Universidad ORT Uruguay

Facultad de Ingeniería

Error analysis in quantum circuits:

The Shor code

Entregado como requisito para la obtención del título

Master en Ingeniería

Ilan Cohn - 160866

Tutores:

André Fonseca De Oliveira

Efrain Buksman Hollander

2023

Declaración de Autoría

Yo, Ilan Cohn Noachas, declaro que el trabajo que se presenta en esa obra es de mi propia mano. Puedo asegurar que:

- La obra fue producida en su totalidad mientras realizaba el desarrollo de tesis del Master en Ingeniería;
- Cuando he consultado el trabajo publicado por otros, lo he atribuido con claridad;
- Cuando he citado obras de otros, he indicado las fuentes. Con excepción de estas citas, la obra es enteramente mía;
- En la obra, he acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, he explicado claramente qué fue contribuido por otros, y qué fue contribuido por mi;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Ilan Cohn Noachas
14 de Agosto de 2023

Acknowledgements

To my advisors, André Fonseca de Oliveira and Efrain Buksman Hollander, for introducing me to quantum computing and providing unwavering support, insightful discussions, guidance, encouragement, and wisdom throughout the years.

To my uncle, Gabriel Noachas, without whom I would have definitely not gotten here, and whose constant encouragement to pursue deep critical thinking has been invaluable throughout my whole life.

To my grandparents, Salomón Noachas and Rebeca Eisenberg, for always encouraging me to study. Your guidance and support have shaped my journey throughout life.

To my parents and sister, Iael Noachas, Morris Saul and Galit Cohn, who have always encouraged me to chase my dreams and shown me that it can be achieved, thank you from the bottom of my heart.

To the best girlfriend someone could have, Natalie Golffed, for her immense support and infinite love during this journey including lots of sleepless nights, late dinners and missing nights out. I could not have done this without you. To our dog-queen Sheine, who kept me company until I went to sleep.

To my nephew, Matías Eliazer, who has shown me how wonderful it is to be able to spend time with someone who is new to this world and makes me want to build a better future.

Lastly, I would like to thank each one of my family members for their endless support, especially during the most challenging times. Your unconditional love towards me has been, and will continue to be, my strength. I love you all.

Abstract

English

This work provides an insight into quantum error modelling, analysis and correction. In particular, we provide a deep analysis for the Shor correcting code. We introduce a custom efficient algorithm to simplify and find algebraic formulas, designed specifically for quantum computing. We show how this algorithm can be used to analyse the resiliency of the Shor code to imperfect quantum gates and channels. We also provide analytical formulas for the resulting states for different error models, and the regions in which it is better to use the code instead of sending the qubit without encoding based on the expected value of the fidelity.

Español

Este trabajo ofrece una perspectiva sobre el modelado, análisis y corrección de errores cuánticos. En particular, proporcionamos un estudio profundo del código corrector de Shor. Introducimos un algoritmo personalizado y eficiente para simplificar y encontrar fórmulas algebraicas, diseñado específicamente para la computación cuántica. Mostramos cómo este algoritmo se puede utilizar para analizar la resiliencia del código de Shor a puertas y canales cuánticos imperfectos. También proporcionamos fórmulas analíticas para los estados resultantes en diferentes modelos de error, y las regiones en las que es mejor utilizar el código en lugar de enviar el qubit sin codificar, basándonos en el valor esperado de la fidelidad.

Keywords

Quantum Error Analysis, Shor Code Error Modelling, Quantum Linear Codes

Table of contents

1. Introduction

Quantum computers have been successfully applied to material sciences, quantum chemistry, optimization problems, and machine learning [1, 2, 3, 4]. While these results have been reported, near-term quantum computers based on circuit models have been built as such computers are still fragile against quantum noise effects. This era is called the noisy intermediate-scale quantum (NISQ) era, and the devices are called NISQ devices [5].

In this NISQ era, quantum noise effects cannot be neglected and are a major obstacle for performing quantum computation. The traditional schemes for mitigating the effect of errors is quantum error correction and fault-tolerant computing [6, 7]. A problem may arise if the imperfect implementation of such techniques introduce more error than a certain threshold; the accumulation of a big-enough error might end up bearing worse results than not using these techniques at all.

One of the first quantum error-correcting codes to have been proposed is the Shor code [8]. In this thesis, we will model imperfect gates and channels for an implementation of the Shor code with different error models and show the resulting state of the circuit (with a closed formula). We will also provide a criteria to define when is it better to send the information without encoding, with their corresponding regions for each model.

The document is structured in the following way. In chapter 1, an introduction to quantum computing and its postulates is formulated. In chapter 2 an introduction to error modeling is given, along with several error models. In chapter 3, we introduce classical and quantum linear codes. In chapter 4, we explain the Shor code. In chapter 5, we show the analysis and novel analytical results for modelling the Shor code with imperfect gates and channels. In chapter 6, we give some discussions and conclusions. The methodology proposed for reduced computational complexity can be found in Annex 1, while the entire formulas for the analytical results can be found in Annex 2.

The following introduction is a summary from [9, 10, 11].

1.1 Postulate 1: The state spaces

Postulate 1 [9]: *"Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space."*

1.1.1 The qubit

The basic unit in quantum computing is the quantum bit (*qubit*). A qubit (using Dirac's bra-ket notation) is represented by

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{1.1}$$

where $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$ and

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{1.2}$$

Since the states are normalized, a qubit can be represented as a point in a unitary sphere (called the *Bloch Sphere* as shown in Fig. 1.1) as follows:

$$|\psi\rangle = e^{i\gamma}\left[\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right], \tag{1.3}$$

where $\theta \in [0, \pi], \varphi \in [0, 2\pi)$ and the factor $e^{i\gamma}$ can be completely ignored since it has no observable effect (we call it *global phase*).

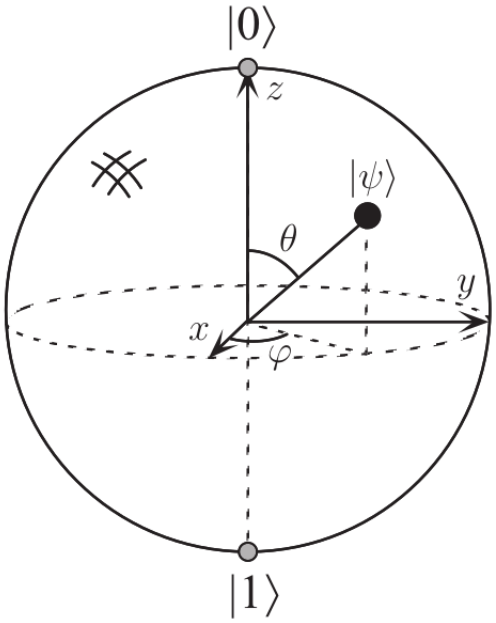

Figure 1.1: *Bloch* sphere. Figure source: [9]

The conjugate transpose of state $|\psi\rangle$ is

$$\langle\psi| = |\psi\rangle^\dagger = \begin{bmatrix}\alpha^* & \beta^*\end{bmatrix} \tag{1.4}$$

where $(\cdot)^*$ is the conjugation operation and $(\cdot)^\dagger$ is conjugation and transposing.

Using this notation, the inner product between two vectors $|\psi\rangle$ and $|\varphi\rangle$ can be written as

$$(|\psi\rangle, |\varphi\rangle) = \langle\psi|\,|\varphi\rangle = \langle\psi|\varphi\rangle. \tag{1.5}$$

The vector representation of a qubit presented in 1.1 is useful only in closed systems. An alternative (and more general) representation is using the *density matrix*.

Definition 1.1.1. Given a quantum state $|\psi\rangle$, its corresponding density matrix operator is defined by

$$\rho_\psi = |\psi\rangle\langle\psi|. \tag{1.6}$$

In this case, the state ρ_ψ is a pure state operator. If a density matrix operator ρ_ψ is a pure state operator, then

- it is hermitian ($\rho = \rho^\dagger$),
- it has one eigenvalue equal to 1 (the other are 0) and
- $\rho_\psi^2 = \rho_\psi$.

Definition 1.1.2. Given a collection of pure quantum states $|\psi_i\rangle$ and its associated probabilities p_i where $\sum_i p_i = 1$ then a mixed state ρ can be defined as

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{1.7}$$

Therefore, a mixed state is a convex lineal combination of pure states.

An operator ρ is a density matrix operator associated to an ensembles of quantum states if and only if

- ρ is hermitian,
- $tr(\rho) = 1$ and
- $\rho \geq 0$,

where $tr(\cdot)$ is the usual trace operator.

Hence, the postulate can be re-written in terms of density operators as follows

"A physical system is associated to a Hilbert space $\mathcal{H}$. The system can be completely described by density operators (semi-definite positive operators with unitary trace) acting on the space $\mathcal{H}$."

A density operator can be decomposed as

$$\rho = \frac{1}{2}\left(S_0\sigma_0 + S_1\sigma_1 + S_2\sigma_2 + S_3\sigma_3\right), \tag{1.8}$$

where σ_i are the Pauli matrices defined by

$$\sigma_0 = I,\ \sigma_1 = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ \sigma_2 = Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\ \sigma_3 = Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{1.9}$$

with I being the identity matrix and S_i are called the Stokes parameters which can be computed as

$$S_j = tr(\rho\sigma_j),\ \text{with } j \in \{0, 1, 2, 3\}. \tag{1.10}$$

The parameters S_1, S_2 and S_3 (often called S_X, S_Y and S_Z) can be used as coordinates of a vector in $\mathbb{R}^3$ in the Bloch sphere. Since $\sqrt{S_X^2 + S_Y^2 + S_Z^2} \leq 1$, and the equality only holds for pure states, one can represent mixed states as vectors in the inside of the Bloch sphere and pure states as vectors in its surface.

1.2 Postulate 2: Evolution of quantum systems

Postulate 2 [9]: *"The evolution of a closed system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state ψ' of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2."* That is that the state $|\psi'\rangle$ is described by

$$|\psi'\rangle = U\,|\psi\rangle\,, \tag{1.11}$$

where U is a unitary operator ($UU^\dagger = U^\dagger U = I$) that depends only on t_1 and t_2. For density matrices the same is computed by

$$\rho' = U\rho U^\dagger. \tag{1.12}$$

In fact, the evolution of a closed quantum system can be described by the Schrödinger equation for vector representation

$$i\hbar\frac{\mathrm{d}\,|\psi\rangle}{\mathrm{d}t} = \mathbb{H}\,|\psi\rangle\,, \tag{1.13}$$

or the Von Neumann equation for density representation

$$i\hbar\frac{\partial\rho}{\partial t} = \mathbb{H}\rho - \rho\mathbb{H}, \tag{1.14}$$

where $\mathbb{H}$ is the Hamiltonian of the system. The solution to these equation is

$$U = e^{-\frac{i}{\hbar}(t_2-t_1)\mathbb{H}}. \tag{1.15}$$

Since the Hamiltonian is hermitian, the matrix U is unitary.

Some examples of 1-qubit unitary matrices are the above-defined *Pauli matrices* in eq. (1.9) and the Hadamard gate defined as

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{1.16}$$

which takes the $|0\rangle$ and $|1\rangle$ to the $|+\rangle$ and $|-\rangle$ states, respectively, defined as the superpositions

$$\begin{aligned} |+\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{and} \\ |-\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \end{aligned} \tag{1.17}$$

Another example, for a 2-qubit system is the *CNOT* gate defined by

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{1.18}$$

which performs a *Controlled-NOT* operation, i.e. applies X in the second qubit if the first one is $|1\rangle$. This gate is usually represented in quantum circuits as in Fig. 1.2.

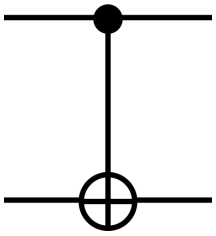

Figure 1.2: *CNOT* gate.

Another useful operation is the *Toffoli* (or *CCNOT*) gate [12] which applies X in the last qubit if the first two qubits are $|11\rangle$, defined as

$$Toffoli = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.19}$$

This gate is usually represented in quantum circuits as in Fig. 1.3.

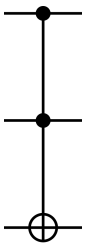

Figure 1.3: *Toffoli* gate.

1.3 Postulate 3: Quantum measurements

Postulate 3 [9]: *"Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that result m occurs is given by*

$$p(m) = \left\langle \psi \left| M_m^\dagger M_m \right| \psi \right\rangle, \tag{1.20}$$

and the state of the system after the measurement is

$$|\psi'\rangle = \frac{M_m |\psi\rangle}{\sqrt{\left\langle \psi \left| M_m^\dagger M_m \right| \psi \right\rangle}}." \tag{1.21}$$

The above mentioned definition implies that the probability outcomes add up to one, which is equivalent to the completeness equation

$$\sum_m M_m^\dagger M_m = I. \tag{1.22}$$

Then it is easy to see that the sum of probabilities add up to 1

$$\sum_m p(m) = \sum_m \left\langle \psi \left| M_m^\dagger M_m \right| \psi \right\rangle = \langle \psi | I | \psi \rangle = 1. \tag{1.23}$$

Equivalently, for density matrix representation one has that

$$p(m) = tr\left(M_m^\dagger M_m \rho\right) \tag{1.24}$$

and the state after measurement becomes

$$\rho' = \frac{M_m \rho M_m^\dagger}{tr\left(M_m^\dagger M_m \rho\right)}. \tag{1.25}$$

Definition 1.3.1. An observable M is a hermitian operator in the state spaces and has spectral decomposition

$$M = \sum_i \lambda_i \left|i\right\rangle \left\langle i\right| = \sum_i \lambda_i P_i, \tag{1.26}$$

where $\left|i\right\rangle$ are the eigenvectors of M corresponding to their eigenvalues λ_i. P_i are the orthogonal projectors in the subspaces of M.

In this case, given a state $\left|\psi\right\rangle$ the probability of having a result in the subspace i (when $M = P_i$) is

$$p(i) = \left\langle \psi \right| P_i \left| \psi \right\rangle \tag{1.27}$$

and the post-measurement state is

$$\left|\psi'\right\rangle = \frac{P_i \left|\psi\right\rangle}{\sqrt{p(i)}}. \tag{1.28}$$

Analogously, for density matrices one has

$$p(i) = tr\left(P_i \rho\right) \tag{1.29}$$

and

$$\rho' = \frac{P_i \rho P_i}{p(i)}. \tag{1.30}$$

It is worth noting that for 1-qubit systems, if one wants to measure the state $\left|\psi\right\rangle$ as in eq. 1.1 in the projectors of the computational basis ($P_0 = \left|0\right\rangle \left\langle 0\right|$ and $P_1 = \left|1\right\rangle \left\langle 1\right|$) then we have

$$\begin{aligned} p(0) &= |\alpha|^2 \quad \text{and} \\ p(1) &= |\beta|^2. \end{aligned} \tag{1.31}$$

1.4 Postulate 4: Composite systems

Postulate 4 [9]:

"The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$".

The same can be described in terms of density operators by $\rho_1 \otimes \rho_2 \otimes \cdots \otimes \rho_n$.

For finite matrices, the tensor product is called the *Kronecker* product. As is usual in quantum computing, we may omit the tensor product symbol and use

$$|\psi_1\rangle \otimes |\psi_2\rangle = |\psi_1\rangle\,|\psi_2\rangle = |\psi_1\psi_2\rangle\,. \tag{1.32}$$

1.5 Partial trace

Suppose we have a composite system comprised of subsystems (A and B) described by ρ_{AB}. In order to describe only the subsystem A one needs to do

$$\rho_A = tr_B(\rho_{AB}), \tag{1.33}$$

where tr_B is called the partial trace of the system over B.

Definition 1.5.1. The partial trace is an operator that holds the following properties:

- linearity

$$tr_B(\rho_{AB} + \lambda\hat{\rho}_{AB}) = tr_B(\rho_{AB}) + \lambda tr_B(\hat{\rho}_{AB}), \tag{1.34}$$

  where ρ_{AB} and $\hat{\rho}_{AB}$ are density operators in $\mathcal{H}_A \otimes \mathcal{H}_B$ and $\lambda \in \mathbb{C}$,

- if ρ_A and ρ_B are operators in $\mathcal{H}_A$ and $\mathcal{H}_B$, respectively, then

$$\begin{aligned} tr_B(\rho_A \otimes \rho_B) &= \rho_A tr(\rho_B), \\ tr_A(\rho_A \otimes \rho_B) &= \rho_B tr(\rho_A). \end{aligned} \tag{1.35}$$

In particular, if operators ρ_A and ρ_B are density operators ($tr(\rho_A) = tr(\rho_B) = 1$), then

$$\begin{aligned} tr_B(\rho_A \otimes \rho_B) &= \rho_A \text{ and} \\ tr_A(\rho_A \otimes \rho_B) &= \rho_B. \end{aligned} \tag{1.36}$$

1.5.1 Entanglement and separable states

An n-qubit system is not the same as having n independent 1-qubit systems and putting them together.

An example will clarify the situation. Suppose we have the 2-qubit system described by the following *Bell* state

$$|\psi_B\rangle = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right). \tag{1.37}$$

This state corresponds to a 2-qubit system. However, we cannot find 1-qubit states $|\psi_1\rangle$, $|\psi_2\rangle$ so that $|\psi_B\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. We say this state is *entangled*. Moreover, if one were to take the partial trace over any one of the two qubits one would get

$$\frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2} = \frac{I}{2}, \tag{1.38}$$

which is a mixed state. This implies that if you have this 2-qubit *Bell* state and "*discard*" 1 qubit (analyse only 1 qubit) you have lost the information on the state.

The above example will be made clear with the following definitions.

Definition 1.5.2. A pure state which cannot be decomposed into tensor products of 1-qubit states is called an *entangled state*.

Definition 1.5.3. An n-qubit state that can be decomposed as tensor products of 1-qubit states is called a *product state* (or *entirely separable state*).

Definition 1.5.4. When a system is represented by a convex sum of density matrices of pure states the state is called *entirely separable*. A density matrix is said to be *entirely*

separable if there exists a decomposition

$$\rho = \sum_i p_i \rho_i, \tag{1.39}$$

where $p_i \geq 0$ and $\sum_i p_i = 1$ such that all ρ_i is an entirely separable state.

Definition 1.5.5. A quantum state is said to be *uncorrelated* if its density operator can be decomposed into tensor product of 1-qubit density operators

$$\rho = \rho_1 \otimes \rho_2 \otimes \cdots \otimes \rho_n, \tag{1.40}$$

where ρ_i are 1-qubit density operators.

Note that an uncorrelated state is separable, but the implication goes only one way (not every separable state is uncorrelated).

1.6 Fidelity

A useful measure of distance between quantum states is the *fidelity* [13]. The fidelity between states ρ and σ is defined by

$$F(\rho, \sigma) = tr\sqrt{\rho^{1/2}\sigma\rho^{1/2}} \tag{1.41}$$

and it can be proved that given a pure state the fidelity is reduced to

$$F(|\phi\rangle, \rho) = \sqrt{\langle\phi| \rho |\phi\rangle}. \tag{1.42}$$

It is useful to note that the fidelity is symmetric, invariant under unitary transformations and gives rise to a metric on density operators by defining the angle between two states as

$$A(\rho, \sigma) = \arccos F(\rho, \sigma). \tag{1.43}$$

Hence, the fidelity is a measure of distance between states.

2. Quantum error modeling

The pursuit of reliable quantum computation requires a thorough understanding and management of quantum errors. These errors, inherent in quantum systems, are studied and characterized through precise error modeling, a process that serves as the foundation for devising robust error correction schemes. In this section we will see some error models that can be applied to study quantum gates, quantum channels, decoherence, noise, and other interactions with the environment.

The following extract is based on information found in [9].

2.1 Introduction to error models in quantum computing

We can model any operation $\mathcal{E}$ (including error operations) as an *open quantum system* (*principal system*) interacting with the *environment*. We can see the *principal system* and the *environment* as a closed quantum system in a higher dimension.

It can be shown that if the principal system is a d-dimensional Hilbert space, then the environment can be modeled as a pure state in a d^2-dimensional Hilbert space. Therefore, we can model the closed system as a product state between the principal system ρ and the environment $\rho_{\text{env}} = |e_0\rangle \langle e_0|$.

This means that the state obtained after performing $\mathcal{E}$ is given by

$$\mathcal{E}(\rho) = \text{tr}_{\text{env}} \left[U(\rho \otimes \rho_{\text{env}}) U^\dagger \right], \tag{2.1}$$

which can be seen as the circuit in Fig. 2.1.

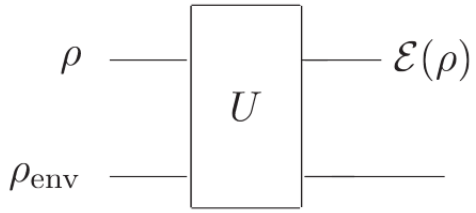

Figure 2.1: Interaction with environment. Figure source: [9].

2.2 Operator-sum representation (OSR)

Given the equation (2.1) and $|e_k\rangle$ an orthonormal basis for the environment, we can see that

$$\begin{aligned}
\mathcal{E}(\rho) &= \mathrm{tr}_{\text{env}}\left[U(\rho \otimes |e_0\rangle\langle e_0|)U^\dagger\right] \\
&= \sum_k \langle e_k| U (\rho \otimes |e_0\rangle\langle e_0|) U^\dagger |e_k\rangle \\
&= \sum_k \langle e_k| U |e_0\rangle \rho \langle e_0| U^\dagger |e_k\rangle \\
&= \sum_k E_k \rho E_k^\dagger
\end{aligned} \tag{2.2}$$

with $E_k = \langle e_k| U |e_0\rangle$ an operator on the state space of the principal system. Eq. (2.2) is known as the operator-sum representation of operation $\mathcal{E}$.

Since the output after performing operation $\mathcal{E}$ is a quantum density operator, it must have unitary trace[1]. This constraint is known as the *completeness relation*

$$\sum_k E_k^\dagger E_k = I, \tag{2.3}$$

where I is the identity operator. Operators that satisfy this constraint are also called

[1]It is possible, and sometimes useful, to define non-trace-preserving operations to account for measurements and encode their probabilities ($\mathrm{tr}(\mathcal{E}(\rho)) \leq 1$), but this will not be used in the entirety of this work.

Kraus operators [14].

It is useful to note some properties of the $\mathcal{E}$ operator acting on quantum systems that will be used throughout this work:

1. Trace preserving: $\mathrm{tr}\left(\mathcal{E}(\rho)\right) = \mathrm{tr}\left(\rho\right)$.
2. Linearity: $\mathcal{E}\left(\sum_i p_i \rho_i\right) = \sum_i p_i \mathcal{E}(\rho_i)$.
3. Completely positive, given $\rho \geq 0$ then $\mathcal{E}(\rho) \geq 0$.

It is worth noting that this representation is not unique. In fact, this is called the Unitary freedom in the operator-sum representation. Suppose we have $\{E_m\}$ and $\{F_n\}$ are operations representing operations $\mathcal{E}$ and $\mathcal{F}$, respectively. Add the null operator as needed so that $m = n$. Then $\mathcal{E} = \mathcal{F}$ if and only if there exists a unitary matrix $U = (u_{ij})$ such that $E_i = \sum_j u_{ij} F_j$.

2.3 Bit flip channel (BF)

The *Bit Flip* Channel $\mathcal{E}_{\mathrm{BF}}$ flips the state $|0\rangle$ to $|1\rangle$ (and vice versa) with probability p and leaves it unaffected with probability $1 - p$ and is defined by

$$\begin{aligned} E_0 &= \sqrt{1-p} I = \sqrt{1-p} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ E_1 &= \sqrt{p} X = \sqrt{p} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \end{aligned} \tag{2.4}$$

This is the analogous case for the classical probability of error.

2.4 Phase flip channel (PF)

The *Phase Flip* Channel $\mathcal{E}_{\mathrm{PF}}$ flips the phase of state $|1\rangle$ with probability p and leaves it unaffected with probability $1-p$ and is defined by

$$\begin{aligned} E_0 &= \sqrt{1-p}I = \sqrt{1-p}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ E_1 &= \sqrt{p}Z = \sqrt{p}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \end{aligned} \tag{2.5}$$

2.5 Bit-phase flip channel (BPF)

The *Bit-Phase Flip* Channel $\mathcal{E}_{\mathrm{BPF}}$ is a combination of a phase flip and a bit flip (note that $Y = iXZ$) with probability of error p and leaves the state unchanged with probability $1-p$. It is defined by

$$\begin{aligned} E_0 &= \sqrt{1-p}I = \sqrt{1-p}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ E_1 &= \sqrt{p}Y = \sqrt{p}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}. \end{aligned} \tag{2.6}$$

2.6 Depolarizing channel (DC)

The *Depolarizing Channel* $\mathcal{E}_{\mathrm{DC}}$ models the noise of having the system considered completely depolarized (taken to a maximally mixed state) with probability p and leaving it unchanged with probability $1-p$.

2.6.1 1-qubit depolarizing channel

The *1*-qubit depolarizing channel is defined as

$$\mathcal{E}(\rho)_{\mathrm{DC_1}} = p\frac{I}{2} + (1-p)\rho, \tag{2.7}$$

where p is the probability of depolarization and $\frac{I}{2}$ is the maximally mixed state for *1*-qubit systems.

This means that the qubit is depolarized with probability p and left unaffected with probability $1-p$.

Since for any ρ it follows that

$$\frac{I}{2} = \frac{I\rho I + X\rho X + Y\rho Y + Z\rho Z}{4}, \tag{2.8}$$

then we have that Eq. (2.7) becomes

$$\mathcal{E}(\rho)_{\mathrm{DC_1}} = \left(1 - \frac{3p}{4}\right) I\rho I + \frac{p}{4}(X\rho X + Y\rho Y + Z\rho Z). \tag{2.9}$$

Hence, the OSR of the *1*-qubit depolarizing channel is described by the set

$$\left\{\sqrt{1-3p/4}I, \sqrt{p/4}X, \sqrt{p/4}Y, \sqrt{p/4}Z\right\}. \tag{2.10}$$

This can be interpreted as the system left unaffected with probability $1-3p/4$ and the operators X, Y and Z applied each with probability $p/4$.

2.6.2 n-qubit depolarizing channel

The depolarizing channel can be generalized to an n-qubit depolarizing channel by defining

$$\mathcal{E}(\rho)_{\mathrm{DC_n}} = p\frac{I}{2^n} + (1-p)\rho. \tag{2.11}$$

This leaves the state untouched with probability $1-p$ and takes it to the maximally mixed state $\frac{I}{2^n}$ with probability p.

2.7 Amplitude damping channel (AD)

The *Amplitude Damping Channel* $\mathcal{E}_{\mathrm{AD}}$ is used to model energy dissipation in a quantum system. Assume that we have a system with a low-energy state (i.e. $|0\rangle$) and a high-energy state (i.e. $|1\rangle$). The AD models a transition from the high-energy state to the low-energy state.

The AD channel ($\mathcal{E}_{\mathrm{AD}}$) is defined by its operators

$$\begin{aligned} E_0 &= \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix} \\ E_1 &= \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}. \end{aligned} \tag{2.12}$$

The AD can be thought of *damping* the amplitude of the state $|1\rangle$ into the state $|0\rangle$ with probability γ.

2.8 Phase damping channel (PD)

The *Phase Damping Channel* $\mathcal{E}_{\mathrm{PD}}$ is used to model information loss without loss of energy. It is defined by

$$\begin{aligned} E_0 &= \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix} \\ E_1 &= \begin{bmatrix} 0 & 0 \\ 0 & \sqrt{\gamma} \end{bmatrix}. \end{aligned} \tag{2.13}$$

At a first glance, this seems to be a completely different operation than the described before since E_1 destroys the state $|0\rangle$ and reduces the amplitude of the state $|1\rangle$. However,

another representation of the same channel is made by setting $\alpha = \frac{1-\sqrt{1-\lambda}}{2}$ and becomes

$$\begin{aligned} \hat{E}_0 &= \sqrt{1-\alpha} I = \sqrt{1-\alpha} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \hat{E}_1 &= \sqrt{\alpha} Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \end{aligned} \tag{2.14}$$

which is a phase flip channel with probability of error α. Hence, the phase damping channel is *the same* as the phase flip channel.

2.9 Generalized amplitude damping channel (GAD)

The *Generalized Amplitude Damping* Channel $\mathcal{E}_{\text{GAD}}$ tries to model the dissipation of a higher-energy state to a lower-energy state, but with finite temperature (it can be thought of taking the state to a stationary state "close" to the state $|0\rangle$ for p close to 1).

It is defined by the operators

$$\begin{aligned} E_0 &= \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix} \\ E_1 &= \sqrt{p} \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix} \\ E_2 &= \sqrt{1-p} \begin{bmatrix} \sqrt{1-\gamma} & 0 \\ 0 & 1 \end{bmatrix} \\ E_3 &= \sqrt{1-p} \begin{bmatrix} 0 & 0 \\ \sqrt{\gamma} & 0 \end{bmatrix} \end{aligned} \tag{2.15}$$

and the stationary state is

$$\rho_\infty = \begin{bmatrix} p & 0 \\ 0 & 1-p \end{bmatrix}, \tag{2.16}$$

that satisfies

$$\mathcal{E}_{\text{GAD}}(\rho_\infty) = \rho_\infty. \tag{2.17}$$

2.10 Other error models

The above-described error models are the most known and typically studied. There are, however, many other errors that can be considered.

An example is the Pauli channels. They refer to the error modeled as applying each of the Pauli matrices (I, X, Y, Z) with different values for each of them. Note that the depolarizing channel is a special case of a Pauli channel.

Another kind of error model is the rotation errors. If instead of performing a rotation of an angle α and the rotation performed is $\alpha + \epsilon$, this is a new class of errors.

A different class of errors occurs when errors happen for control qubits. Imagine if the $CNOT$ gate applied the bit flip even when the control qubit is in state $|0\rangle$ with some probability p (or not applied if the control qubit is in state $|1\rangle$). This is yet another type of error model.

Finally, one last example of different types of error is the measurement errors. These happen when the measured state of a qubit does not reflect its actual state. This can happen due to the physical implementation of the measurement device, interference, or noise.

3. Linear codes

The following introduction is a summary from [11, 9].

In the pursuit to achieve reliable quantum computing (and quantum error correction), we must first understand classical linear codes. Most of the concepts that are known for classical linear codes will be analogous in their quantum counterpart. Hence, we first introduce concepts from classical linear codes and then from their quantum counterpart.

3.1 Classical codes

Classical linear codes have been fundamental in the storage and transmission of information in classical computing systems. They are structured to detect and correct errors that may arise during transmission or storage of information.

Suppose that we want to use the linear code to protect the information against error in the codewords. What is the maximum error-correcting capability of the code? This question will be answered below.

3.1.1 Basic definitions

Definition 3.1.1. An $[n, k, d]_q$ *linear code* of length n and dimension k is a subspace $\mathcal{C}$ with dimension k of the vector space $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is a finite field with q elements. We call d the distance of the code and it is the minimum distance between any two codewords. The number of codewords in a code is called its *size* and is equal to q^k. When $q = 2$ we say that the code is a binary code and the parameter q is omitted.

Definition 3.1.2. The *Hamming distance* [15] between two codewords is defined as the number of elements in which they are different.

Definition 3.1.3. The *Hamming weight* [15] of a codeword is the number of elements that are non-zero.

Note that since linear codes live in a vector subspace, the linear combination of any codeword is another codeword. In particular, the addition and subtraction of any two codewords is another codeword, and the null vector is always a codeword.

Another important fact to note is that the distance of the code d is the minimum weight of the codewords. This can be proved easily as follows:

$$\begin{aligned} \min_{c_1,c_2\in\mathcal{C}} D(c_1,c_2) &= \min_{c_1,c_2\in\mathcal{C}} D(c_1-c_1,c_2-c_1) \\ &= \min_{c_1,c_2\in\mathcal{C}} D(0,c_2-c_1) \\ &= \min_{c\in\mathcal{C}} D(0,c) \\ &= \min_{c\in\mathcal{C}} w(c), \end{aligned} \tag{3.1}$$

where $D(\cdot,\cdot)$ is the Hamming distance and $w(\cdot)$ is the Hamming weight.

3.1.2 Parity and Generator matrices

A classical linear code can be described by an $n \times k$ *generator matrix* G. The matrix G is used to encode the original words into their codewords. Suppose we have a message x we want to encode, then we encode it by doing Gx, where x is a column vector. The arithmetic operations correspond to the operations in the corresponding finite field (modulo q).

An equivalent definition for the code can be done by the *parity matrix* (also called *parity check matrix*) H, defined by $Hx = 0$, where H is an $(n-k) \times n$ matrix. The description of the parity matrix is very useful since if we have an error, we receive $z = x + e$ and we have $Hz = H(x+e) = Hx + He = He$, which is called the *syndrome* of the received message.

For every code $\mathcal{C}$ we can define its dual code $\mathcal{C}^\perp$ as the code with generator matrix H^T and parity check matrix G^T. It is indeed the orthogonal space of the subspace spanned by $\mathcal{C}$.

3.1.3 The Hamming and singleton bounds

One important bound for the error-correcting capability of the code is the Hamming bound. For a capability of correcting up to t errors the Hamming bound [15] states that

$$q^k \sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leq q^n, \tag{3.2}$$

where the correcting capability is related to the distance of the code. Namely,

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor. \tag{3.3}$$

The codes that satisfy this bound with equality are called perfect codes.

Another important bound is the called Singleton bound [16] and states that the parameters of the code must satisfy

$$d - 1 \leq n - k, \tag{3.4}$$

and the codes that satisfy this bound with equality are known as maximum distance separable codes. One example of this are the Reed-Solomon codes [17].

3.1.4 The Gilbert-Varshamov bound

A well known, yet very important, result for classical linear codes is the *Gilbert-Varshamov bound* [18, 19]. It states that for large n, there exists an $[n, k]$ code protecting against errors on t bits for some k such that

$$\frac{k}{n} \geq 1 - H\left(\frac{2t}{n}\right), \tag{3.5}$$

where $H(x) = -x\log(x) - (1-x)\log(1-x)$ is the binary Shannon entropy [20].

This bound *guarantees* the existence of good codes, provided that the encoding rate k/n is *reasonable*.

3.2 Quantum linear codes

Quantum error correction is more complicated than classical error correction. First, due to the continuum nature of the qubit, the errors are also continuous. Second, since we cannot clone a qubit (because of the no-cloning theorem [21, 22]). And, lastly, quantum measurements destroy quantum information.

We will formally describe some aspects that are key to understanding quantum linear codes and error correction. To begin, we will formally introduce some concepts and then provide an overview of fault-tolerant computing. We will denote by $\mathcal{C} = [n, k]$ a linear quantum error correcting code (QECC), encoding k qubits into n qubits.

3.2.1 Stabilizers

QECC codes can be described either by their encoding, correction and decoding, or by its stabilizers.

Definition 3.2.1. A *Pauli* operator on n qubits has the form $cO_1 \otimes O_2 \otimes \cdots \otimes O_n$, where each $O_j \in \{I, X, Y, Z\}$ and $c \in \{1, -1, i, -i\}$. The set of Pauli operators on n qubits forms the *multiplicative Pauli group* G_n.

Definition 3.2.2. The *stabilizer group* S is the largest Abelian subgroup of G_n acting on a set W that fixes all elements from the quantum code $\mathcal{C}$. Namely,

$$S = \{g \in G_n : gw = w\}, \tag{3.6}$$

where $w \in W$.

The stabilizer $S \subseteq G_n$ is constructed by a set of $n - k$ operators (the generators of S). Any element of the stabilizer group can then be written in terms of a unique product of powers of the generators

$$s = g_1^{c_1} \cdots g_{n-k}^{c_{n-k}}, \tag{3.7}$$

where $c_j \in \{0, 1\}$. Hence, the elements of S are represented by the generators and the

binary string $c = c_1 c_2 \cdots c_{n-k}$.

This representation perfectly describes the QECC and was introduced by Daniel Gottesman [23].

3.2.2 Degenerate codes

Definition 3.2.3. A QECC is said to be a *degenerate code* if the effect of errors in different qubits can lead to the same corrupted codeword.

It is a purely quantum phenomenon that there exist degenerate quantum codes, since in classical error-correcting codes errors on different bits will always result in different corrupted codewords.

3.2.3 Quantum Hamming and singleton bounds

The quantum Hamming bound [24] is analogous to the classical Hamming bound (Eq. (3.2)). Suppose a non-degenerate code $\mathcal{C} = [n, k]$ is meant to correct errors on at most t qubits, then we have the quantum Hamming bound

$$2^k \sum_{j=0}^{t} \binom{n}{j} 3^j \leq 2^n, \tag{3.8}$$

where the factor 3^j corresponds to the errors on each of the X, Y, Z direction. Note that this bound only applies to non-degenerate codes.

Similarly, the quantum Singleton bound [24] can be constructed where the bound becomes

$$n - k \geq 4t, \tag{3.9}$$

where we are encoding k qubits into n with arbitrary erros in at most t qubits.

3.2.4 The quantum Gilbert-Varshamov bound

Analogously to the classical Gilbert-Varshamov bound (Eq. (3.5)), there is the quantum Gilbert-Varshamov bound [24] that *guarantees* the existence of *good codes* provided the encoding rate is *reasonable*. The bound changes to

$$\frac{k}{n} \geq 1 - 2H\left(\frac{2t}{n}\right), \tag{3.10}$$

where $H(\cdot)$ is the Shannon entropy as defined before.

3.2.5 CSS codes

A special class of quantum error correcting codes is the so-called *Calderbank-Shor-Steane* (*CSS*) codes, named after their inventors [25, 26].

Definition 3.2.4. Let $\mathcal{C}_1, \mathcal{C}_2$ be two classical linear codes with parameters $[n, k_1, d_1]$, $[n, k_2, d_2]$, $\mathcal{C}_2 \subset \mathcal{C}_1$, and $\mathcal{C}_1, \mathcal{C}_2^{\perp}$ correct t errors. The $CSS(\mathcal{C}_1, \mathcal{C}_2)$ quantum code is an $[n, k_1 - k_2]$ quantum code, capable of correcting errors on t qubits, defined as a vector space spanned by

$$|x + \mathcal{C}_2\rangle = \frac{1}{\sqrt{|\ \mathcal{C}_2\ |}} \sum_{y \in \mathcal{C}_2} |x + y\rangle\,, \tag{3.11}$$

where $x \in \mathcal{C}_1$ is any codeword in the code $\mathcal{C}_1$ and $+$ is bit-wise addition modulo 2.

3.2.6 Fault-tolerant quantum computing

One of the most important results and applications of quantum error correction is the so-called *fault-tolerant (FT) quantum computing*. This states that, given a fixed error probability for quantum gates is below a certain threshold, arbitrarily good quantum computation can be achieved. Hence the name fault-tolerant quantum computing, since it is tolerant to faulty quantum gates and channels.

Gottesman made one of the biggest contributions to the fault-tolerant construction formalism, showing how to perform FT quantum computation with *any* stabilizer code [23].

It can be shown that the FT quantum computing can do successive operations directly on encoded quantum states (no decoding is required). The idea is to change every part of the quantum computer with its FT version.

One of the most important results of FT computing is, as stated above, *the threshold theorem for quantum computation.* A quantum circuit containing $p(n)$ gates may be simulated with probability of error at most ϵ using

$$O(\text{poly}(\log p(n)/\epsilon)p(n)) \tag{3.12}$$

hardware gates with probability of failing $p_{\text{fail}} < p_{\text{threshold}}$ with $p_{\text{threshold}}$ being a constant threshold.

4. The Shor code

The mutual dependency between error modeling and error correction forms a critical axis in the journey to achieve fault-tolerant quantum computing[27, 28, 29, 30, 31, 32]. Our study of quantum errors and their behaviors is useful for developing effective countermeasures, reinforcing the reliability and practicality of quantum computing systems.

One of the most famous error correction codes is Shor's 9-qubit [8] error correction code. We will study this in depth in this section, explaining how the code works and how it is built. In particular, we will show how the code is constructed based on the linearity of the code and the concatenation of 3-qubit codes protecting against bit-flip and phase-flip errors.

Recall that any linear code correction circuit can be interpreted as in Fig. 4.1, where w is the amount of *ancilla* qubits introduced (for the Shor code $w = 8$).

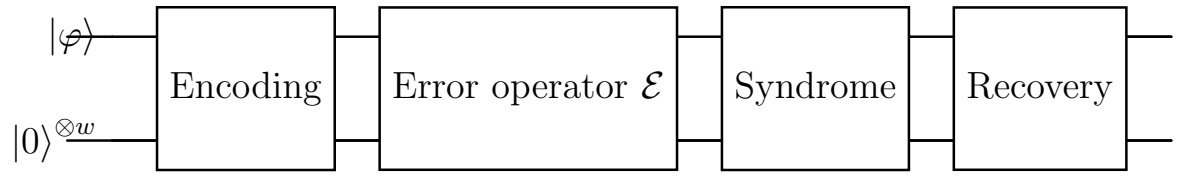

Figure 4.1: General quantum error correcting code.

4.1 3-qubit code against bit flip errors

Suppose we have a channel that can be modelled by the bit flip channel as in Eq. (2.4). That is, it takes the state $|\varphi\rangle$ into $X|\varphi\rangle$ with probability p. We will protect $|\varphi\rangle$ against at most one BF error by using a repetition code.

Let us encode the qubit in the following way

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle = |000\rangle \\ |1\rangle &\rightarrow |1_L\rangle = |111\rangle \\ |\varphi\rangle = a\,|0\rangle + b\,|1\rangle &\rightarrow |\varphi_L\rangle = a\,|000\rangle + b\,|111\rangle\,, \end{aligned} \tag{4.1}$$

which can be achieved with the circuit in Fig. 4.2.

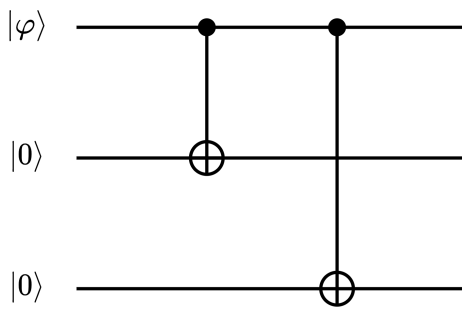

Figure 4.2: 3-qubit repetition code encoding

Now, we will assume at most one bit flip error occurred and will see how the syndrome can be calculated. The possible outcomes after the error are

$$\begin{aligned} &a\,|000\rangle + b\,|111\rangle && \text{(no error)} \\ &a\,|100\rangle + b\,|011\rangle && \text{(error in qubit \#1)} \\ &a\,|010\rangle + b\,|101\rangle && \text{(error in qubit \#2)} \\ &a\,|001\rangle + b\,|110\rangle && \text{(error in qubit \#3),} \end{aligned} \tag{4.2}$$

and the syndrome can be calculated with the circuit in Fig. 4.3.

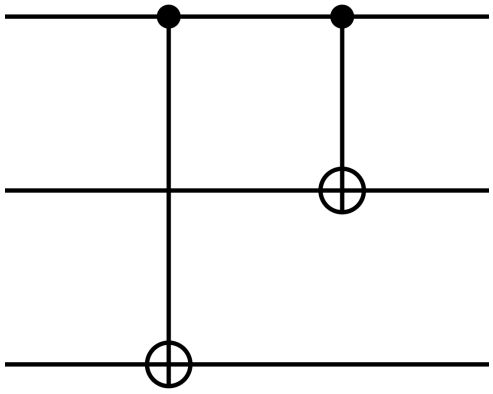

Figure 4.3: 3-qubit repetition code syndrome

After the syndrome the resulting state is one of the following

$$\begin{aligned}
&(a\,|0\rangle + b\,|1\rangle) \otimes |00\rangle && \text{(no error)} \\
&(a\,|1\rangle + b\,|0\rangle) \otimes |11\rangle && \text{(error in qubit \#1)} \\
&(a\,|0\rangle + b\,|1\rangle) \otimes |10\rangle && \text{(error in qubit \#2)} \\
&(a\,|0\rangle + b\,|1\rangle) \otimes |01\rangle && \text{(error in qubit \#3).}
\end{aligned} \tag{4.3}$$

Hence, we can apply a NOT gate only for syndrome corresponding to $|11\rangle$, which can achieved using a Toffoli gate after syndrome calculation as in Fig. 4.4, to obtain the original state $|\varphi\rangle$.

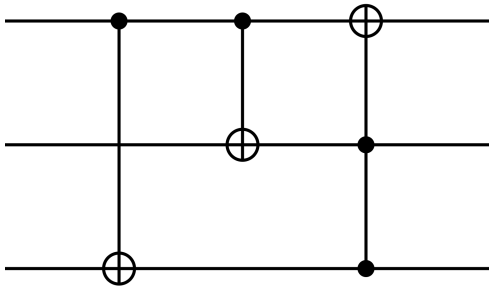

Figure 4.4: 3-qubit repetition code syndrome and correction

The complete circuit can be seen in Fig. 4.5

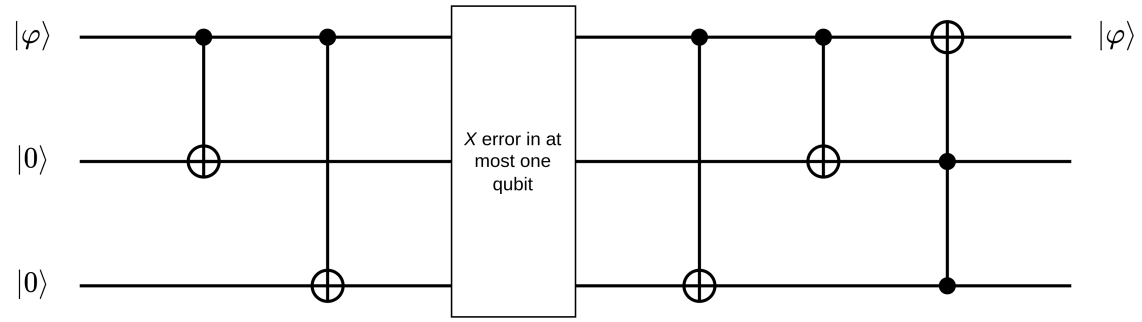

Figure 4.5: 3-qubit repetition code

This can also be represented by its set of stabilizers [9]: $\{Z_1Z_2, Z_2Z_3\}$, where Z_i is the Z gate acting on the i-th qubit.

4.2 3-qubit code against phase flip errors

Suppose we have a channel that can be modelled by the phase flip channel as in Eq. (2.5). That is, it takes the state $|\varphi\rangle$ into $Z|\varphi\rangle$ with probability p.

We can turn the bit flip correcting code from above to correct for phase flips. If instead of working with the basis states $\{|0\rangle, |1\rangle\}$ we work with $\{|+\rangle, |-\rangle\}$, then the phase flip turns $|+\rangle$ into $|-\rangle$ (and vice-versa). That is, the phase flip is a bit flip on the basis states $\{|+\rangle, |-\rangle\}$. This can be seen by noting that

$$X = HZH, \tag{4.4}$$

where X is the *bit flip* gate, Z is the *phase flip* gate and H is the *Hadamard gate*.

Hence, we can encode the qubit as follows to protect the state against phase flip errors:

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle = |+++\rangle \\ |1\rangle &\rightarrow |1_L\rangle = |---\rangle, \end{aligned} \tag{4.5}$$

which is accomplished by the circuit in Fig. 4.6.

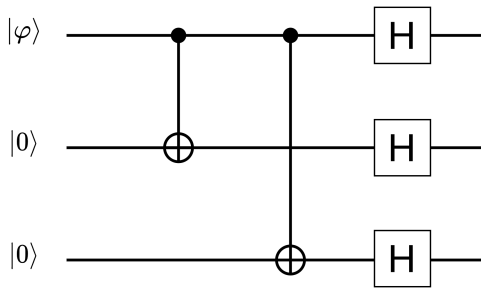

Figure 4.6: 3-qubit repetition code encoding for PF errors

The complete circuit can be seen in Fig. 4.7.

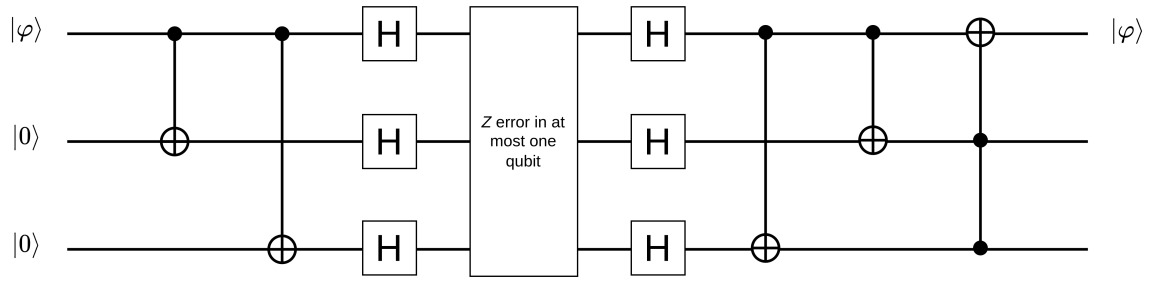

Figure 4.7: 3-qubit repetition code for PF errors

This can also be represented by its set of stabilizers [9]: $\{X_1X_2, X_2X_3\}$, where X_i is the X gate acting on the i-th qubit.

4.3 1-qubit error decomposition

We will now show how correcting against bit flips and phase flips, we can correct against *any* kind of error in 1-qubit. Without loss of generality, we will assume that the error occurs in the first qubit.

Assume that we have a system with n qubits in which the error will occur. It is helpful

to define the bit flip and phase flip gates affecting the first qubit only as follows

$$\begin{aligned} X_1 &= X \otimes I_{n-1} \\ Z_1 &= Z \otimes I_{n-1}, \end{aligned} \tag{4.6}$$

where I_{n-1} is the identity matrix for $n-1$-qubit systems.

If we call $|\varphi\rangle$ to the encoded qubit, the noise will act on the qubit with some OSR given by the set $\{E_i\}$ and the state becomes

$$\mathcal{E}(|\varphi\rangle\langle\varphi|) = \sum_i E_i |\varphi\rangle\langle\varphi| E_i^\dagger. \tag{4.7}$$

But E_i can be decomposed in terms of bit flips, phase flips and bitphase flips for the first qubit

$$E_i = e_{i0} I_n + e_{i1} X_1 + e_{12} Z_1 + e_{i3} X_1 Z_1. \tag{4.8}$$

Hence, if we can protect the first qubit against bit flip and phase flip errors, we can correct against *any* kind of error in the first qubit.

4.4 The Shor code

The Shor code leverages the above result in which protecting one qubit against bit flips and phase flips, it protects it against any type of error (see 4.3). This, together with the concatenation of two 3-qubit repetition codes to correct for bit flips and phase flips, gives rise to *The Shor Code*.

As stated above, the Shor code encodes the qubit in two concatenated steps. First, it takes the encoding against phase flips. Then, each of these qubits is encoded against bit flips. The resulting encoding is therefore

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle = \frac{1}{\sqrt{8}}(|000\rangle + |111\rangle)^{\otimes 3} \\ |1\rangle &\rightarrow |1_L\rangle = \frac{1}{\sqrt{8}}(|000\rangle - |111\rangle)^{\otimes 3}. \end{aligned} \tag{4.9}$$

The analogous process is performed for the detection of syndromes and the correction of errors, which results in the circuit shown in Fig. 4.8.

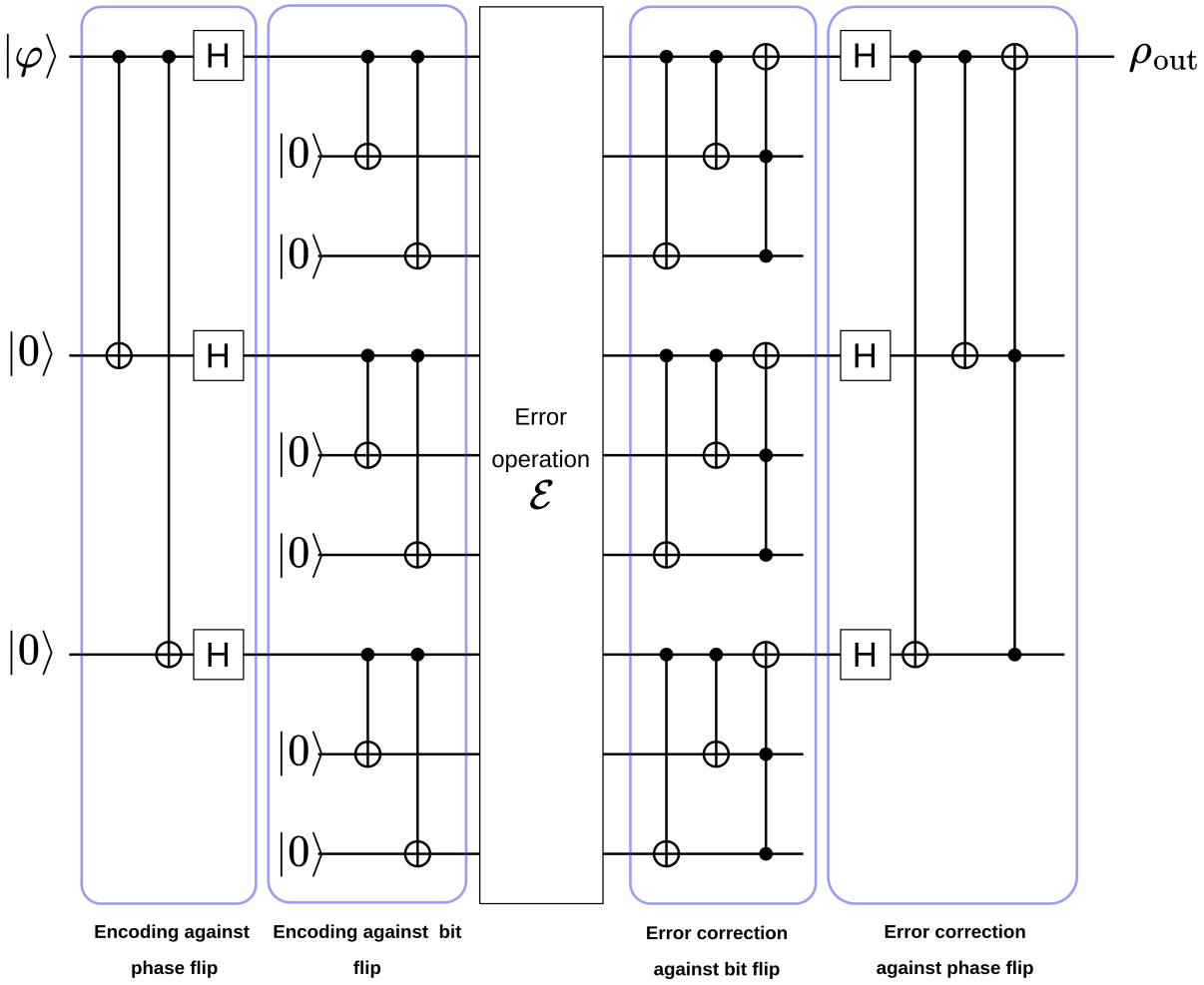

Figure 4.8: The Shor Code

This can also be represented by its set of stabilizers [9]:

$$\{Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9, X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9\}, \tag{4.10}$$

where Z_i and X_i are the Z and X gate acting on the i-th qubit.

5. Shor's code with imperfect gates and channels

In this section, we will show different error modeling for both the quantum gates and the channels, providing analytical formulas for the resulting state in each of them. We will now present a study with novel results on an implementation of the Shor code with errors. The method used to obtain these can be found in Annex 2 and the reader is encouraged to look at it.

We would like to emphasize that even though we present the results with some hypothesis, the main contribution is the way to do the needed calculations. If we were to be given details about a specific implementation of the technology, we could change the hypothesis and error models and study the effects of these with the same methodology.

We will model both quantum gates errors and transmission channel errors with the following hypotheses:

1. The errors will be 1-qubit errors (they will not correlate different qubits).
2. The errors will be of the same type for each operation.
3. The errors for the gates will have an error width (γ) different from that of the channel (δ), but they will be of the same type.
4. The errors for the gates will occur after every operation, only on the qubits affected by the gate.

Why do we introduce these hypotheses? The first hypothesis is to simplify the model (we assume that if there is some error for a subsystem of a higher number of qubits, it will be negligible). The second hypothesis seems reasonable if the channels in which the qubits are transmitted are constructed in a similar way (both inside the quantum computer

and the transmission channel). The third hypothesis differentiates the error width of the quantum computer from the error width of the transmission channel. And finally, the fourth hypothesis is to assume that the computing (real gates) error introduction is only caused by the internal transmission of the qubit (i.e. the gates are "perfect" but the output causes some noise internally).

An example of a quantum circuit modeled by our hypothesis is given in Fig. 5.1, our channel error model can be seen in Fig. 5.2, and the complete implementation of the Shor code can be seen in Fig. 5.3. Every error model is represented by $\mathcal{E}^{p}_{\text{type}}$, where p is the probability of error (or error width) and type is the type of error. For example, $\mathcal{E}^{\gamma}_{\text{AD}}$ refers to an amplitude damping error with parameter γ. We will always denote by γ the width for gate error and by δ the width for channel error.

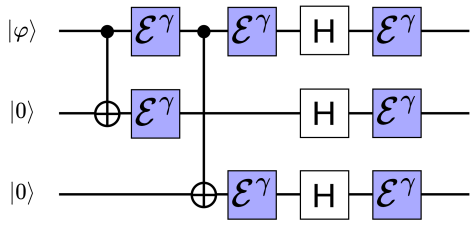

Figure 5.1: Gate error example

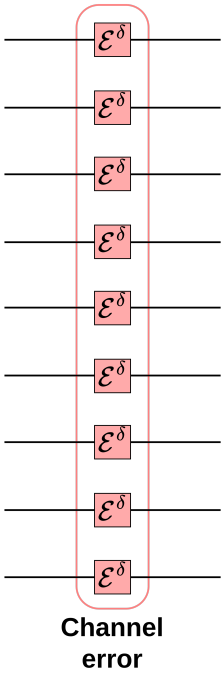

Figure 5.2: Our transmission channel error model

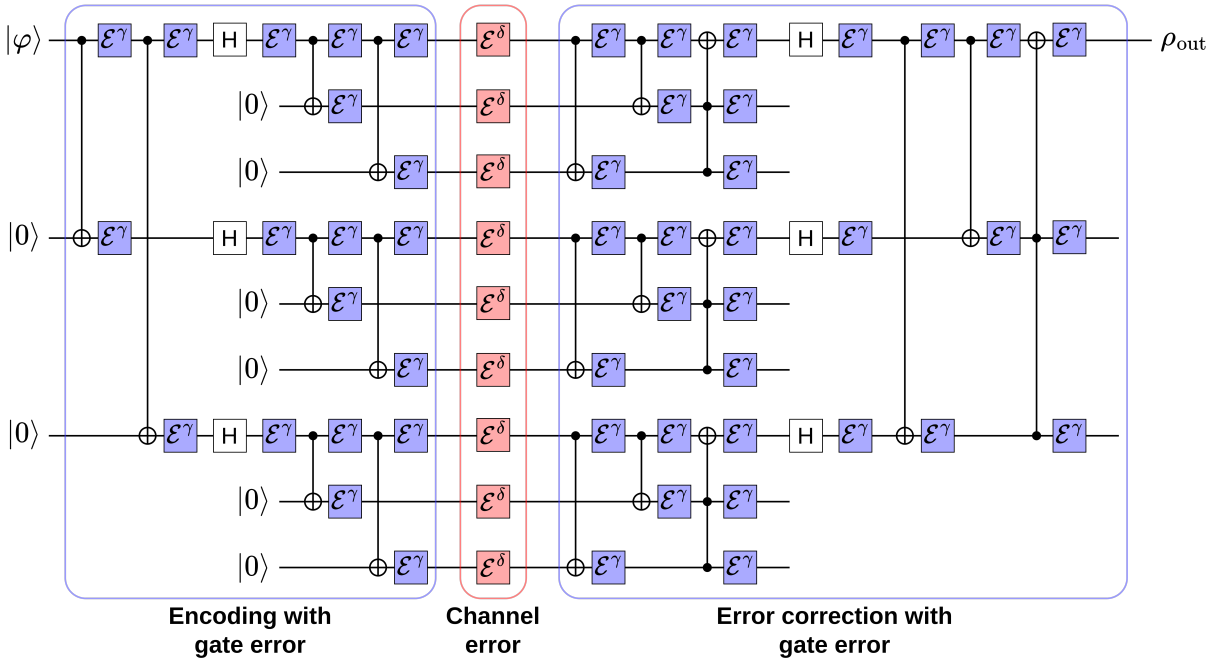

Figure 5.3: Shor code implementation with our error model

The results obtained here are not using any approximation besides the hypothesis mentioned above. They are the result of months of algebraic calculations, simplifications, expansions, and heuristic algebraic simplification approach using Wolfram Mathematica software [33]. The key decomposition for the method can be found in Annex 1. The results were verified numerically with the *QuantumLab Simulator* (see Appendix A of [10]), a numerical tool built by the quantum computing group at *Universidad ORT Uruguay* on top of SciLab [34].

We will provide analytical expressions for the output of the modeled implementation of the Shor code for any value of $\gamma, \delta \in [0, 1]$. We believe this to be of utmost importance for the Noise Intermediate Scale Quantum (NISQ) era [35]. We will also provide approximations for small values of γ and δ. We have modeled the following error types: depolarizing channel (DC), bit flip (BF), phase flip (PF), bit-phase flip (BPF), and amplitude damping (AD).

We will divide this section into two sections. In the first section, we will show the first-order approximation for small values of γ and δ. And in the second section, we will assume perfect gates ($\gamma = 0$), analyze and give an upper bound for when it is better to use the encoding or not.

The input state (the state to be encoded) will be represented as

$$\rho_{in} = |\varphi\rangle \langle\varphi| = \begin{bmatrix} \alpha^2 & \alpha\sqrt{1-\alpha^2}e^{-i\phi} \\ \alpha\sqrt{1-\alpha^2}e^{i\phi} & 1-\alpha^2 \end{bmatrix}, \tag{5.1}$$

where $\alpha \in [0, 1]$ and $\phi \in [0, 2\pi)$. Analogously the output state will be the state taken after tracing out the 8 ancillary qubits and we will denote it as

$$\rho_{out} = \begin{bmatrix} m_{11} & m_{12} \\ m_{12}^* & 1-m_{11} \end{bmatrix}, \tag{5.2}$$

which is the basic description of a density operator.

5.1 Analytical expressions for different error models

The analytical expressions for each of the approximations shown here can be found in Annex 2. For clarity and ease of reading, we will only include Taylor series approximations for small values of γ, δ. All presented graphs and thresholds were found using complete analytical formulas.

5.1.1 Depolarizing Channel

The presented results are for the operators $\mathcal{E}^{\gamma}_{\mathrm{DC}}$ for the gate error model and $\mathcal{E}^{\delta}_{\mathrm{DC}}$ for the channel error model (see Eq. (2.7)). An approximation (around $\gamma = 0, \delta = 0$) for the resulting state is

$$\begin{aligned} m_{11} &\approx \alpha^2\left(1-\frac{27}{2}\delta^2-4\gamma-\frac{287}{2}\gamma^2\right)+\frac{27}{4}\delta^2+(2+45\delta)\gamma+\frac{287}{4}\gamma^2 \\ m_{12} &\approx \frac{1}{4}\alpha\sqrt{1-\alpha^2}e^{-i\phi}\Big[4+9\delta^2\left(-5+3e^{2i\phi}\right)+\gamma\left(-100+6\delta\left(-17+11e^{2i\phi}\right)\right)+ \\ &\quad \gamma^2\left(1148+40e^{2i\phi}\right)\Big]. \end{aligned} \tag{5.3}$$

5.1.2 Bit flip

The presented results are for the operators $\mathcal{E}^{\gamma}_{\mathrm{BF}}$ for the gate error model and $\mathcal{E}^{\delta}_{\mathrm{BF}}$ for the channel error model (see Eq. (2.4)). An approximation (around $\gamma = 0, \delta = 0$) for the resulting state is

$$\begin{aligned} m_{11} &\approx \alpha^2(1-8\gamma+2\gamma^2)+4\gamma-\gamma^2 \\ m_{12} &\approx \alpha\sqrt{1-\alpha^2}\Big[e^{-i\phi}\left(1-18\delta^2-36\delta\gamma+529\gamma^2\right)+ \\ &\quad e^{i\phi}\left(4\gamma-121\gamma^2\right)\Big]. \end{aligned} \tag{5.4}$$

5.1.3 Phase flip

The presented results are for the operators $\mathcal{E}^{\gamma}_{\mathrm{PF}}$ for the gate error model and $\mathcal{E}^{\delta}_{\mathrm{PF}}$ for the channel error model (see Eq. (2.5)). An approximation (around $\gamma = 0, \delta = 0$) for the resulting state is

$$\begin{aligned} m_{11} &\approx \alpha^2(1 - 54\delta^2 - 288\delta\gamma - 384\gamma^2) + 27\delta^2 + 144\delta\gamma + 192\gamma^2 \\ m_{12} &\approx \alpha\sqrt{1-\alpha^2}e^{-i\phi}\Big[1 + 27\left(-1 + e^{2i\phi}\right)\delta^2 \\ &\quad -4\left(5 - 36\delta\left(-1 + e^{2i\phi}\right)\right)\gamma + 12\left(-1 + 16e^{2i\phi}\right)\gamma^2\Big]. \end{aligned} \tag{5.5}$$

5.1.4 Bit-phase flip

The presented results are for the operators $\mathcal{E}^{\gamma}_{\mathrm{BPF}}$ for the gate error model and $\mathcal{E}^{\delta}_{\mathrm{BPF}}$ for the channel error model (see Eq. (2.6)). An approximation (around $\gamma = 0, \delta = 0$) for the resulting state is

$$\begin{aligned} m_{11} &\approx \alpha^2\left(1 - 54\delta^2 - 8\gamma - 574\gamma^2\right) + 27\delta^2 + (4 + 180\delta)\gamma + 287\gamma^2 \\ m_{12} &\approx \alpha\sqrt{1-\alpha^2}e^{-i\phi}\Big[1 + 9\left(-5 + 3e^{2i\phi}\right)\delta^2 - 2\left(-6\delta + 2e^{2i\phi} + 23\right)\gamma \\ &\quad + \left(989 + 199e^{2i\phi}\right)\gamma^2\Big]. \end{aligned} \tag{5.6}$$

5.1.5 Amplitude damping

The presented results are for the operators $\mathcal{E}^{\gamma}_{\mathrm{AD}}$ for the gate error model and $\mathcal{E}^{\delta}_{\mathrm{AD}}$ for the channel error model (see Eq. (2.12)). An approximation (around $\gamma = 0, \delta = 0$) for the

resulting state is

$$\begin{aligned} m_{11} &\approx \alpha^2\left(1-\frac{27\delta^2}{8}+\gamma(-27\delta-4)-52\gamma^2\right)+ \\ &\quad \frac{27}{16}\delta^2+\left(4+\frac{63}{4}\delta\right)\gamma+34\gamma^2 \\ m_{12} &\approx \frac{1}{32}\alpha\sqrt{1-\alpha^2}e^{-i\phi}\Big[32+18\left(-19+3e^{2i\phi}\right)\delta^2- \\ &\quad 4\left(160+6\left(19+e^{2i\phi}\right)\delta\right)\gamma+\left(5896-8e^{2i\phi}\right)\gamma^2\Big]. \end{aligned} \tag{5.7}$$

5.2 Coding vs no-coding analysis - upper bound

In this section we will analyze (and give an upper bound) for when is it better to encode the qubit instead of sending it through the same kind of channel without encoding it. To do this, we will assume that the gate error is null ($\gamma = 0$), and thus the threshold values provided below are an upper bound on the actual values. However, if we assume that $\gamma \ll \delta$ then these results hold approximately.

The criterion we will use to decide when is it better to encode (or not) will be the expected value of the fidelity squared between the input state and output state, comparing the encoding version to the non-encoding version. When the expected value of the fidelity of non-encoding is greater than encoding, we will say it is not worth encoding at all.

After setting the gate error to null, the resulting implementation is as shown in Fig. 5.4.

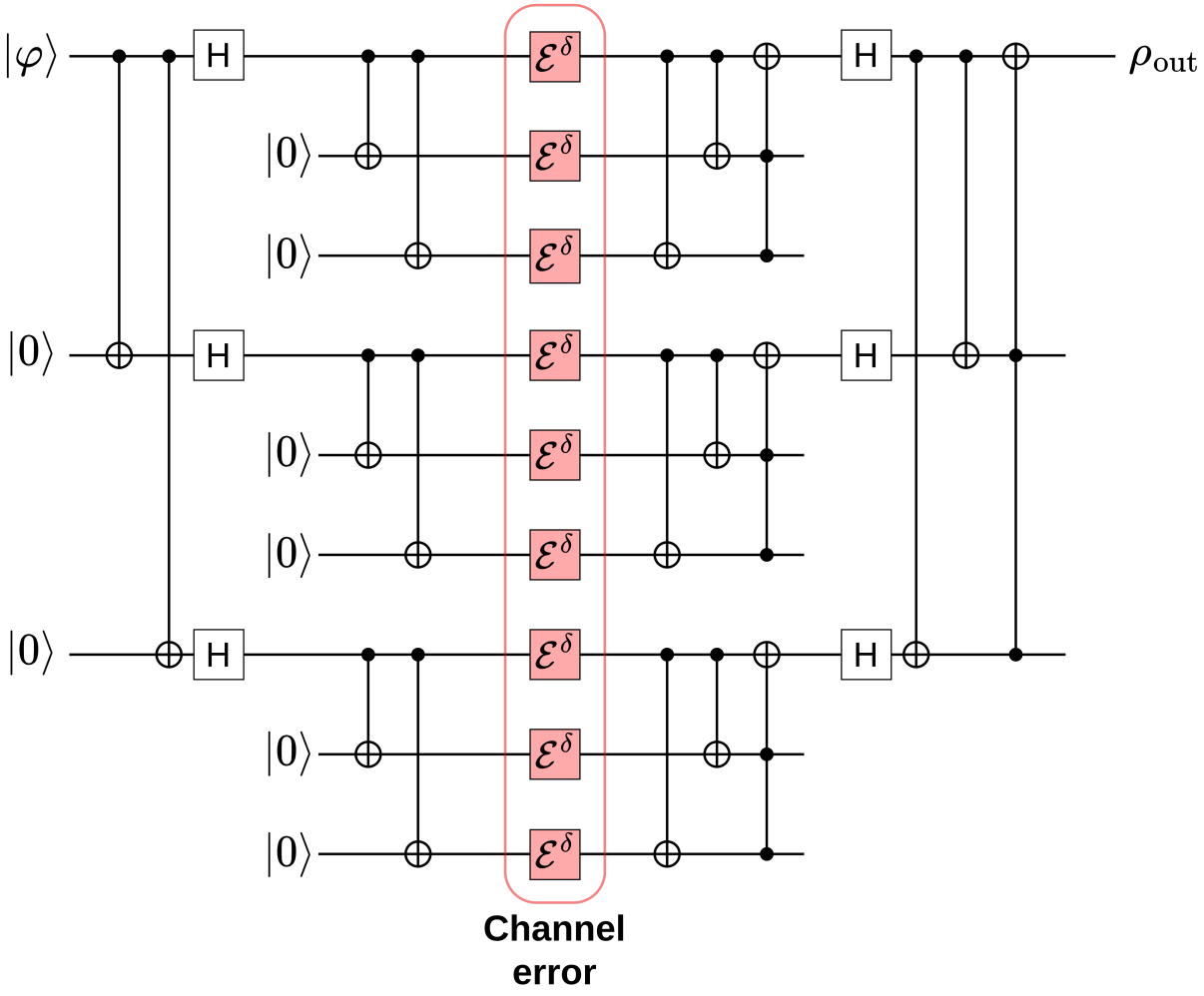

Figure 5.4: Shor code implementation only with channel error

We want to define a criterion for when it is better to encode a qubit using the Shor code. For this purpose, we will use the fidelity Eq. (1.42) as follows. Let us call the input state

$$\begin{aligned} \rho_{in} &= |\varphi\rangle\langle\varphi| \\ |\varphi\rangle &= \alpha|0\rangle + \alpha\sqrt{1-\alpha^2}e^{-i\phi}|1\rangle\,, \end{aligned} \tag{5.8}$$

where $\alpha \in [0,1]$ and $\phi \in [0, 2\pi)$, the output after going through Shor code ρ_{out} , and the output from sending the qubit without encoding through the same channel $\hat{\rho}_{out}$. The comparison we are going to do can be described by Fig. 5.5.

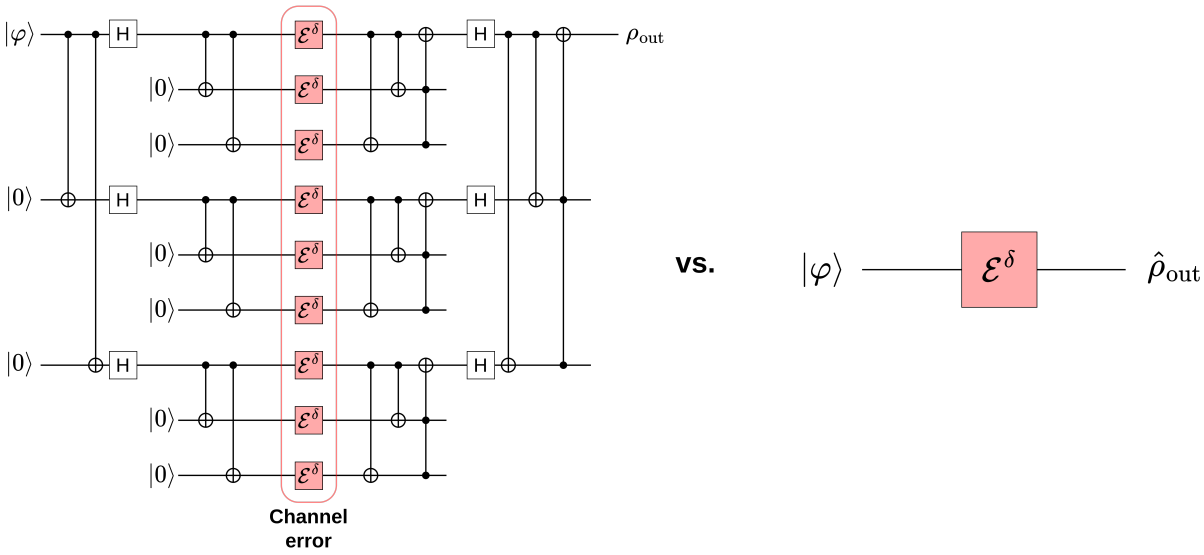

Figure 5.5: Coding vs. no-coding comparison - upper bound

The criterion for encoding vs. non-encoding we chose is, as said before, the expected value of the fidelity squared. When this value is greater than or equal without encoding than with encoding, then it does not make sense to encode. We will also assume that we have equal probability of sending any state (this assumption can be modified easily).

The expected value of the fidelity squared for a random variable state $|\psi\rangle$, with probability density function $p(x)$ defined in a domain Ω , and an output state ρ is

$$\mu_{p,\rho} = E\left[F^2(|\psi\rangle, \rho)\right] = \int_{x\in\Omega} F^2\left[|\psi(x)\rangle, \rho(x)\right] p(x) d\Omega. \tag{5.9}$$

For equal probability of sending any state (Ω is the whole Bloch sphere), we get a uniform distribution and, using the description for the state $|\varphi\rangle$ as in (5.8) ($p(x) = 1/(2\pi)$), this simplifies to

$$\mu_\rho = E\left[F^2(|\varphi\rangle, \rho)\right] = \frac{1}{2\pi}\int_0^1 \int_0^{2\pi} F^2\left[|\varphi(\alpha,\phi)\rangle, \rho(\alpha,\phi)\right] d\phi d\alpha, \tag{5.10}$$

where we have made explicit that the reference state and output state depend on the parameters α and ϕ.

In particular, we want to find out (for each type of modeled error) if there is a region for the error width δ in which $\mu_{\hat{\rho}_{out}} - \mu_{\rho_{out}} \geq 0$. This is the region where we will say it is better to send the state without encoding, since the expected value of the fidelity is higher without coding than with coding.

The results obtained here are by using the complete formulas (see Annex 2), setting $\gamma = 0$ and integrating over the whole Bloch sphere, as in Eq. (5.10)

5.2.1 Depolarizing Channel

The result obtained for the expected fidelity after encoding with the Shor code is

$$\mu_{\rho_{out}} = \frac{1}{20}(\delta - 2)^2 \left(3\delta^7 - 15\delta^6 + 39\delta^5 - 57\delta^4 + 57\delta^3 - 27\delta^2 + 5\delta + 5\right), \tag{5.11}$$

while sending it without encoding is

$$\mu_{\hat{\rho}_{out}} = 1 - \frac{\delta}{2}. \tag{5.12}$$

These results can be seen in Fig. 5.6 and Fig. 5.7.

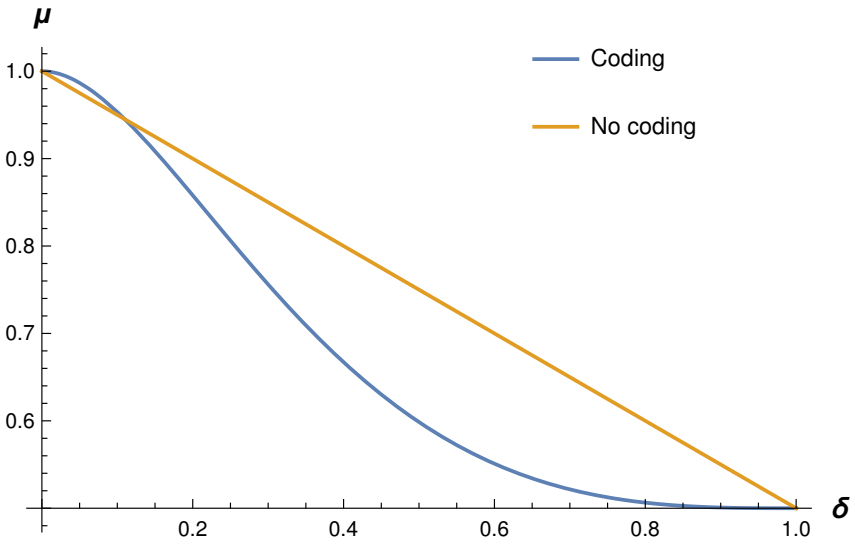

Figure 5.6: DC - Expected fidelity squared.

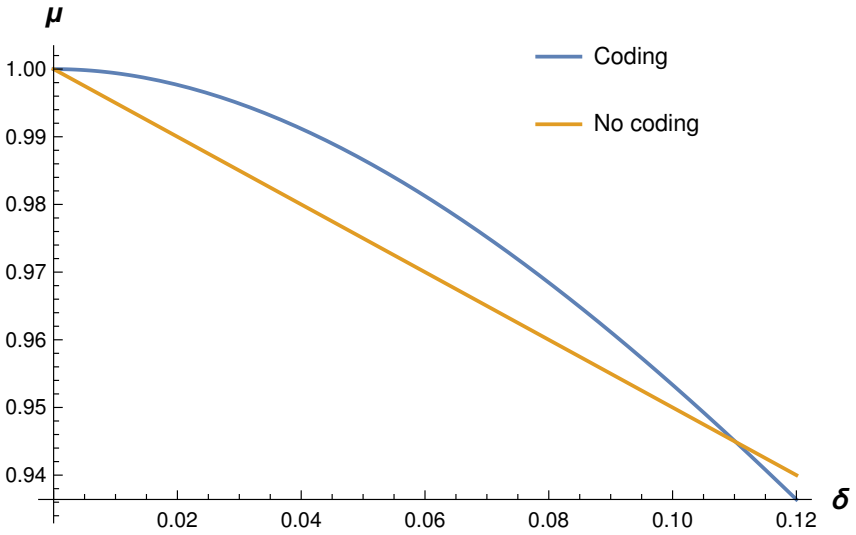

Figure 5.7: DC - Expected fidelity squared, zoomed in.

And hence, an upper bound for sending the data without coding is

$$\delta_{\text{threshold}} \approx 0.110. \tag{5.13}$$

5.2.2 Bit Flip

The result obtained for the expected fidelity after encoding with the Shor code is

$$\mu_{\rho_{out}} = 1 - \frac{8}{15}\delta^2(3-2\delta)\left(2(2\delta-3)\left(4\delta^3-6\delta^2+3\right)\delta^2+3\right), \tag{5.14}$$

while sending it without encoding is

$$\mu_{\hat{\rho}_{out}} = 1 - \frac{11}{15}\delta. \tag{5.15}$$

These results can be seen in Fig. 5.8 and Fig. 5.9.

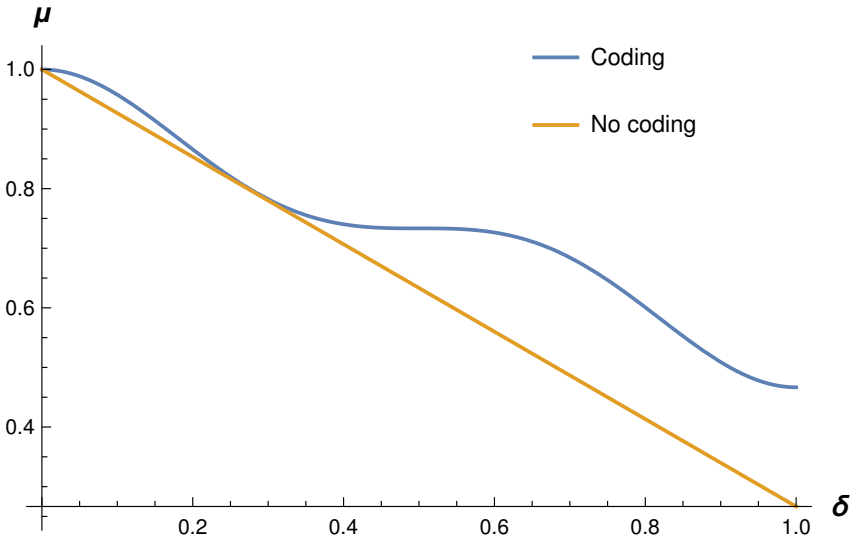

Figure 5.8: BF - Expected fidelity squared.

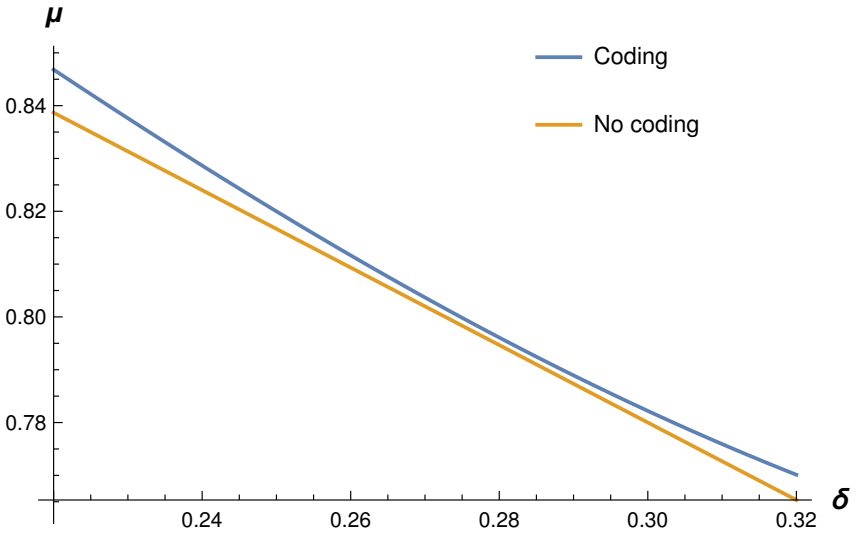

Figure 5.9: BF - Expected fidelity squared, zoomed in.

And hence, there is not an upper bound for sending the data without coding and, with this approximation, it is always better to encode.

5.2.3 Phase Flip

The result obtained for the expected fidelity after encoding with the Shor code is

$$\mu_{\rho_{out}} = 1 - \frac{11}{15}\delta^2 \left(3 - 2\delta \left(4\delta^2 - 6\delta + 3\right)\right) \left(4\delta^2 - 6\delta + 3\right)^2, \tag{5.16}$$

while sending it without encoding is

$$\mu_{\hat{\rho}_{out}} = 1 - \frac{8}{15}\delta. \tag{5.17}$$

These results can be seen in Fig. 5.10 and Fig. 5.11.

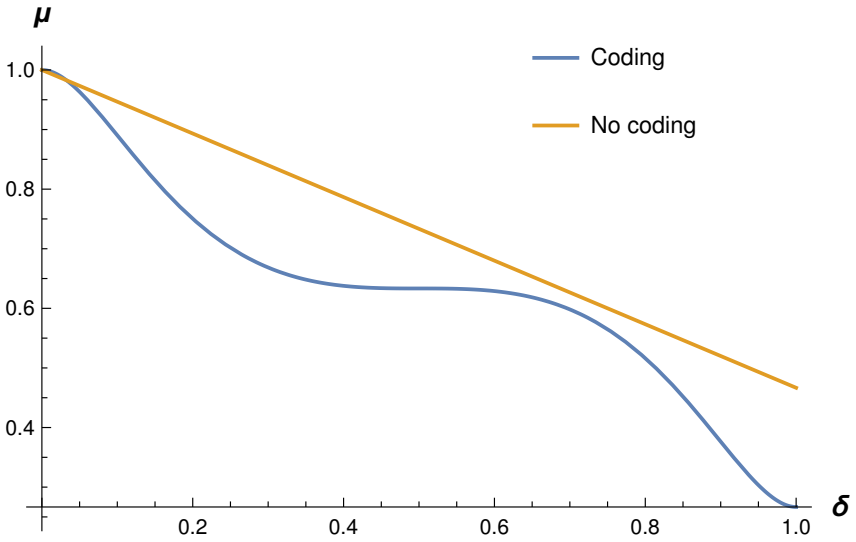

Figure 5.10: PF - Expected fidelity squared.

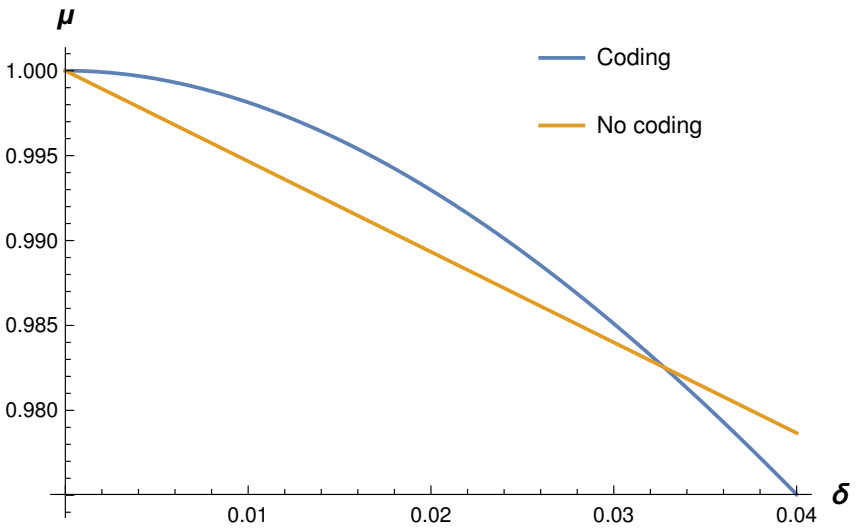

Figure 5.11: PF - Expected fidelity squared, zoomed in.

And hence, an upper bound for sending the data without coding is

$$\delta_{\text{threshold}} \approx 0.0328. \tag{5.18}$$

5.2.4 Bit-Phase Flip

The result obtained for the expected fidelity after encoding with the Shor code is

$$\mu_{\rho_{out}} = 1 + \frac{1024\delta^9}{15} - 288\delta^8 + \frac{2592\delta^7}{5} - 560\delta^6 + \frac{2256\delta^5}{5} - 288\delta^4 + 122\delta^3 - \frac{123\delta^2}{5}, \tag{5.19}$$

while sending it without encoding is

$$\mu_{\hat{\rho}_{out}} = 1 - \frac{11}{15}\delta. \tag{5.20}$$

These results can be seen in Fig. 5.12 and Fig. 5.13.

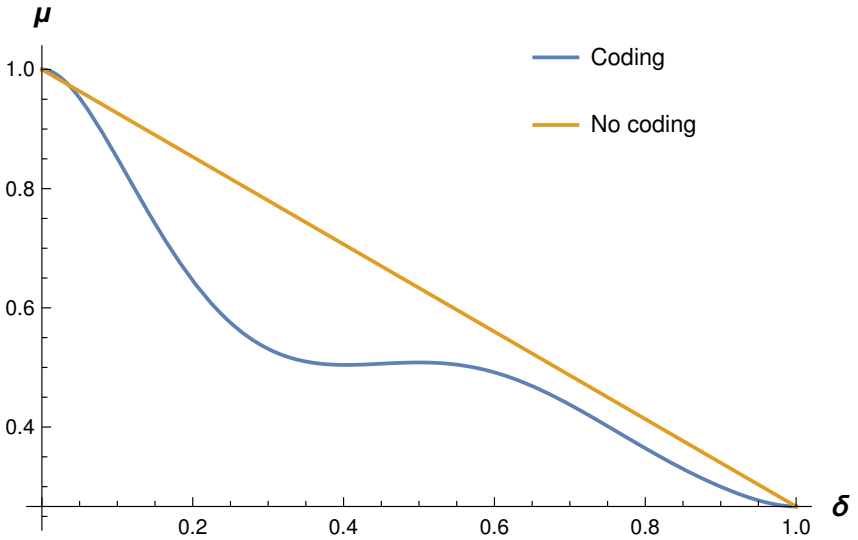

Figure 5.12: BPF - Expected fidelity squared.

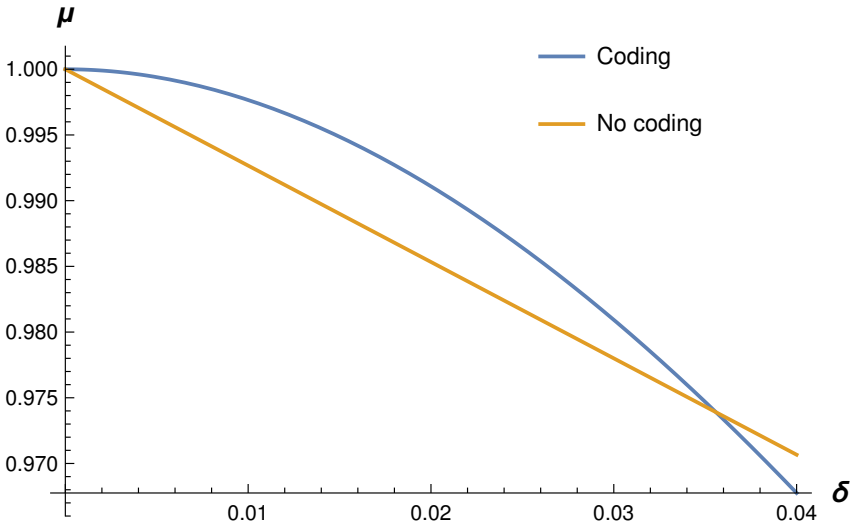

Figure 5.13: BPF - Expected fidelity squared, zoomed in.

And hence, an upper bound for sending the data without coding is

$$\delta_{\text{threshold}} \approx 0.0356. \tag{5.21}$$

5.2.5 Amplitude Damping

The result obtained for the expected fidelity after encoding with the Shor code is

$$\begin{aligned}\mu_{\rho_{out}} &= \frac{1}{60}\sqrt{1-\delta}\Big(-48\delta^7+192\delta^6-252\delta^5+49\delta^4+\\ &\quad 164\delta^3-138\delta^2+11\delta+22\Big)+\\ &\quad \frac{1}{60}\Big[64\delta^9-288\delta^8+432\delta^7-120\delta^6-288\delta^5+216\delta^4\\ &\quad +48\delta^3-72\delta^2+38\Big],\end{aligned} \tag{5.22}$$

while sending it without encoding is

$$\mu_{\hat{\rho}_{out}} = \frac{1}{15}\left(-6\delta+4\sqrt{1-\delta}+11\right). \tag{5.23}$$

These results can be seen in Fig. 5.14 and Fig. 5.15.

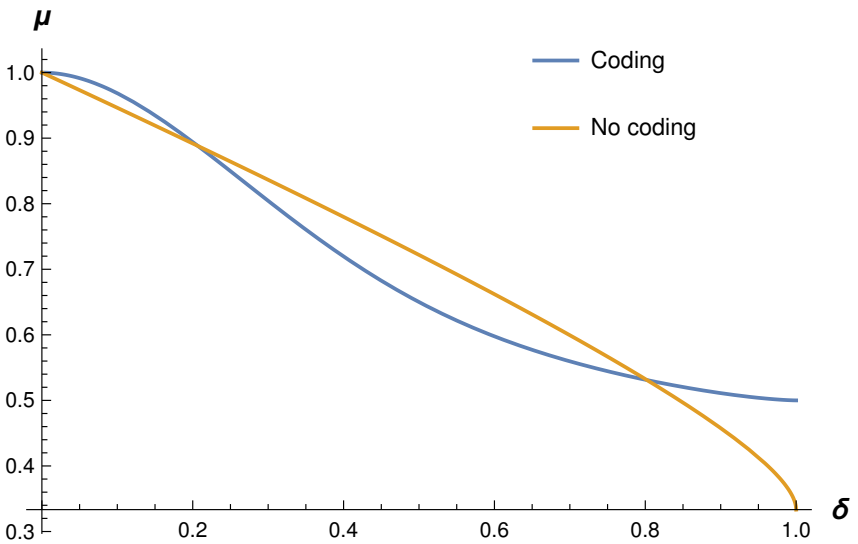

Figure 5.14: AD - Expected fidelity squared.

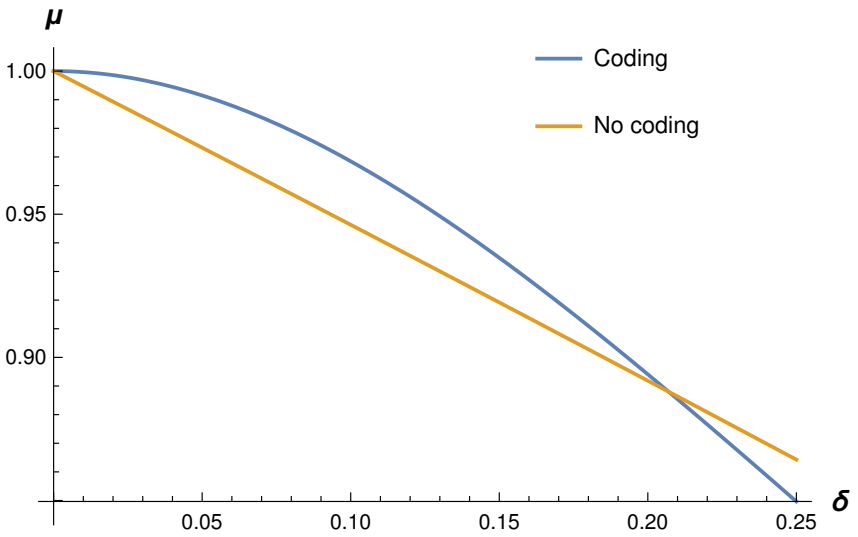

Figure 5.15: AD - Expected fidelity squared, zoomed in.

And hence, an upper bound for sending the data without coding is

$$\delta_{\text{threshold}} \approx 0.207. \tag{5.24}$$

5.3 Coding vs no-coding analysis - exact results

In the previous setting, we found an upper bound by setting $\gamma = 0$. Now, we will not assume this restriction and therefore try to find regions for which region in the γ, δ plane we find $\mu_{\hat{\rho}_{out}} - \mu_{\rho_{out}} \geq 0$. In this region, we will say that it is better not to use the Shor code.

We believe this to be a more fair comparison, since using the Shor code implies also encoding and decoding, and thus also gate error. The comparison we will make can be described by Fig. 5.16. Again, all results shown are obtained using the analytical formulas in Annex 2.

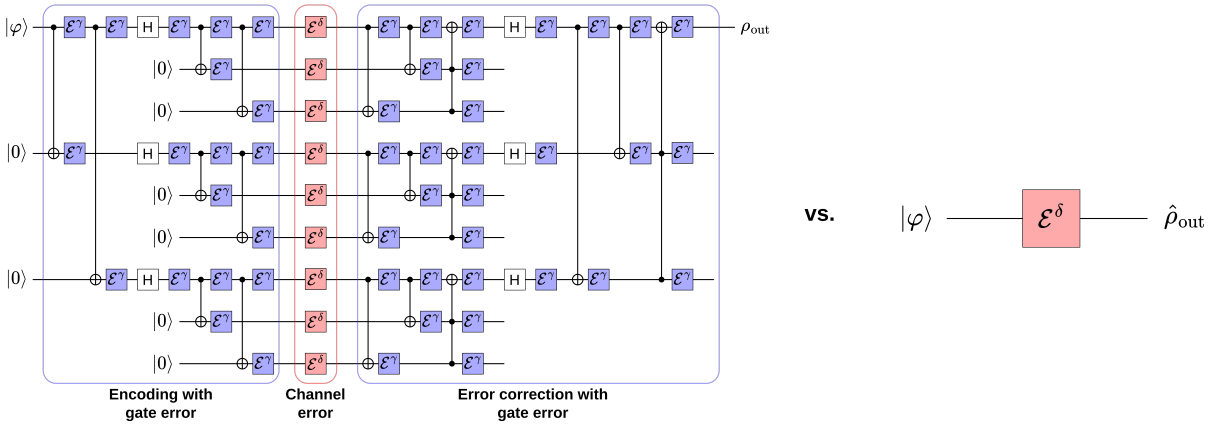

Figure 5.16: Coding vs. no-coding comparison - exact results

5.3.1 Depolarizing Channel

In Figs. 5.17 and 5.18 the expected value of the fidelity squared can be seen for the coding and without coding (using or not the Shor code). In Fig. 5.19 the region for which it is still better to use the Shor code is shown in gray and was found by finding the contour plot $\mu_{\hat{\rho}_{out}} - \mu_{\rho_{out}} = 0$.

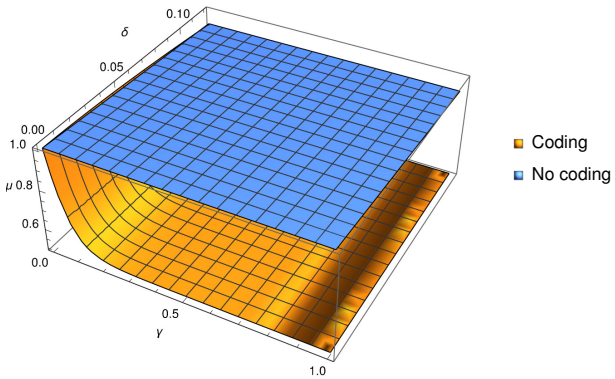

Figure 5.17: DC - Expected fidelity squared.

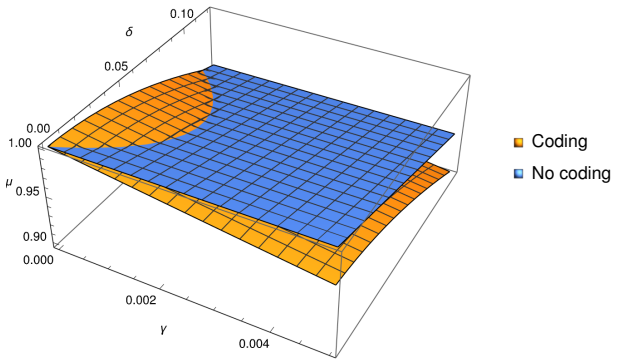

Figure 5.18: DC - Expected fidelity squared, zoomed in.

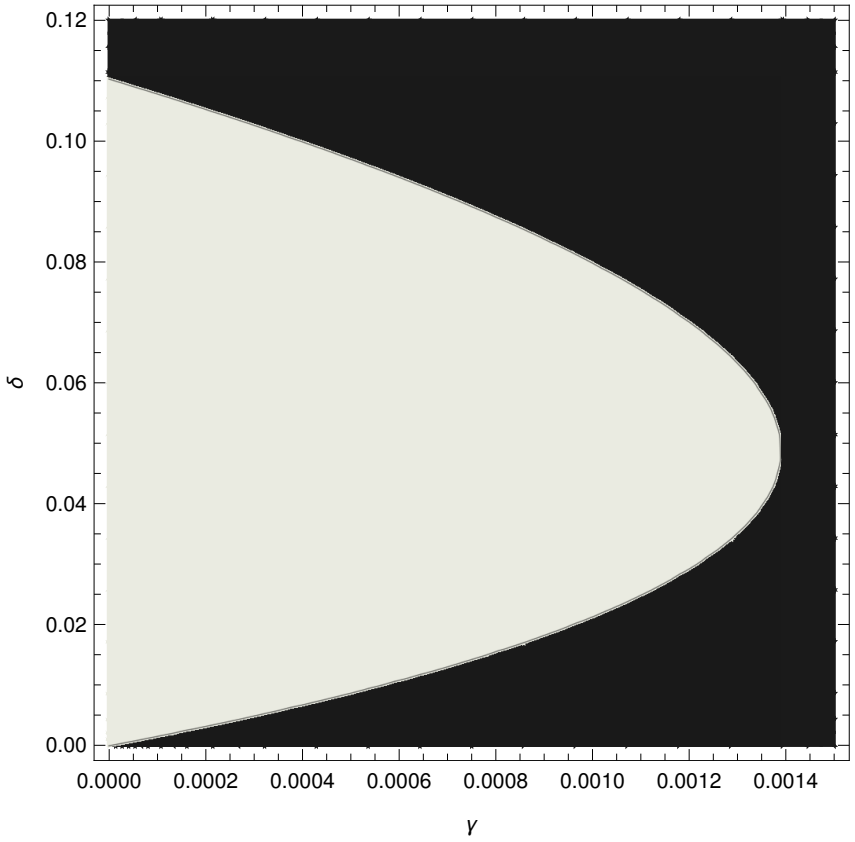

Figure 5.19: DC - Regions for coding vs. no coding.

5.3.2 Bit Flip

In Figs. 5.20 and 5.21 the expected value of the fidelity squared can be seen for the coding and without coding (using or not the Shor code). In Figs. 5.22 and 5.23 the region for which it is still better to use the Shor code is shown in gray and was found by finding the contour plot $\mu_{\hat{\rho}_{out}} - \mu_{\rho_{out}} = 0$.

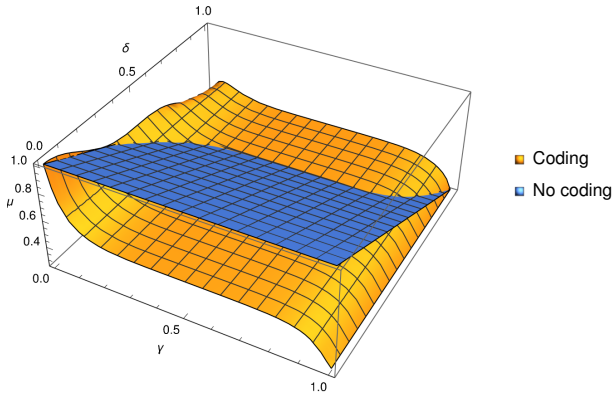

Figure 5.20: BF - Expected fidelity squared.

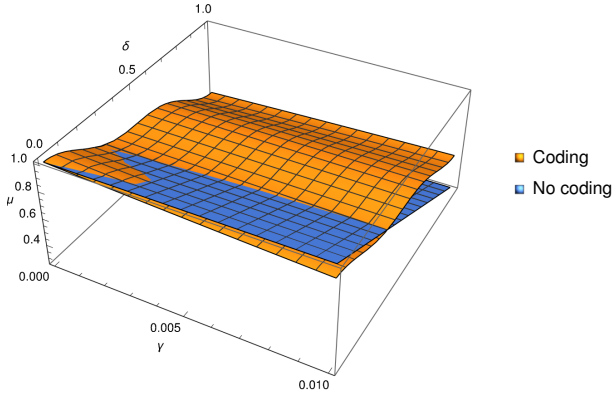

Figure 5.21: BF - Expected fidelity squared, zoomed in.

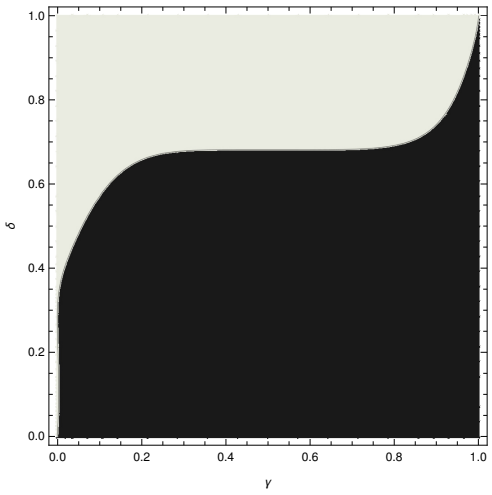

Figure 5.22: BF - Regions for coding vs. no coding.

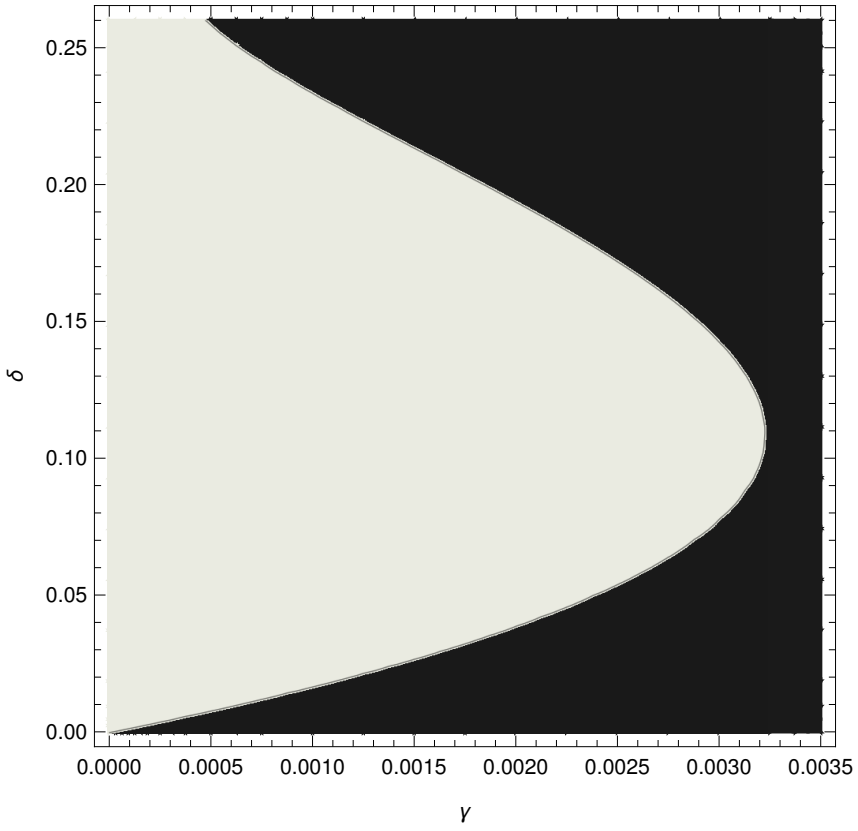

Figure 5.23: BF - Regions for coding vs. no coding, zoomed in.

5.3.3 Phase Flip

In Figs. 5.24 and 5.25 the expected value of the fidelity squared can be seen for the coding and without coding (using or not the Shor code). In Fig. 5.26 the region for which it is still better to use the Shor code is shown in gray and was found by finding the contour plot $\mu_{\hat{\rho}_{out}} - \mu_{\rho_{out}} = 0$.

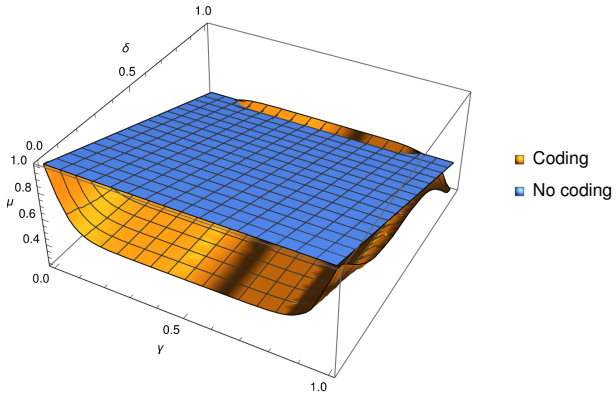

Figure 5.24: PF - Expected fidelity squared.

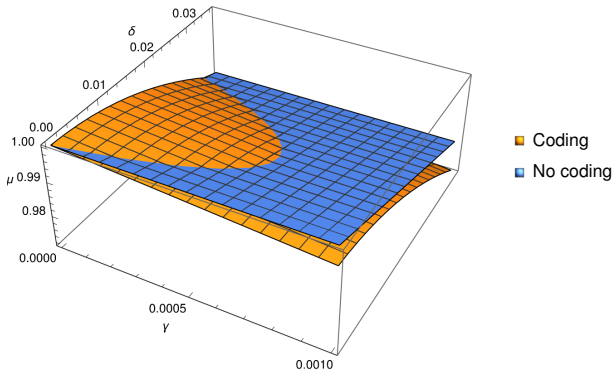

Figure 5.25: PF - Expected fidelity squared, zoomed in.

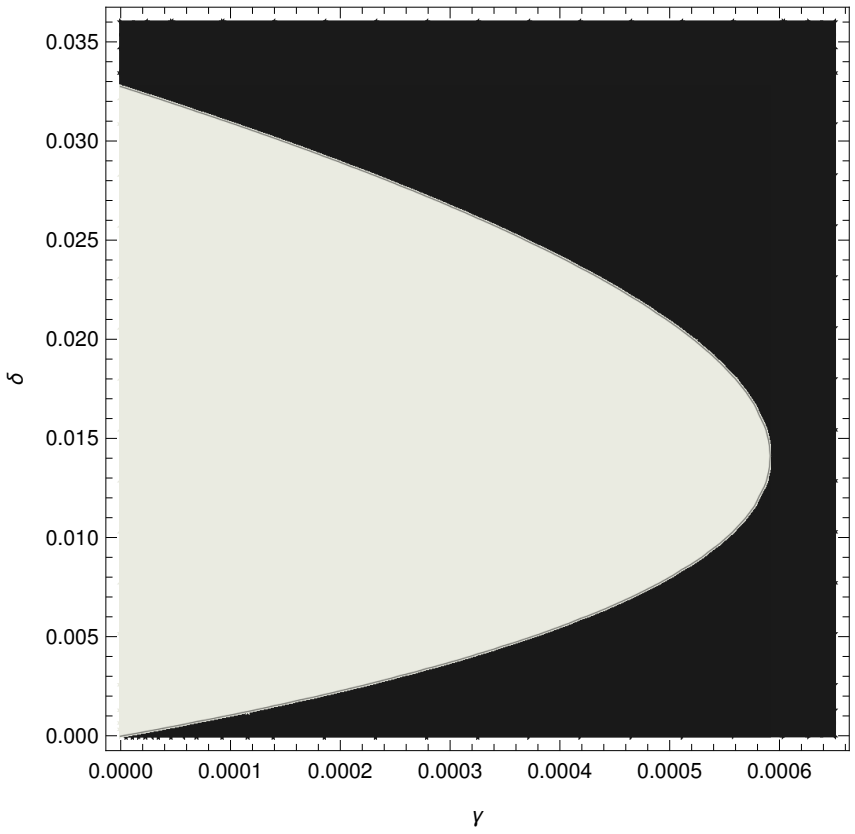

Figure 5.26: PF - Regions for coding vs. no coding.

5.3.4 Bit-Phase Flip

In Figs. 5.27 and 5.28 the expected value of the fidelity squared can be seen for the coding and without coding (using or not the Shor code). In Fig. 5.29 the region for which it is still better to use the Shor code is shown in gray and was found by finding the contour plot $\mu_{\hat{\rho}_{out}} - \mu_{\rho_{out}} = 0$.

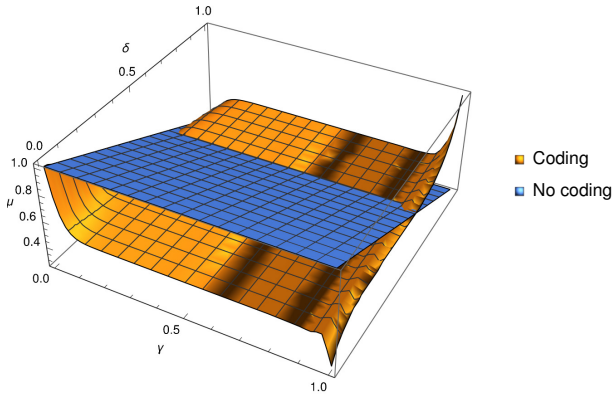

Figure 5.27: BPF - Expected fidelity squared.

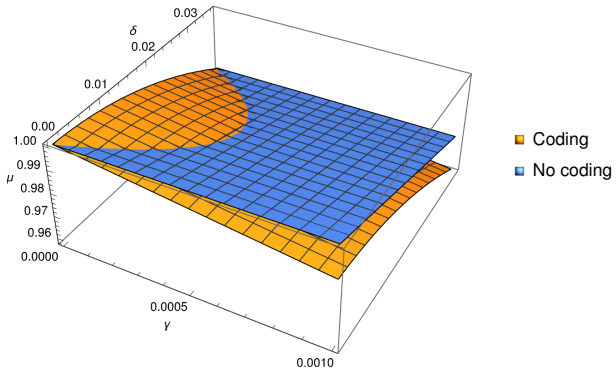

Figure 5.28: BPF - Expected fidelity squared, zoomed in.

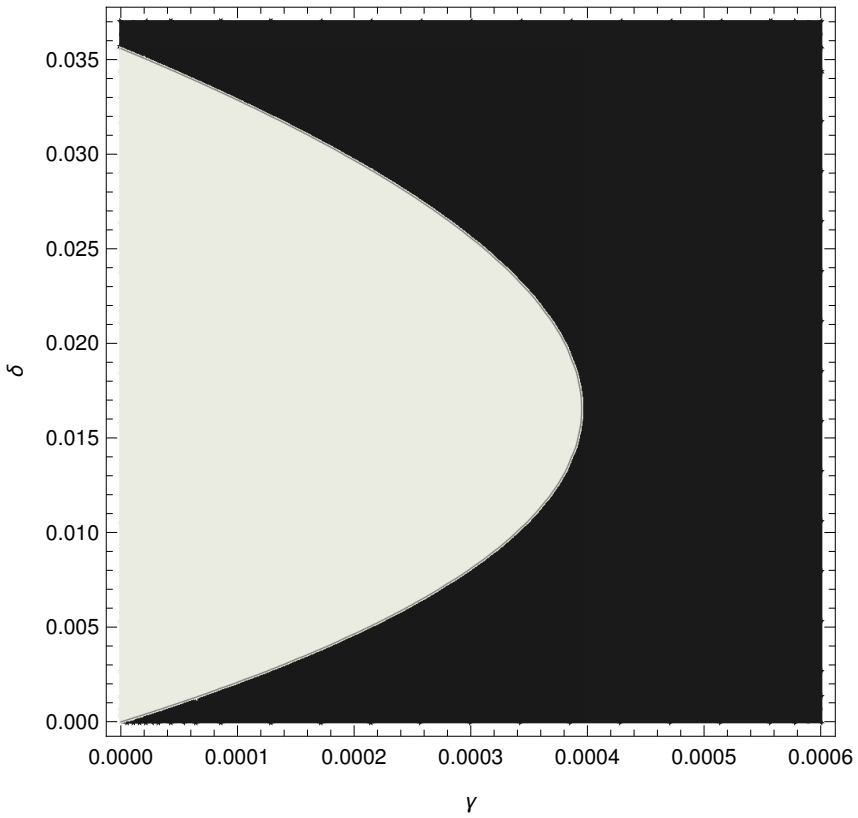

Figure 5.29: BPF - Regions for coding vs. no coding.

5.3.5 Amplitude Damping

In Figs. 5.30 and 5.31 the expected value of the fidelity squared can be seen for the coding and without coding (using or not the Shor code). In Fig. 5.32 the region for which it is still better to use the Shor code is shown in gray and was found by finding the contour plot $\mu_{\hat{\rho}_{out}} - \mu_{\rho_{out}} = 0$.

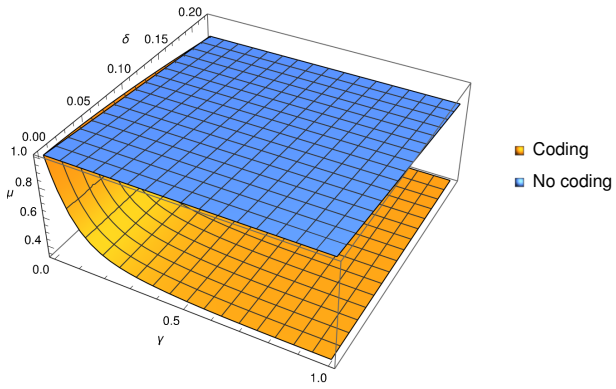

Figure 5.30: AD - Expected fidelity squared.

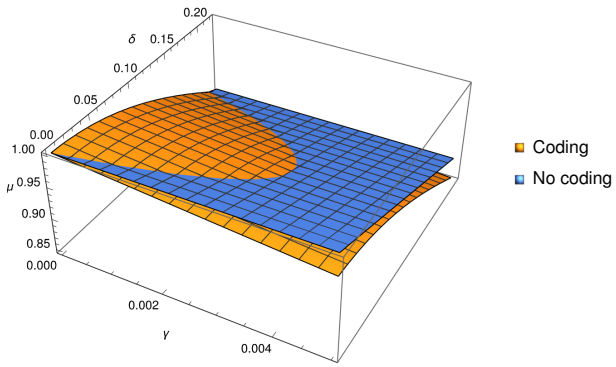

Figure 5.31: AD - Expected fidelity squared, zoomed in.

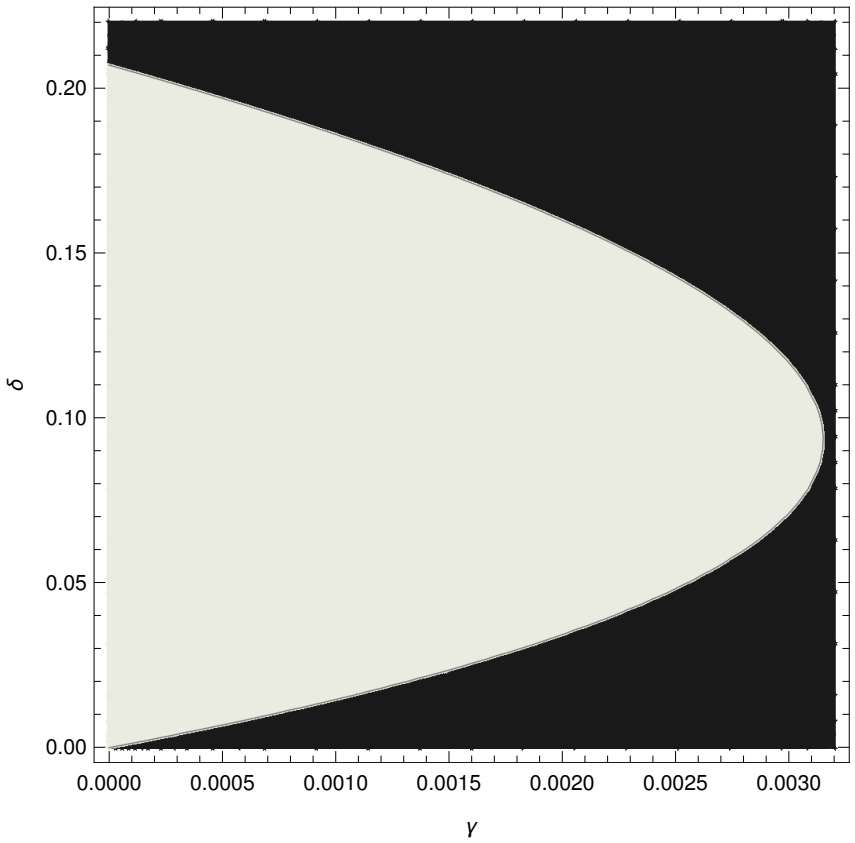

Figure 5.32: AD - Regions for coding vs. no coding.

6. Conclusions

In this thesis, we presented an overview of quantum computing with its postulates. An introduction to quantum error modeling was also explored, with focus on various channels such as bit flip and amplitude damping. We have also introduced classical linear codes, and introduced a generalization to quantum linear codes.

We have studied in depth the Shor code, focusing on how it can be built with the concatenation of two 3-qubit repetition codes for bit-flip and phase-flip. We have focused on the effect of error in an implementation of the Shor code, where novel analytical expressions have been found for different error models. The regions for which it is worth to use the Shor code instead of sending the information without encoding, both by giving a simple upper bound and an exact result based on the analytical formulas obtained, have been shown. This is particularly useful for the transmission of quantum information, where the quantum error of the channel might not be negligible.

Apart from this, we have introduced a method to simplify and minimize the computational complexity needed for symbolical calculations with tools such as *Wolfram Mathematica* [33], specifically designed for quantum computing.

For future work, we propose to be use the described methodology to change the error models in accordance to a specific hardware. For example, one could study the gates for *IBM Q* [36] or *Rigetti* [37, 38], model their errors with statistical analysis and apply the methods to find analytical results. Another possible study is to introduce models for correlating errors such as in CNOT and Toffoli gates. With these results, a number of conclusions can be found (by classical simulation and analytical analysis) without the need to do statistical analysis for the whole circuit and input in the corresponding hardware.

References

[1] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru=Guzik, "Quantum chemistry in the age of quantum computing" *Chemical Reviews*, vol. 119, no. 19, pp. 10 856–10 915, aug 2019.

[2] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, feb 2016.

[3] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, "Quantum computational chemistry," *Reviews of Modern Physics*, vol. 92, no. 1, mar 2020.

[4] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, sep 2018.

[5] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, may 2019.

[6] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, no. 3, pp. 226–245, jul 2019.

[7] E. T. Campbell, B. M. Terhal, and C. Vuillot, "Roads towards fault-tolerant universal quantum computation," *Nature*, vol. 549, no. 7671, pp. 172–179, sep 2017.

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

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 7th ed. Cambridge, England: Cambridge University Press, 2011.

[10] A. L. Fonseca de Oliveira, "Propagación de errores en circuitos cuánticos e isotropıa," Ph.D. dissertation, Universidad Politecnica de Madrid, 2016.

[11] I. B. Djordjevic, *Quantum Information Processing, Quantum Computing, and Quantum Error Correction: An Engineering Approach*, 2nd ed. London, England: Academic Press, 2021.

[12] T. Toffoli, "Reversible computing," in *Automata, Languages and Programming*. Springer Berlin Heidelberg, 1980, pp. 632–644.

[13] R. Jozsa, "Fidelity for mixed quantum states," *Journal of Modern Optics*, vol. 41, no. 12, pp. 2315–2323, dec 1994.

[14] K. Kraus, A. Böhm, J. D. Dollard, and W. H. Wootters, Eds., *States, Effects, and Operations Fundamental Notions of Quantum Theory*. Springer Berlin Heidelberg, 1983.

[15] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, apr 1950.

[16] R. Singleton, "Maximum distance q-nary codes," *IEEE Transactions on Information Theory*, vol. 10, no. 2, pp. 116–118, apr 1964.

[17] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, jun 1960.

[18] E. N. Gilbert, "A comparison of signalling alphabets," *Bell System Technical Journal*, vol. 31, no. 3, pp. 504–522, may 1952.

[19] R. R. Varshamov, "Estimate of the number of signals in error correcting codes," *Doklady Akademii Nauk SSSR*, vol. 117, no. 3, pp. 739–741, 1957.

[20] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, jul 1948.

[21] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, oct 1982.

[22] D. Dieks, "Communication by EPR devices," *Physics Letters A*, vol. 92, no. 6, pp. 271–272, nov 1982.

[23] D. E. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, 1997. [Online]. Available: https://resolver.caltech.edu/CaltechETD:etd-07162004-113028

[24] A. Ashikhmin and S. Litsyu, "Upper bounds on the size of quantum codes," *IEEE Transactions on Information Theory*, vol. 45, no. 4, pp. 1206–1215, may 1999.

[25] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, no. 2, pp. 1098–1105, aug 1996.

[26] A. Steane, "Multiple-particle interference and quantum error correction," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 452, no. 1954, pp. 2551–2577, nov 1996.

[27] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *Quantum Info. Comput.*, vol. 6, no. 2, p. 97–165, mar 2006.

[28] B. M. Terhal, "Quantum error correction for quantum memories," *Reviews of Modern Physics*, vol. 87, no. 2, pp. 307–346, apr 2015.

[29] P. Shor, "Fault-tolerant quantum computation," in *Proceedings of 37th Conference on Foundations of Computer Science*. IEEE Comput. Soc. Press, 1996.

[30] J. Preskill, "Reliable quantum computers," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 385–410, jan 1998.

[31] M. A. Nielsen, "A simple formula for the average gate fidelity of a quantum dynamical operation," *Physics Letters A*, vol. 303, no. 4, pp. 249–252, oct 2002.

[32] E. Knill, R. Laflamme, and W. H. Zurek, "Resilient quantum computation," *Science*, vol. 279, no. 5349, pp. 342–345, jan 1998.

[33] S. Wolfram, "Mathematica," Champaign, IL, 1988.

[34] Scilab, 2012. [Online]. Available: http://www.scilab.org

[35] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, aug 2018.

[36] IBM Q team, "Ibm q experience and its quantum computation roadmap". [Online] https://www.research.ibm.com/ibm-q/technology/roadmap/, 2023. Accessed on: August 5, 2023.

[37] R. Computing, "Rigetti computing - quantum computing for the real world," 2023. [Online]. Available: https://www.rigetti.com. Accessed on: August 5, 2023.

[38] C. Rigetti and M. H. Devoret, "Fully microwave-tunable universal gates in superconducting qubits with linear interactions and fixed transition frequencies," *Physical Review B*, vol. 81, no. 13, p. 134507, 2010.

Annex 1: Efficient calculation of analytical formulas

The method used to simplify algebraic expressions is divided into two optimizations: the first one is to only use the upper triangular part of the density operator to perform the calculations, and the second one is the recurrent extraction of parameters due to linearity.

1.1 Upper triangular matrix optimization

Suppose we have an n-qubit density matrix $\rho = (\rho_{ij})$ (with $N = 2^n$ basis states). The result of a quantum operation $\mathcal{E}$, given by its OSR with the set $\{E_m\}$ on the density operator ρ can be achieved by performing calculations only in its upper triangular matrix.

This is trivial to prove since the resulting operator must also be a density matrix σ (hermitian), and we have

$$\sigma_{ab} \;=\; \langle a|\,\mathcal{E}(\rho)\,|b\rangle = \langle a|\left(\sum_m E_m \rho E_m^\dagger\right)|b\rangle = \langle a|\left(\sum_{m,i,j} \rho_{ij} E_m\,|i\rangle\,\langle j|\,E_m^\dagger\right)|b\rangle\,,$$

where the elements in the summation can be computed efficiently by calculating elements in the upper triangular matrix and conjugating and transposing because

$$\left(\rho_{ij} E_m\,|i\rangle\,\langle j|\,E_m^\dagger\right)^\dagger = \rho_{ij}^* E_m\,|j\rangle\,\langle i|\,E_m^\dagger = \rho_{ji} E_m\,|j\rangle\,\langle i|\,E_m^\dagger,$$

which is changing i with j and conjugating. Hence, computing the σ density operator can be achieved efficiently by computing the upper triangular matrix of the input state ρ instead of the whole matrix.

1.2 Recurrent extraction of parameters

Since in quantum computing an operation $\mathcal{E}$ is linear (not on parameters, but by using matrix multiplication), we can exploit this to store (persist) a lot of matrices and remove

parameters.

Let's recall that the resulting density matrix after applying $\mathcal{E}$ is

$$\mathcal{E}(\rho) = \sum_m E_m \rho E_m^\dagger = \sum_{m,i,j} \rho_{ij} E_m |i\rangle \langle j| E_m^\dagger.$$

The idea is to store the result $E_m |i\rangle \langle j| E_m^\dagger$ for each coefficient and propagate it "simple enough" (notice the matrix $|i\rangle \langle j|$ is a matrix with only one element equal to 1 in the ij position and the rest equal to 0). So, instead of doing small costly operations, we will do a lot of easier operations, store them separately and then combine it with its coefficient.

Let us start by showing an example using an Amplitude Damping channel on a 1-qubit state (see Eq. (2.12)). Since the density matrix ρ is a 1-qubit state it can be defined as

$$\begin{aligned}
\rho &= \begin{bmatrix} \rho_{00} & \rho_{10} \\ \rho_{10}^* & \rho_{11} \end{bmatrix} = \rho_{00} |0\rangle \langle 0| + \rho_{01} |0\rangle \langle 1| + \rho_{10} |1\rangle \langle 0| + \rho_{11} |1\rangle \langle 1| \\
&= \rho_{00} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \rho_{01} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \rho_{10} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + \rho_{11} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.
\end{aligned}$$

We will show the example for the term $\rho_{11} |1\rangle \langle 1|$, the rest is analogous. By applying linearity we can see that the contribution to the output given by the input ρ_{11} is

$$\rho_{11} \left(E_0 |1\rangle \langle 1| E_0^\dagger + E_1 |1\rangle \langle 1| E_1^\dagger \right),$$

and now we can compute $E_0 |1\rangle \langle 1| E_0^\dagger$ and $E_1 |1\rangle \langle 1| E_1^\dagger$ and store its results

$$\begin{aligned}
E_0 |1\rangle \langle 1| E_0^\dagger &= \begin{bmatrix} 0 & 0 \\ 0 & 1-\gamma \end{bmatrix} = (1-\gamma) |1\rangle \langle 1| \\
E_1 |1\rangle \langle 1| E_1^\dagger &= \begin{bmatrix} \gamma & 0 \\ 0 & 0 \end{bmatrix} = \gamma |0\rangle \langle 0|,
\end{aligned}$$

and the contribution due to ρ_{11} becomes

$$\rho_{11}(1-\gamma)\left|1\right\rangle\left\langle 1\right| + \rho_{11}\gamma\left|0\right\rangle\left\langle 0\right|.$$

So if we wanted to apply some other operation we can decompose the result again in terms of $\rho'_{00}\left|0\right\rangle\left\langle 0\right|$, $\rho'_{01}\left|0\right\rangle\left\langle 1\right|$, $\rho'_{10}\left|1\right\rangle\left\langle 0\right|$ and $\rho'_{11}\left|1\right\rangle\left\langle 1\right|$, which gives the algorithm its power by recursion and persistence.

Let us define the algorithm. Given an operation with OSR $\mathcal{E}$ applied to a matrix W the result is

$$\mathcal{E}(W) = \sum_{m,i,j} w_{ij} E_m \left|i\right\rangle\left\langle j\right| E_m^\dagger.$$

The algorithm for the function extract(W) is as follows [1]

1. For $i = 0$ to $N-1, j = 0$ to $N-1$ do
   (a) Save the number w_{ij}.
   (b) For each m do
      i. Execute $E_m \left|i\right\rangle\left\langle j\right| E_m^\dagger$
      ii. If the result has more than one term decompose it as above and return $w_{ij}\sum_{a,b} w'_{ab}$extract $\left(E_m \left|a\right\rangle\left\langle b\right| E_m^\dagger\right)$ [2]
      iii. If the result has only one term it can be decomposed as $w_{ab}\left|a\right\rangle\left\langle b\right|$ for some elements of the basis return the matrix $w_{ij}w_{ab}\left|a\right\rangle\left\langle b\right|$ (end the recursion)

1.3 Efficient algorithm with recurrent extraction

We will now modify the algorithm to reduce the number of operations by using only the upper triangular matrix and conjugating the output.

[1] Optionally, you can add a step either at ii or iii to simplify terms. Whether or not this is better to perform algebraic simplification depends on the problem and can be more of a heuristic approach.

[2] If memory is not enough, use persistence to track these calls and effectively compute the result in the end.

The efficient algorithm with recurrent extraction can be achieved by combining the efforts of recurrent extraction and efficiently getting the transpose conjugate to get the output of the input via its upper triangular matrix. In fact, we change j to go from $j = i$ to $N - 1$ and then sum the terms with conjugate terms w_{ij}^* , w_{ab}^* and $conjugateTranspose(partial)$ where you save the result of extract in $partial = extract(E_m |a\rangle \langle b| E_m^\dagger)$.

The algorithm for the function extractEfficient(W) is as follows [3]

1. Set $sum = 0$, $partial = \emptyset$.
2. For each m do
   (a) Execute $E_m |i\rangle \langle j| E_m^\dagger$
   (b) For $i = 0$ to $N - 1, j = i$ to $N - 1$ do
      i. Save the number w_{ij}.
      ii. If the result has more than one term decompose it as above and add to the list $partial$ the term $partial_{mab} = \text{extractEfficient}\left(E_m |a\rangle \langle b| E_m^\dagger\right)$ [4]
      iii. If the result has only one term it can be decomposed as $w_{ab} |a\rangle \langle b|$ for some elements of the basis return the matrix $w_{ij} w_{ab} |a\rangle \langle b|$ (end the recursion)
   (c) Set
   $$sum_m = \sum_{i=0}^{N-1} \sum_{j=i}^{N-1} w_{ij} partial_{mij} + \sum_{i=1}^{N-1} \sum_{j=0}^{i-1} w_{ij}^* (partial_{mij})^\dagger$$
   (d) Set $sum = sum + sum_m$
3. Return sum

[3] Optionally, you can add a step either at ii or iii to simplify terms. Whether or not this is better to perform algebraic simplification depends on the problem and can be more of a heuristic approach

[4] If memory is not enough, use persistence to track these calls and effectively compute the result in the end

Annex 2: Analytical formulas

The following are the analytical formulas for an input state as in Eq. (5.1) and output state as in Eq. (5.2), where we use m_{11} = m11, m_{12} = m12, α = a, ϕ = phi, γ = g, and δ = d.

2.1 Depolarizing Channel - DC

```
m11 = (-2203961430*g^16*(-1+d)^9+2333606220*g^17*(-1+d)^9-2203961430*
    g^18*(-1+d)^9+1855967520*g^19*(-1+d)^9-1391975640*g^20*(-1+d)
    ^9+927983760*g^21*(-1+d)^9-548354040*g^22*(-1+d)^9+286097760*g
    ^23*(-1+d)^9-131128140*g^24*(-1+d)^9+52451256*g^25*(-1+d)
    ^9-18156204*g^26*(-1+d)^9+5379616*g^27*(-1+d)^9-1344904*g^28*(-1+d
    )^9+278256*g^29*(-1+d)^9-46376*g^30*(-1+d)^9+5984*g^31*(-1+d)
    ^9-561*g^32*(-1+d)^9+34*g^33*(-1+d)^9-g^34*(-1+d)^9-d^2*(3-3*d+d
    ^2)^2*(-3+3*d-3*d^2+d^3)+2*g*(-1+d)^3*(-4-102*d+255*d^2-340*d
    ^3+255*d^4-102*d^5+17*d^6)+17*g^3*(-1+d)^3*(287-2112*d+5280*d
    ^2-7040*d^3+5280*d^4-2112*d^5+352*d^6)-g^2*(-1+d)^3*(287-3366*d
    +8415*d^2-11220*d^3+8415*d^4-3366*d^5+561*d^6)-748*g^6*(-1+d)
    ^3*(1785-10788*d+26970*d^2-35960*d^3+26970*d^4-10788*d^5+1798*d^6)
    +44*g^5*(-1+d)^3*(6181-37944*d+94860*d^2-126480*d^3+94860*d
    ^4-37944*d^5+6324*d^6)-4290*g^10*(-1+d)^3*(30565-183396*d+458490*d
    ^2-611320*d^3+458490*d^4-183396*d^5+30566*d^6)-g^4*(-1+d)
    ^3*(43295-278256*d+695640*d^2-927520*d^3+695640*d^4-278256*d
    ^5+46376*d^6)+858*g^9*(-1+d)^3*(61123-366792*d+916980*d^2-1222640*
    d^3+916980*d^4-366792*d^5+61132*d^6)+11*g^7*(-1+d)
    ^3*(488003-2934336*d+7335840*d^2-9781120*d^3+7335840*d^4-2934336*d
    ^5+489056*d^6)-33*g^8*(-1+d)^3*(549863-3301128*d+8252820*d
    ^2-11003760*d^3+8252820*d^4-3301128*d^5+550188*d^6)+120*g^13*(-1+d
    )^3*(7733197-46399188*d+115997970*d^2-154663960*d^3+115997970*d
    ^4-46399188*d^5+7733198*d^6)+13*g^11*(-1+d)^3*(22007381-132045120*
    d+330112800*d^2-440150400*d^3+330112800*d^4-132045120*d
```

```
^5+22007520*d^6)-13*g^12*(-1+d)^3*(42181037-253086480*d+632716200*
d^2-843621600*d^3+632716200*d^4-253086480*d^5+42181080*d^6)-8*g
^14*(-1+d)^3*(173996953-1043981730*d+2609954325*d^2-3479939100*d
^3+2609954325*d^4-1043981730*d^5+173996955*d^6)+g^15*(-1+d)
^3*(1855967519-11135805120*d+27839512800*d^2-37119350400*d
^3+27839512800*d^4-11135805120*d^5+1855967520*d^6)+2*a^2*(-1+g)
^13*(-1+d)^3*(2+1330*g^3*(-1+d)^6-5985*g^4*(-1+d)^6+20349*g^5*(-1+
d)^6-54264*g^6*(-1+d)^6+116280*g^7*(-1+d)^6-203490*g^8*(-1+d)
^6+293930*g^9*(-1+d)^6-352716*g^10*(-1+d)^6+352716*g^11*(-1+d)
^6-293930*g^12*(-1+d)^6+203490*g^13*(-1+d)^6-116280*g^14*(-1+d)
^6+54264*g^15*(-1+d)^6-20349*g^16*(-1+d)^6+5985*g^17*(-1+d)
^6-1330*g^18*(-1+d)^6+210*g^19*(-1+d)^6-21*g^20*(-1+d)^6+g^21*(-1+
d)^6+6*d-15*d^2+20*d^3-15*d^4+6*d^5-d^6+g^2*(-209+1260*d-3150*d
^2+4200*d^3-3150*d^4+1260*d^5-210*d^6)+3*g*(6-42*d+105*d^2-140*d
^3+105*d^4-42*d^5+7*d^6)))/4;

m12 = (a*Sqrt[(1-a^2)]*(-1+g)^25*(-1+d)^3*(2*(2+g^3*(-1+d)^2+2*d+3*g
*(-2+d)*d-d^2+g^2*(-2+6*d-3*d^2))^3*Cos[phi]+I*(-1+g)^3*(-2+g^2)
*(-1+d)^2*(-8+138*g^7*(-1+d)^4-38*g^8*(-1+d)^4-24*g^9*(-1+d)^4+24*
g^10*(-1+d)^4-8*g^11*(-1+d)^4+g^12*(-1+d)^4-40*d+24*d^2-4*d^3+d^4-
g^6*(-1+d)^2*(151-310*d+155*d^2)-12*g*(2-18*d+13*d^2-4*d^3+d^4)+3*
g^2*(44-176*d+156*d^2-68*d^3+17*d^4)+2*g^5*(6-50*d+105*d^2-80*d
^3+20*d^4)+3*g^4*(55-176*d+200*d^2-112*d^3+28*d^4)-2*g^3*(110-370*
d+389*d^2-204*d^3+51*d^4))*Sin[phi]))/16;
```

2.2 Bit Flip Channel - BF

```
m11 = -(a^2*(-1+2*g)^5*(1+2*g-18*g^2+40*g^3-40*g^4+16*g^5))+g*(4-g
-110*g^2+620*g^3-1792*g^4+3232*g^5-3808*g^6+2880*g^7-1280*g^8+256*
g^9);

m12 = a*Sqrt[(1-a^2)]*(-1+2*g)^15*(-1+2*d)^3*(1+4*g^3*(1-2*d)^2+2*d
```

```
+12*g*(-1+d)*d-2*d^2-4*g^2*(1-6*d+6*d^2))^3*(Cos[phi]+I*(-1+2*g)
^5*(1+2*g-18*g^2+40*g^3-40*g^4+16*g^5)*Sin[phi]);
```

2.3 Phase Flip Channel - PF

```
m11 = a^2*(1-2*g)^8*(-1+2*d)^3*(-1-16*g*(1-2*d)^6+240*g^2*(1-2*d)
^6-2240*g^3*(1-2*d)^6+14560*g^4*(1-2*d)^6-69888*g^5*(1-2*d)
^6+256256*g^6*(1-2*d)^6-732160*g^7*(1-2*d)^6+1647360*g^8*(1-2*d)
^6-2928640*g^9*(1-2*d)^6+4100096*g^10*(1-2*d)^6-4472832*g^11*(1-2*
d)^6+3727360*g^12*(1-2*d)^6-2293760*g^13*(1-2*d)^6+983040*g
^14*(1-2*d)^6-262144*g^15*(1-2*d)^6+32768*g^16*(1-2*d)^6-6*d+30*d
^2-80*d^3+120*d^4-96*d^5+32*d^6)-(-3+6*d-12*d^2+8*d^3-16*g*(-1+2*d
)^3+112*g^2*(-1+2*d)^3-448*g^3*(-1+2*d)^3+1120*g^4*(-1+2*d)
^3-1792*g^5*(-1+2*d)^3+1792*g^6*(-1+2*d)^3-1024*g^7*(-1+2*d)
^3+256*g^8*(-1+2*d)^3)*(-8*g*(-1+2*d)^3+56*g^2*(-1+2*d)^3-224*g
^3*(-1+2*d)^3+560*g^4*(-1+2*d)^3-896*g^5*(-1+2*d)^3+896*g^6*(-1+2*
d)^3-512*g^7*(-1+2*d)^3+128*g^8*(-1+2*d)^3+d*(3-6*d+4*d^2))^2;

m12 = a*Sqrt[1-a a]*(1-2*g)^10*(Cos[(phi)]-I*(1-2*g)^8*(-1+2*d)
^3*(-1-16*g*(1-2*d)^6+240*g^2*(1-2*d)^6-2240*g^3*(1-2*d)^6+14560*g
^4*(1-2*d)^6-69888*g^5*(1-2*d)^6+256256*g^6*(1-2*d)^6-732160*g
^7*(1-2*d)^6+1647360*g^8*(1-2*d)^6-2928640*g^9*(1-2*d)^6+4100096*g
^10*(1-2*d)^6-4472832*g^11*(1-2*d)^6+3727360*g^12*(1-2*d)
^6-2293760*g^13*(1-2*d)^6+983040*g^14*(1-2*d)^6-262144*g^15*(1-2*d
)^6+32768*g^16*(1-2*d)^6-6*d+30*d^2-80*d^3+120*d^4-96*d^5+32*d^6)*
Sin[(phi)]);
```

2.4 Bit-Phase Flip Channel - BPF

```
m11 = -36109704069120*g^16*(-1+2*d)^9+76467608616960*g^17*(-1+2*d)
^9-144438816276480*g^18*(-1+2*d)^9+243265374781440*g^19*(-1+2*d)
^9-364898062172160*g^20*(-1+2*d)^9+486530749562880*g^21*(-1+2*d)
```

^9−574990885847040∗g^22∗(−1+2∗d)^9+599990489579520∗g^23∗(−1+2∗d)
^9−549991282114560∗g^24∗(−1+2∗d)^9+439993025691648∗g^25∗(−1+2∗d)
^9−304610556248064∗g^26∗(−1+2∗d)^9+180509959258112∗g^27∗(−1+2∗d)
^9−90254979629056∗g^28∗(−1+2∗d)^9+37346888122368∗g^29∗(−1+2∗d)
^9−12448962707456∗g^30∗(−1+2∗d)^9+3212635537408∗g^31∗(−1+2∗d)
^9−602369163264∗g^32∗(−1+2∗d)^9+73014444032∗g^33∗(−1+2∗d)
^9−4294967296∗g^34∗(−1+2∗d)^9−d^2∗(3−6∗d+4∗d^2)^2∗(−3+6∗d−12∗d
^2+8∗d^3)+4∗g∗(−1+2∗d)^3∗(−1−51∗d+255∗d^2−680∗d^3+1020∗d^4−816∗d
^5+272∗d^6)+ 34∗g^3∗(−1+2∗d)^3∗(287−4224∗d+21120∗d^2−56320∗d
^3+84480∗d^4−67584∗d^5+22528∗d^6)−g^2∗(−1+2∗d)^3∗(287−6732∗d
+33660∗d^2−89760∗d^3+134640∗d^4−107712∗d^5+35904∗d^6)−11968∗g
^6∗(−1+2∗d)^3∗(1785−21576∗d+107880∗d^2−287680∗d^3+431520∗d^4−
345216∗d^5+115072∗d^6)+352∗g^5∗(−1+2∗d)^3∗(6181−75888∗d+379440∗d
^2−1011840∗d^3+1517760∗d^4−1214208∗d^5+404736∗d^6)−1098240∗g
^10∗(−1+2∗d)^3∗(30565−366792∗d+1833960∗d^2−4890560∗d^3+7335840∗d
^4−5868672∗d^5+1956224∗d^6)−4∗g^4∗(−1+2∗d)^3∗(43295−556512∗d
+2782560∗d^2−7420160∗d^3+11130240∗d^4−8904192∗d^5+2968064∗d^6)
+109824∗g^9∗(−1+2∗d)^3∗(61123−733584∗d+3667920∗d^2−9781120∗d
^3+14671680∗d^4−11737344∗d^5+3912448∗d^6)+352∗g^7∗(−1+2∗d)
^3∗(488003−5868672∗d+29343360∗d^2−78248960∗d^3+117373440∗d
^4−93898752∗d^5+31299584∗d^6)−2112∗g^8∗(−1+2∗d)^3∗(549863−6602256∗
d+33011280∗d^2−88030080∗d^3+132045120∗d^4−105636096∗d^5+35212032∗d
^6)+245760∗g^13∗(−1+2∗d)^3∗(7733197−92798376∗d+463991880∗d
^2−1237311680∗d^3+1855967520∗d^4−1484774016∗d^5+494924672∗d^6)
+6656∗g^11∗(−1+2∗d)^3∗(22007381−264090240∗d+1320451200∗d
^2−3521203200∗d^3+5281804800∗d^4−4225443840∗d^5+1408481280∗d^6)
−13312∗g^12∗(−1+2∗d)^3∗(42181037−506172960∗d+2530864800∗d^2−
6748972800∗d^3+10123459200∗d^4−8098767360∗d^5+2699589120∗d^6)
−32768∗g^14∗(−1+2∗d)^3∗(173996953−2087963460∗d+10439817300∗d
^2−27839512800∗d^3+41759269200∗d^4−33407415360∗d^5+11135805120∗d
^6)+8192∗g^15∗(−1+2∗d)^3∗(1855967519−22271610240∗d+111358051200∗d
^2−296954803200∗d^3+445432204800∗d^4−356345763840∗d

```
^5+118781921280*d^6)+a^2*(-1+2*g)^13*(-1+2*d)^3*(1+5320*g^3*(1-2*d
)^6-47880*g^4*(1-2*d)^6+325584*g^5*(1-2*d)^6-1736448*g^6*(1-2*d)
^6+7441920*g^7*(1-2*d)^6-26046720*g^8*(1-2*d)^6+75246080*g^9*(1-2*
d)^6-180590592*g^10*(1-2*d)^6+361181184*g^11*(1-2*d)^6-601968640*g
^12*(1-2*d)^6+833495040*g^13*(1-2*d)^6-952565760*g^14*(1-2*d)
^6+889061376*g^15*(1-2*d)^6-666796032*g^16*(1-2*d)^6+392232960*g
^17*(1-2*d)^6-174325760*g^18*(1-2*d)^6+55050240*g^19*(1-2*d)
^6-11010048*g^20*(1-2*d)^6+1048576*g^21*(1-2*d)^6+6*d-30*d^2+80*d
^3-120*d^4+96*d^5-32*d^6+6*g*(3-42*d+210*d^2-560*d^3+840*d^4-672*d
^5+224*d^6)-2*g^2*(209-2520*d+12600*d^2-33600*d^3+50400*d^4-40320*
d^5+13440*d^6));

m12 = a*Sqrt[(1-a^2)]*(1-2*g)^10*((-1+2*g)^15*(-1+2*d)^3*(1+4*g
^3*(1-2*d)^2+2*d+12*g*(-1+d)*d-2*d^2-4*g^2*(1-6*d+6*d^2))^3*Cos[(
phi)]-I*(1+16384*g^18*(1-2*d)^8-72*d^2+336*d^3-504*d^4+672*d
^6-576*d^7+144*d^8-24576*g^17*(1-2*d)^6*(5-24*d+24*d^2)+4096*g
^16*(1-2*d)^4*(105-1028*d+3508*d^2-4960*d^3+2480*d^4)-4096*g
^15*(1-2*d)^2*(227-3416*d+20240*d^2-60848*d^3+98424*d^4-81600*d
^5+27200*d^6)-2048*g^13*(1-2*d)^2*(721-13718*d+96162*d^2-323512*d
^3+558316*d^4-475872*d^5+158624*d^6)+1024*g^14*(1-2*d)
^2*(1349-22536*d+144120*d^2-456352*d^3+761136*d^4-639552*d
^5+213184*d^6)-256*g^9*(1-2*d)^2*(-177-11772*d+205312*d^2-1068376*
d^3+2237428*d^4-2043888*d^5+681296*d^6)-512*g^11*(1-2*d)
^2*(1183-35244*d+322696*d^2-1276056*d^3+2390908*d^4-2103456*d
^5+701152*d^6)+256*g^10*(1-2*d)^2*(647-35972*d+412984*d^2-1832728*
d^3+3613124*d^4-3236112*d^5+1078704*d^6)-32*g^7*(1-2*d)
^2*(-1401+14030*d+145898*d^2-1523536*d^3+3770968*d^4-3611040*d
^5+1203680*d^6)+64*g^8*(1-2*d)^2*(-1229-3092*d+303656*d^2-2016168*
d^3+4545684*d^4-4245120*d^5+1415040*d^6)+256*g^12*(1-2*d)
^2*(4461-101084*d+794652*d^2-2873984*d^3+5154112*d^4-4460544*d
^5+1486848*d^6)+g*(-22+60*d+924*d^2-4448*d^3+4200*d^4+9456*d
^5-22864*d^6+16896*d^7-4224*d^8)+2*g^2*(85-122*d-4522*d^2+17168*d
```

```
^3+1028*d^4-93608*d^5+167992*d^6-117248*d^7+29312*d^8)-4*g
^3*(173+496*d-15588*d^2+39688*d^3+84460*d^4-483744*d^5+758880*d
^6-512256*d^7+128064*d^8)+8*g^4*(173+3070*d-35738*d^2+37088*d
^3+447468*d^4-1666312*d^5+2400600*d^6-1581568*d^7+395392*d^8)+64*g
^6*(-211+5769*d-16459*d^2-222452*d^3+1551990*d^4-4056232*d
^5+5230936*d^6-3324736*d^7+831184*d^8)-8*g^5*(-83+15472*d-103068*d
^2-175416*d^3+2799620*d^4-8393392*d^5+11362960*d^6-7341568*d
^7+1835392*d^8))*Sin[(phi)]);
```

2.5 Amplitude Damping Channel - AD

```
m11 = (2-2*Sqrt[1-d]-(-54839+54840*a^2)*g^20*(1-d)^(9/2)+11646*(-1+a
^2)*g^21*(1-d)^(9/2)-1772*(-1+a^2)*g^22*(1-d)^(9/2)+172*(-1+a^2)*g
^23*(1-d)^(9/2)-8*(-1+a^2)*g^24*(1-d)^(9/2)-Sqrt[1-d]*d+6*Sqrt[1-d
]*d^2-4*Sqrt[1-d]*d^3+Sqrt[1-d]*d^4-2*a^2*Sqrt[1-d]*(-2-d+6*d^2-4*
d^3+d^4)+g^17*(-1+d)^3*(1444+1299753*Sqrt[1-d]-1299753*Sqrt[1-d]*d
+76*a^2*(-19-17115*Sqrt[1-d]+17115*Sqrt[1-d]*d))+g^19*(-1+d)
^3*(12+196975*Sqrt[1-d]-196975*Sqrt[1-d]*d+2*a^2*(-6-98497*Sqrt[1-
d]+98497*Sqrt[1-d]*d))-2*g^18*(-1+d)^3*(96+280487*Sqrt[1-d
]-280487*Sqrt[1-d]*d+a^2*(-96-280573*Sqrt[1-d]+280573*Sqrt[1-d]*d)
)+g^15*(-1+d)^3*(22357+4013428*Sqrt[1-d]-4013428*Sqrt[1-d]*d+4*a
^2*(-5579-1006468*Sqrt[1-d]+1006468*Sqrt[1-d]*d))-g^16*(-1+d)
^3*(6787+2494053*Sqrt[1-d]-2494053*Sqrt[1-d]*d+a^2*(-6784-2498082*
Sqrt[1-d]+2498082*Sqrt[1-d]*d))-g^2*(-2*(7+61*Sqrt[1-d])+(69+1028*
Sqrt[1-d])*d-3*(23+604*Sqrt[1-d])*d^2+(23+1208*Sqrt[1-d])*d^3-302*
Sqrt[1-d]*d^4+2*a^2*(-1+d)*(-4*(1+25*Sqrt[1-d])+(8+711*Sqrt[1-d])*
d-(4+711*Sqrt[1-d])*d^2+237*Sqrt[1-d]*d^3))-g^3*(4*(57+449*Sqrt[1-
d])-5*(141+1765*Sqrt[1-d])*d+3*(235+4686*Sqrt[1-d])*d^2-
(235+9372*Sqrt[1-d])*d^3+2343*Sqrt[1-d]*d^4-2*a^2*(-1+d)
*(-2*(32+643*Sqrt[1-d])+(128+4995*Sqrt[1-d])*d-(64+4995*Sqrt[1-d])
*d^2+1665*Sqrt[1-d]*d^3))-g^4*(-2*(725+5993*Sqrt[1-d])
+(4371+51259*Sqrt[1-d])*d-3*(1457+26182*Sqrt[1-d])*d
```

```
^2+(1457+52364*Sqrt[1-d])*d^3-13091*Sqrt[1-d]*d^4+8*a^2*(-1+d)
*(-122-1963*Sqrt[1-d]+4*(61+1605*Sqrt[1-d])*d-2*(61+3210*Sqrt[1-d
])*d^2+2140*Sqrt[1-d]*d^3))-g^5*(6232+54438*Sqrt[1-d
]-9*(2079+24716*Sqrt[1-d])*d+3*(6237+112004*Sqrt[1-d])*d
^2-(6237+224008*Sqrt[1-d])*d^3+56002*Sqrt[1-d]*d^4-2*a^2*(-1+d)
*(-2358-33292*Sqrt[1-d]+3*(1572+34223*Sqrt[1-d])*d- 3*(786+34223*
Sqrt[1-d])*d^2+34223*Sqrt[1-d]*d^3))-g^6*(-16*(1235+11806*Sqrt[1-d
])+(59277+760204*Sqrt[1-d])*d-3*(19759+380872*Sqrt[1-d])*d
^2+(19759+761744*Sqrt[1-d])*d^3-190436*Sqrt[1-d]*d^4+4*a^2*(-1+d)
*(-4046-54737*Sqrt[1-d]+(8092+165471*Sqrt[1-d])*d-(4046+165471*
Sqrt[1-d])*d^2+55157*Sqrt[1-d]*d^3))-g^7*(48298+526786*Sqrt[1-d
]-5*(28977+421970*Sqrt[1-d])*d+3*(48295+1055376*Sqrt[1-d])*d
^2-(48295+2110752*Sqrt[1-d])*d^3+527688*Sqrt[1-d]*d^4-4*a^2*(-1+d)
*(-7*(1495+20922*Sqrt[1-d])+(20930+439983*Sqrt[1-d])*d
-(10465+439983*Sqrt[1-d])*d^2+146661*Sqrt[1-d]*d^3))+g^9*(-1+d)
*(3*(48477+778825*Sqrt[1-d]) -6*(48477+1167883*Sqrt[1-d])*d
+3*(48477+2335766*Sqrt[1-d])*d^2-2335766*Sqrt[1-d]*d^3+4*a
^2*(-33891-615214*Sqrt[1-d]+6*(11297+307496*Sqrt[1-d])*d
-3*(11297+614992*Sqrt[1-d])*d^2+614992*Sqrt[1-d]*d^3))-g
^8*(-93458-1212101*Sqrt[1-d]+(280371+4848299*Sqrt[1-d])*d
-3*(93457+2424132*Sqrt[1-d])*d^2+(93457+4848264*Sqrt[1-d])*d
^3-1212066*Sqrt[1-d]*d^4+2*a^2*(-1+d)*(-42224-653223*Sqrt[1-d
]+2*(42224+979659*Sqrt[1-d])*d-2*(21112+979659*Sqrt[1-d])*d
^2+653106*Sqrt[1-d]*d^3))- g^14*(-1+d)*(54861+5463396*Sqrt[1-d
]-6*(18287+2731696*Sqrt[1-d])*d+3*(18287+5463392*Sqrt[1-d])*d
^2-5463392*Sqrt[1-d]*d^3+4*a^2*(-3*(4550+457799*Sqrt[1-d])
+12*(2275+343349*Sqrt[1-d])*d-6*(2275+686698*Sqrt[1-d])*d
^2+1373396*Sqrt[1-d]*d^3))-g^12*(-1+d)*(156091+6254935*Sqrt[1-d
]-26*(12007+721701*Sqrt[1-d])*d+13*(12007+1443402*Sqrt[1-d])*d
^2-6254742*Sqrt[1-d]*d^3+4*a^2*(-38324-1587271*Sqrt[1-d
]+26*(2948+183141*Sqrt[1-d])*d-26*(1474+183141*Sqrt[1-d])*d
^2+1587222*Sqrt[1-d]*d^3))-g^10*(-1+d)*(183469+3809144*Sqrt[1-d
```

```
]-2*(183469+5712510*Sqrt[1-d])*d+(183469+11425020*Sqrt[1-d])*d
^2-3808340*Sqrt[1-d]*d^3+2*a^2*(-143*(612+13795*Sqrt[1-d])
+102*(1716+58007*Sqrt[1-d])*d-102*(858+58007*Sqrt[1-d])*d
^2+1972238*Sqrt[1-d]*d^3))+g^11*(-1+d)*(188045+5283323*Sqrt[1-d
]-2*(188045+7924227*Sqrt[1-d])*d+(188045+15848454*Sqrt[1-d])*d
^2-5282818*Sqrt[1-d]*d^3+2*a^2*(-91234-2703775*Sqrt[1-d
]+2*(91234+4055265*Sqrt[1-d])*d-2*(45617+4055265*Sqrt[1-d])*d
^2+2703510*Sqrt[1-d]*d^3))+g^13*(-1+d)*(104039+6327442*Sqrt[1-d
]-2*(104039+9491100*Sqrt[1-d])*d+(104039+18982200*Sqrt[1-d])*d
^2-6327400*Sqrt[1-d]*d^3+2*a^2*(-51506-3193199*Sqrt[1-d
]+2*(51506+4789767*Sqrt[1-d])*d-2*(25753+4789767*Sqrt[1-d])*d
^2+3193178*Sqrt[1-d]*d^3))+g*(6+10*Sqrt[1-d]+(3+65*Sqrt[1-d])*d
-3*(1+50*Sqrt[1-d])*d^2+(1+100*Sqrt[1-d])*d^3-25*Sqrt[1-d]*d^4+4*a
^2*Sqrt[1-d]*(-4-29*d+66*d^2-44*d^3+11*d^4)))/4;

m12 = -((-1+g)^3*(-(2*g^10*(-1+d)^3+d^2*(-3+2*d)-2*g^9*(-1+d)
^2*(-7+9*d)+g^5*(71-421*d+714*d^2-364*d^3)+g^3*(3-133*d+314*d
^2-184*d^3)+g^7*(77-323*d+430*d^2-184*d^3)-g*(5+5*d-28*d^2+18*d^3)
+2*g^2*(4+19*d-60*d^2+37*d^3)+g^8*(-43+158*d-189*d^2+74*d^3)+g
^6*(-90+440*d-658*d^2+308*d^3)+g^4*(-34+286*d-560*d^2+308*d^3))
^3+2*a*Sqrt[1-a^2]*(-1+g)^20*(2*g^4*(-1+d)^3+(-1+d)^2*(1+2*d)-2*g
^3*(-1+d)^2*(-1+3*d)+g*(1-5*d+10*d^2-6*d^3)+g^2*(-1+8*d-15*d^2+8*d
^3))^3*Cos[phi]+I*a*(-1+g)^16*(-1+d)^3*Sqrt[(-1+a^2)*(-1+d)]*(2+4*
g^12*(-1+d)^4+7*d-9*d^2-12*d^3+12*d^4-8*g^11*(-1+d)^3*(-3+5*d)+4*g
^10*(-1+d)^2*(18-61*d+49*d^2)-2*g*(1+17*d-12*d^2-58*d^3+52*d^4)-4*
g^9*(34-213*d+474*d^2-449*d^3+154*d^4)+g^2*(-11+76*d+66*d^2-532*d
^3+428*d^4)+4*g^4*(-7-37*d+384*d^2-804*d^3+509*d^4)-2*g^3*(-13+29*
d+258*d^2-776*d^3+556*d^4)-2*g^7*(77-705*d+2130*d^2-2624*d^3+1136*
d^4)+g^8*(173-1290*d+3321*d^2-3572*d^3+1372*d^4)-2*g^5*(5-299*d
+1476*d^2-2484*d^3+1384*d^4)+g^6*(87-1120*d+4098*d^2-5840*d
^3+2864*d^4))*Sin[phi]))/2;
```