XZ",[1,4,5]).Q282a:
Q282c:=Feynman_quantum_operator(12,"H",[1]).Q282b:
Q282:=Feynman_quantum_operator(12,"Z",[1]).Q282c:
# Generator  $M_3$ 
Q283a:=Feynman_quantum_operator(12,"c","ZX",[1,3,4]).Q28:
Q283b:=Feynman_quantum_operator(12,"c","XZ",[1,5,6]).Q283a:
Q283c:=Feynman_quantum_operator(12,"H",[1]).Q283b:
Q283:=Feynman_quantum_operator(12,"Z",[1]).Q283c:
# Generator  $M_4$ 
Q284a:=Feynman_quantum_operator(12,"c","ZX",[1,4,5]).Q28:
Q284b:=Feynman_quantum_operator(12,"c","XZ",[1,6,7]).Q284a:
Q284c:=Feynman_quantum_operator(12,"H",[1]).Q284b:
Q284:=Feynman_quantum_operator(12,"Z",[1]).Q284c:
```

```

# Generator M5
Q285a:=Feynman_quantum_operator(12,"c","ZX",[1,6,7]).Q28:
Q285b:=Feynman_quantum_operator(12,"c","XZ",[1,8,9]).Q285a:
Q285c:=Feynman_quantum_operator(12,"H",[1]).Q285b:
Q285:=Feynman_quantum_operator(12,"Z",[1]).Q285c:
# Generator M6
Q286a:=Feynman_quantum_operator(12,"c","ZX",[1,7,8]).Q28:
Q286b:=Feynman_quantum_operator(12,"c","XZ",[1,9,10]).Q286a:
Q286c:=Feynman_quantum_operator(12,"H",[1]).Q286b:
Q286:=Feynman_quantum_operator(12,"Z",[1]).Q286c:
# Generator M7
Q287a:=Feynman_quantum_operator(12,"c","ZX",[1,8,9]).Q28:
Q287b:=Feynman_quantum_operator(12,"c","XZ",[1,10,11]).Q287a:
Q287c:=Feynman_quantum_operator(12,"H",[1]).Q287b:
Q287:=Feynman_quantum_operator(12,"Z",[1]).Q287c:
# Generator M8
Q288a:=Feynman_quantum_operator(12,"c","ZX",[1,9,10]).Q28:
Q288b:=Feynman_quantum_operator(12,"c","XZ",[1,11,12]).Q288a:
Q288c:=Feynman_quantum_operator(12,"H",[1]).Q288b:
Q288:=Feynman_quantum_operator(12,"Z",[1]).Q288c:
# Generator M9
# Error
Q27a:=Feynman_quantum_operator(16,"X",[1]).Q26:
# Adding one superposed qubit state
Q28a:=Feynman_evaluate("Kronecker product",H.<1,0>,Q27a):
Q289a:=Feynman_quantum_operator(17,"c","ZX",[1,10,11]).Q28a:
Q289b:=Feynman_quantum_operator(17,"c","XZ",[1,12,13]).Q289a:
Q289c:=Feynman_quantum_operator(17,"H",[1]).Q289b:
Q289:=Feynman_quantum_operator(17,"Z",[1]).Q289c:
# Generator M10
Q290a:=Feynman_quantum_operator(17,"c","ZX",[1,11,12]).Q28a:
Q290b:=Feynman_quantum_operator(17,"c","XZ",[1,13,14]).Q290a:
Q290c:=Feynman_quantum_operator(17,"H",[1]).Q290b:
Q290:=Feynman_quantum_operator(17,"Z",[1]).Q290c:

```

8.4 Detection and correction from qubit 2 to qubit 12

The tables 2,3,4 present the syndromes in order to facilitate the error detection. The errors Y_3 and Z_2Z_4 are put in the first row because they contain the first four value equal to one in their syndrome. The error Z_2X_5 is in the second position because it has the first three values and the value for M_5 equal to one in their syndromes and so on untill the error Z_3X_6 witch has a zero for M_1 in its syndrome. We continue to place the errors so that the values equal to one in the syndromes are shifted to the right untill the last one X_1X_4 witch has no value equal to one in its syndrome. Then, for each error E, we search in the tables

other errors having similar syndromes and determine the generators to be measured to identify this error. We present below the detection procedure for the two first (table 2) and the two final (table 4) rows with first some explained instructions:

```

if evalb(Equal(Q281,-Q281c))=true and evalb(Equal(Q282,-Q282c))=true
and evalb(Equal(Q283,-Q283c))=true and evalb(Equal(Q284,-Q284c))= true
then print(" Y3 or Z2Z4 error detected"):fi: We measure the eigenvalues al-
lowing to identify two possible errors. The most likely one is choosen depending on
the transmitting channel (a single error is generally more likely than a double one).
if evalb(Equal(Q281,-Q281c))=true and evalb(Equal(Q282,-Q282c))=true
and evalb(Equal(Q283,-Q283c))=true and evalb(Equal(Q285,-Q285c))=true
then Q29:=Feynman_quantum_operator(11,"ZX",[2,5]).Q27: fi:
if evalb(Equal(Q29,Q25))=true then print("Z2X5 error detected"):fi: The
measured syndromes allow an error identification witch gives fidelity equal to one.

```

```

# Detection giving a list of two errors (table 2)
if evalb(Equal(Q281,-Q281c))=true and evalb(Equal(Q282,-Q282c))=true
and evalb(Equal(Q283,-Q283c))=true and evalb(Equal(Q284,-Q284c))=true
then print(" Y3 or Z2Z4 error detected"):fi:
# Correction of error perfectly identified (table 2)
if evalb(Equal(Q281,-Q281c))=true and evalb(Equal(Q282,-Q282c))=true
and evalb(Equal(Q283,-Q283c))=true and evalb(Equal(Q285,-Q285c))=true
then Q29:=Feynman_quantum_operator(11,"ZX",[2,5]).Q27: fi:
if evalb(Equal(Q29,Q25))=true then print("Error Z2X5 corrected"):fi:
# Correction of error perfectly identified (table 4)
if evalb(Equal(Q285,Q285c))=true and evalb(Equal(Q287,Q287c))=true
and evalb(Equal(Q288,Q288c))=true and evalb(Equal(Q289,Q289c))=true
and evalb(Equal(Q290,-Q290c))=true
then Q29:=Feynman_quantum_operator(11,"ZX",[10,11]).Q27: fi:
if evalb(Equal(Q29,Q25))=true then print("Error Z10X11 corected"):fi:
# Detection giving a possibility of no error or one error (table 4)
if evalb(Equal(Q281,Q281c))=true and evalb(Equal(Q282,Q282c))=true
and evalb(Equal(Q283,Q283c))=true and evalb(Equal(Q284,Q284c))=true
and evalb(Equal(Q285,Q285c))=true and evalb(Equal(Q286,Q286c))=true
and evalb(Equal(Q287,Q287c))=true and evalb(Equal(Q288,Q288c))=true
and evalb(Equal(Q289,Q289c))=true and evalb(Equal(Q290,Q290c))=true
then print("No error or error X1X4 detected"):fi:

```

Below some obtained outputs : Welcome to Feynman (April 2008)

```

" Y3 or Z2Z4 error detected "
"Error Z2X5 detected "

```

8.5 Decoding from qubit 2 to qubit 12

The decoding procedure is simulated using the coding circuit of the figure 2 but the six layers of gates are considered in the inverse order. We note that all the instructions concern an eleven qubits system with a high running time. Perhaps that Feynman Program offers the opportunities or needs modifications to decode by small qubits groups. Below the simulation with first some explained instructions:

Q30:=Feynman_quantum_operator(11,"cz",[3,2]).Q29: The gate Z is applied on the second qubit if the third qubit is one.
Q31:=Feynman_quantum_operator(11,"cn",[3,4]).Q30: The gate X is applied on the fourth qubit if the third qubit is one.
Q47:=Feynman_quantum_operator(11,"Z",[10]).Q46: The gate Z is applied on the tenth qubit.
Q59:=Feynman_quantum_operator(11,"H",[1]).Q58: The gate H is applied on the first qubit. We can apply at one time the gates H, by the next instruction:
Feynman_quantum_operator(11,"HHHHHHHHH",[1,2,3,5,6,7,8,10,11]).
We have avoided this procedure because it takes too much time.

```
# Sixth layer
Q30:=Feynman_quantum_operator(11,"cz",[3,2]).Q29:
Q31:=Feynman_quantum_operator(11,"cn",[3,4]).Q30:
Q32:=Feynman_quantum_operator(11,"cz",[3,5]).Q31:
Q33:=Feynman_quantum_operator(11,"cz",[8,7]).Q32:
Q34:=Feynman_quantum_operator(11,"cn",[8,9]).Q33:
Q35:=Feynman_quantum_operator(11,"cz",[8,10]).Q34:
# Fifth layer
Q36:=Feynman_quantum_operator(11,"cz",[2,1]).Q35:
Q37:=Feynman_quantum_operator(11,"cz",[2,4]).Q36:
Q38:=-I.Feynman_quantum_operator(11,"cn",[2,4]).Q37:
Q39:=Feynman_quantum_operator(11,"cz",[2,5]).Q38:
Q40:=Feynman_quantum_operator(11,"cz",[7,6]).Q39:
Q41:=Feynman_quantum_operator(11,"cz",[7,9]).Q40:
Q42:=-I.Feynman_quantum_operator(11,"cn",[7,9]).Q41:
Q43:=Feynman_quantum_operator(11,"cz",[7,10]).Q42:
# Fourth layer
Q44:=Feynman_quantum_operator(11,"cz",[6,5]).Q43:
Q45:=Feynman_quantum_operator(11,"cz",[6,9]).Q44:
Q46:=-I.Feynman_quantum_operator(11,"cn",[6,9]).Q45:
Q47:=Feynman_quantum_operator(11,"cz",[6,10]).Q46:
Q48:=Feynman_quantum_operator(11,"cz",[1,4]).Q47:
Q49:=-I.Feynman_quantum_operator(11,"cn",[1,4]).Q48:
```

```

Q50:=Feynman_quantum_operator(11,"cz",[1,5]).Q49:
Q51:=Feynman_quantum_operator(11,"cz",[11,10]).Q50:
# Third layer
Q52:=Feynman_quantum_operator(11,"cn",[5,4]).Q51:
Q53:=Feynman_quantum_operator(11,"cn",[10,9]).Q52:
# Second layer
Q54:=Feynman_quantum_operator(11,"Z",[1]).Q53:
Q55:=Feynman_quantum_operator(11,"Z",[2]).Q54:
Q56:=Feynman_quantum_operator(11,"Z",[6]).Q55:
Q57:=Feynman_quantum_operator(11,"Z",[7]).Q56:
Q58:=Feynman_quantum_operator(11,"Z",[11]).Q57:
# First layer
Q59:=Feynman_quantum_operator(11,"H",[1]).Q58:
Q60:=Feynman_quantum_operator(11,"H",[2]).Q59:
Q61:=Feynman_quantum_operator(11,"H",[3]).Q60:
Q62:=Feynman_quantum_operator(11,"H",[5]).Q61:
Q63:=Feynman_quantum_operator(11,"H",[6]).Q62:
Q64:=Feynman_quantum_operator(11,"H",[7]).Q63:
Q65:=Feynman_quantum_operator(11,"H",[8]).Q64:
Q66:=Feynman_quantum_operator(11,"H",[10]).Q65:
Q67:=Feynman_quantum_operator(11,"H",[11]).Q66:

```