

UNIVERSITY OF TRIESTE

DEPARTMENT OF PHYSICS



Master's degree in Theoretical Physics

Entanglement and Robustness of the Quantum Fourier Transform in Shor's Algorithm

Supervisor:

prof.

Antonello Scardicchio

Candidate:

Giulia Tranquillini

Correlator:

prof.

Fabio Benatti

Mat. SM2300550

Academic Year 2022/2023

UNIVERSITÀ DEGLI STUDI DI TRIESTE

DIPARTIMENTO DI FISICA



Corso di Laurea Magistrale in Fisica Teorica

Entanglement e Robustezza della Quantum Fourier Transform nell'Algoritmo di Shor

Relatore:

prof.

Antonello Scardicchio

Candidato:

Giulia Tranquillini

Correlatore:

prof.

Fabio Benatti

Mat. SM2300550

A.Accademico 2022/2023

Abstract

In 1994 Peter W. Shor discovered an efficient way to divide a number N into a couple of prime factors p and q exploiting quantum mechanics. This single, apparently simple operation implies the use of a complicated set of gates and unitary operations when performed using quantum computation. This thesis aims to conduct a thorough analysis of the renowned algorithm, employing an innovative approach to examine its entanglement dynamics and assess its resilience against random errors that may arise during evolution. The presented arguments, the section about entanglement in particular, are built upon the works of Gianni Mossi [Mos16], inspired by an article of W. J. Munro [KM04]. The second part derives from the approach of Devitt and Fowler described in detail in [DFH06].

Chapter 1 and Chapter 2 are meant to be an introduction to the algorithm and the mathematical tools used in the thesis. As regards the entanglement between qubits, several witness quantities have been considered in Chapter 3 to have a clear understanding of it and its overall impact. The analyses presented in this work have been performed entirely using the aid of the Aer simulator in Qiskit (see Appendix A). This allowed computational and visual support to the thesis and helped clarify each pattern recognised with plots and representations.

Then, in Chapter 4 and Chapter 5, particular emphasis has been given to the Quantum Fourier Transform (QFT), for exploring how much the impact of a Noisy-QFT with random unitary gates applied along the circuit affects the factorisation. The aim is to investigate the resistance of the system to errors when they increase in number but remain localised in a single portion of the circuit and therefore verify the influence of such a QFT on the general Shor's efficiency.

Abstract

Questo lavoro di tesi si pone come obiettivo l'analisi del noto algoritmo di fattorizzazione di Peter W. Shor che, nel 1994, elaborò un metodo efficace di fattorizzare un qualunque numero N in una coppia di fattori primi p e q . Tale operazione, classicamente svolta in pochi passaggi, nell'ambito quantistico viene invece risolta da una complessa successione di gates definiti da operazioni unitarie.

Il primo obiettivo che ci si pone è l'analisi di pattern e proprietà di due fondamentali caratteristiche di un circuito quantistico: il suo entanglement e la sua risposta ad errori casuali ai quale viene sottoposto durante l'evoluzione temporale. Punto di partenza per questa serie di approfondimenti è stato il lavoro di Gianni Mossi [Mos16] , a sua volta ispirato da un articolo di W. J. Munro [KM04]. Nei primi capitoli verranno introdotti nel dettaglio l'algoritmo nonché una serie di grandezze utilizzate come strumenti di rilevazione della presenza o assenza di entanglement, che permetteranno di catalogarne in modo chiaro la dinamica e l'impatto ad ogni step della procedura. L'analisi è stata portata avanti tramite il simulatore Qiskit-Aer, uno strumento facente parte della libreria Qiskit di Python. Tale dispositivo ha permesso di avere un supporto grafico e computazionale, fornendo rappresentazioni visive a sostegno delle conclusioni raggiunte.

Nella seconda parte della tesi, è stata invece presa in considerazione in modo più approfondito la Quantum Fourier Transform (QFT), sottogruppo di gates fondamentale per numerosi algoritmi, per andare a verificare la robustezza dell'algoritmo sotto l'azione di una Noisy-QFT. Tale trasformata presenterà all'interno un numero crescente di rotazioni unitarie che in questo lavoro modelizzeranno degli errori randomici. Infine viene svolta un'analisi sull'efficacia dell'algoritmo di Shor e sulla sua capacità di fattorizzare in modo corretto quando sottoposto ai suddetti errori localizzati a una sua singola sotto-funzione.

La lotta contro la paura

Comincia dalla salatura della pasta

Prosegue poi con la cottura della stessa.

Per la signora Pina questa cucina è diventata un campo di battaglia

Chissà la sua famiglia come la prenderà se sbaglia

Chissà se il piccolo Renato

Di fronte a un piatto di spaghetti scotto e troppo saporito

Non si lamenterà

Ma dirà: "Mamma grazie perché

Mi insegni ad essere un uomo coraggioso"

G. Truppi

Contents

Abstract	II
Sommario	III
Quote	IV
1 Quantum Computation	1
1.1 Introduction	1
1.2 Postulates of quantum mechanics	2
1.3 Bits and qubits	4
1.4 Quantum gates	5
1.5 Quantum advantage	6
2 Shor's Algorithm	9
2.1 An historic introduction	9
2.2 The algorithm speed-up	10

2.3	The structure	13
2.3.1	Initialisation	14
2.3.2	Modular exponentiation	15
2.3.3	Inverse quantum Fourier transform	16
2.3.4	Measure and post-processing	17
2.4	The implemented circuit	18
3	Quantum Entanglement	21
3.1	Entanglement between registers	22
3.2	Entanglement within registers	24
3.3	Pairwise entanglement	25
3.4	Bipartite entanglement	26
3.5	Entanglement in the ideal IQFT	29
3.6	Single qubit entanglement	30
4	Error Analysis in the QFT	33
4.1	The quantum Fourier transform	33
4.2	Quantum errors	35
4.2.1	Bit flips and phase flips	37
4.2.2	Implementation of errors	37

4.3	Adding a single error to the QFT	38
4.4	Single error generalisation	40
4.5	Return probability	43
4.6	Single error results	45
4.7	Why the symmetry?	49
4.8	Multiple error results	50
4.9	Amplitude attenuation	56
5	Noisy-QFT for Factoring Numbers	59
5.1	Selecting the right witness quantity	59
5.2	Useful states and robustness	63
5.3	Considerations on the error model	68
5.4	Factoring 27	70
5.5	Factoring 63	74
5.6	Fitting the curves	75
Appendices		81
A	The Qiskit Library	82
A.1	What is Qiskit	82
A.2	Qiskit components	82

B	Further Plots	85
B.1	Case 2: single error before the swapping procedure	85
B.2	Case 3 - single error after the swapping procedure	87
B.3	Case 4 - one error before each Hadamard	89
B.4	Case 5 - one error after each Hadamard	91
B.5	Average behaviour	94
C	Notebooks Index	95
Bibliography		97

Chapter 1

Quantum Computation

This first chapter is a very brief introduction to quantum computing basics, going from the mathematical fundamentals of quantum mechanics to quantum gates, in order to make the subsequent notation clear. Then, the current state of the art is briefly described together with some information about the utility and power of quantum computers. For further and deeper information see [Yan07].

1.1 Introduction

The mathematical backbone of quantum physics is provided by the theory of linear operators on *Hilbert spaces* denoted by \mathcal{H} which are complex vector spaces equipped with a scalar inner product $\langle \cdot , \cdot \rangle$. Given two vectors x and y in \mathbb{C}^n , it is defined as:

$$\langle x, y \rangle = \sum_{i=1}^n x_i^* y_i = \begin{pmatrix} x_1^* & x_1^* & \dots & x_n^* \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad (1.1)$$

where the symbol $*$ denotes the complex conjugate. The inner product defines automatically a norm:

$$\|x\| = \sqrt{\langle x | x \rangle} \quad (1.2)$$

and therefore the metric (or distance) of the space:

$$d(x; y) = \|x - y\|. \quad (1.3)$$

Let \mathcal{H} and \mathcal{K} be finite-dimensional Hilbert spaces. Their algebraic tensor product plays a fundamental role in quantum mechanics, particularly in describing composite quantum systems and defining entanglement.

$$\sum_{i,j=1}^n x_i \otimes y_j \quad (x_i \in \mathcal{H}, y_j \in \mathcal{K}) \quad (1.4)$$

If e_1, e_2, \dots and f_1, f_2, \dots are bases in \mathcal{H} and \mathcal{K} respectively, then $\{e_i \otimes e_j : i, j\}$ is a base in the tensor product space. This shows that:

$$\dim(\mathcal{H} \otimes \mathcal{K}) = \dim(\mathcal{H}) \times \dim(\mathcal{K}). \quad (1.5)$$

Hilbert spaces can be both finite-dimensional as \mathbb{C}^n or infinite dimensional as $\mathcal{L}^2(\mathbb{R}^n)$.

1.2 Postulates of quantum mechanics

The mathematical description of quantum mechanics has at its heart the definitions of quantum states and quantum observables, which is done via the introduction of postulates:

Postulate 1. *The state of an isolated physical system is represented, at a fixed time t, by a state vector $|\psi\rangle$ belonging to a Hilbert space \mathcal{H} called the state space.*

A *pure* physical state of a system determines a corresponding state vector up to a phase. It is a projection onto normalised vector states and can be identified with a $|\cdot\rangle$. Traditionally in QM, states are distinguished between pure and *mixed states*, described by density matrices. A density matrix (or statistical operator) is defined as:

$$\rho = \sum_{j=1}^n \lambda_j |\psi_j\rangle \langle \psi_j|. \quad (1.6)$$

The three main properties of a density matrix are its normalisation, positivity and the fact that it is a hermitian operator:

1. $\text{Tr}(\rho) = \sum_{j=1}^n \lambda_j \langle \psi_j | \psi_j \rangle = \sum_{j=1}^n \lambda_j = 1$
2. $\langle \phi | \rho | \phi \rangle = \sum_{j=1}^n |\langle \phi | \psi_j \rangle|^2 \geq 0$
3. $\rho = \rho^\dagger$

The density matrix describes a statistical ensemble where states $|\psi_j\rangle$ are associated with weights $\lambda_j \geq 0$ such that $\sum_j \lambda_j = 1$. Weights are probabilities of finding the system S described by ρ in the states $|\psi_j\rangle$ if and only if the $|\psi_j\rangle$ are orthonormal and the λ_j are eigenvalues of ρ . In general $\rho = \rho^\dagger = \sum_j \lambda_j |\psi_j\rangle \langle \psi_j|$ can always be diagonalized as:

$$\rho = \sum_{k=1}^n r_k |r_k\rangle \langle r_k| \quad (1.7)$$

where $r_k \geq 0$, $\sum_k r_k = 1$ and $\langle r_k | r_l \rangle = \delta_{kl}$. Both statistical ensembles $\epsilon_\rho = \{|\psi_j\rangle, \lambda_j\}$ and $\epsilon_\rho^d = \{|r_k\rangle, r_k\}$ are described by the same density matrix.

Postulate 2. *Every measurable physical quantity A is described by a Hermitian operator A acting on \mathcal{H} . This operator is an observable, meaning that its eigenvectors form a basis for \mathcal{H} . The result of measuring a physical quantity A must be one of the eigenvalues of the corresponding observable.*

Given a discrete observable $A = A^\dagger = \sum_{\alpha=1}^n a_\alpha |\alpha\rangle \langle \alpha|$ of S and a state vector $|\psi\rangle \in \mathcal{H}$, the measure of A returns the eigenvalue a_α with $\mathcal{P} = |\langle a_\alpha | \psi \rangle|^2$ and leaves the state in the eigenstate $|a_\alpha\rangle$. The process is defined as:

$$\rho \rightarrow \sum_{\alpha} P_{\alpha} \rho P_{\alpha} \quad (1.8)$$

where $P_{\alpha} = |a_{\alpha}\rangle \langle a_{\alpha}|$ specifies a projector.

Postulate 3. *The state vector $|\psi(t)\rangle$ evolves following the Schrödinger equation where $H(t)$ is the Hamiltonian: the observable associated with the total energy of the system.*

The evolution can be mathematically written as:

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H(t) |\psi(t)\rangle . \quad (1.9)$$

The solution is a unitary matrix ($U^\dagger U = 1$) defined by $U = \exp(-iHt/\hbar)$. In this way:

$$|\psi(t)\rangle = U(t; t_0) |\psi(t_0)\rangle . \quad (1.10)$$

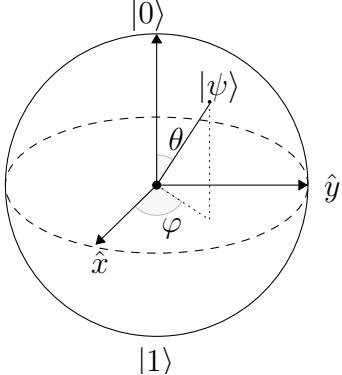
1.3 Bits and qubits

The fundamental unit of quantum computing is the *qubit*, a particular quantum system which only has two levels. In classical formalism, the usual way to represent information is the *bit*: an object which only takes values 0 and 1. Its quantum version is the two-level quantum mechanical system that can be exemplified as the spin of an electron or the polarisation of a photon. Differently from the classical bit, the qubit can live in the superposition of the two base states 0 and 1. In general, it can be written as:

$$|q\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \quad (1.11)$$

where $|\cdot\rangle$ is the standard Dirac notation and α_0 and α_1 are complex numbers subjected to the condition $|\alpha_0|^2 + |\alpha_1|^2 = 1$. In the standard notation, the *computational basis* is chosen to be $\{|0\rangle, |1\rangle\}$ with:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} . \quad (1.12)$$



The Bloch sphere is an alternative way to visualise the state of a single qubit. In the image, the North Pole of the sphere represents the $|0\rangle$ state, whereas the South Pole represents the $|1\rangle$ state. Every point between the two represents a superposition with relative phase φ and amplitude angle θ . Every state can be represented as

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle. \quad (1.13)$$

1.4 Quantum gates

Quantum computers can manipulate information using unitary transformations. Citing the article [JL03], a generic stage of a quantum computation is a n -qubit state (usually entangled) of the form:

$$|\alpha_0\rangle = \sum a_{i_1, \dots, i_n} |i_1\rangle \dots |i_n\rangle. \quad (1.14)$$

Now, suppose to apply a 1-qubit unitary U to the first qubit; the state then becomes

$$|\alpha_1\rangle = \sum a'_{i_1, \dots, i_n} |i_1\rangle \dots |i_n\rangle \quad (1.15)$$

where

$$a'_{i_1, \dots, i_n} = \sum_{j_1} U_{i_1 j_1} a_{j_1, \dots, i_n} \quad (1.16)$$

and $U_{i_1 j_1}$ are the matrix elements of U . Generalising this premises, it is possible to define a *quantum computational process* of size m (a fixed positive integer) as the sequence

$$\mathcal{A}_m = (U_{i_0}, \vec{a}_0), \dots (U_{i_m}, \vec{a}_m). \quad (1.17)$$

The U_{i_j} s are the unitary matrices performing the j^{th} operation, and the vector \vec{a}_i collects the coefficients of the qubits involved in that step. After the evolution, qubits are measured in the computational basis resulting in the probability distribution. The simpler possible unitaries are single-qubits operations as the Hadamard gate, a gate that transforms $|0\rangle$ in state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and state $|1\rangle$ in $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. Later on, it

will be clear how to use this gate to create a homogeneous superposition of all the base states. Another fundamental family of gates are the controlled operations. The CNOT for example, is a gate that flips the second qubit (the target) only if the first, the control, is a $|1\rangle$

$$U_{\text{CNOT}} = |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes \mathbb{X}. \quad (1.18)$$

The CNOT, together with the single qubit operation, compose a universal set of quantum gates. It can be generalised to a controlled- U gate where the transformation U only acts if the target qubit is $|1\rangle$:

$$c - U = |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes \mathbb{U}. \quad (1.19)$$

Two other fundamental unitary transformations in this study are the *swap gate*

$$U_{\text{SWAP}} |\psi_1, \psi_2\rangle = |\psi_2, \psi_1\rangle \quad (1.20)$$

and the controlled-phase

$$c - P = |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes \mathbb{P} \quad (1.21)$$

which induces a phase flip on the target qubit depending on the control state.

1.5 Quantum advantage

Quantum computation is generally regarded as being more powerful than the classical approach, and although limitations are still not entirely characterised, future implications are promising. There are two fundamental aspects of quantum mechanics leading to a significant speed-up: the compact representation, and the action of the previously described unitaries. First, to classically represent a superposition of 2^n levels, an exponentially large number of states is necessary and this number is also exponentially increasing with the number of qubits n . In contrast, given the role of entanglement, in quantum theory, a general 2^n state superposition can be described in n two-level systems, and the amount of physical resources only grows linearly with n . Along with that goes the second point, known as *quantum parallelism*. One can perform an operation of a state in superposition, and that will act on all the states of the superposition; basically in one operation on a quantum computer, it is possible to do what would take an exponential number of

operations on a classical computer.

Chapter 2

Shor's Algorithm

2.1 An historic introduction

In 1981 Richard Feynman proposed for the first time the idea of simulating quantum physics using quantum computers for problems that are too hard to simulate using conventional ones. This gave rise to a completely new field of study. In his famous conference [Fey82] Feynman pointed out the inefficiency of simulating quantum phenomena in the classical paradigm and the impossibility to make an exhaustive simulation of Nature because it is fundamentally non-classical. In the next decades, Feynman's ideas gradually grew and in 1985 Deutsch formalised the notion of *quantum computer* (see [Deu85]) conjecturing a possible advantage of this new type of technology over the old one at solving quantum problems as well as questions independent from the field of physics.

In those years Daniel Simon started his work showing that a quantum computer could achieve an exponential speed up in solving an idealised version of the problem of finding the period of a function. In the article '*On the Power of Quantum Computation*' [Sim94] Simon described by a clear example, the potentiality of this new technology: he thought of representing classical probabilistic computation on a Turing Machine (TM) as a levelled tree where each node corresponds to a state (or configuration), and each level to a step of the computation, reachable with non-zero probability. This tree, resembling a Markov process, conforms to the laws of information theory, so that the probability of a particular

path being followed from the root to a node, is simply the product of the probabilities along its edges. The probability that a particular configuration is reached at a certain step i of the computation is the sum of the probabilities of all the nodes corresponding to that configuration at level i in the tree. Also, the sum of the probabilities of all configurations at any level of the tree is always 1, regardless of the starting point.

The quantum equivalent of this TM-tree, a QTM, can be represented by a similar structure, but instead of probabilities now each edge has an associated *amplitude*. Such a quantum tree obeys the property of unitary sum at any level too, but the substantial difference is that the probability of a configuration is now the *square* of its amplitude. Therefore, the probability of a particular final configuration is the square of the sum of the amplitudes of all leaf nodes corresponding to it. The situation here is fundamentally different: amplitudes can be thought of as complex numbers with magnitude at most 1 (numbers in the interval $[-1, +1]$) so that it is possible to witness a final configuration with zero probability originated by two non-zero leaf nodes with amplitudes $+\alpha$ and $-\alpha$. Similarly, for two leaf nodes with $+\alpha$ amplitude, the final configuration occurs with $\mathcal{P} = 4 \cdot \alpha^2$, twice as frequently as the classical $2 \cdot \alpha^2$. Such a phenomenon is the reason why quantum computation in a theoretical sense is considered to be more powerful: different branches of the tree influence each other causing *interference*. Only unitary operations are acceptable: preceding configurations can always be determined given the current one in a reversible way. We, therefore, think about a quantum computation as resulting, at any step, in the superposition of all the branches of the tree simultaneously, keeping the entire tree in mind rather than assuming we can just follow a deterministic branch.

Simon's work was followed by Shor's, who in 1994 succeeded in finding an algorithm solving both discrete logarithms and prime factorisation of large numbers in polynomial time exploiting the quantum paradigm for finding the period of a function.

2.2 The algorithm speed-up

In number theory the problem of decomposing an integer number in two prime factors is known since Euclid: the fundamental theorem of arithmetic states that every positive integer has a unique prime factorisation. Testing whether an integer is prime can be done in polynomial time (the time required to execute the algorithm is bounded by a

polynomial function of the input size), but obtaining the factors when the numbers are sufficiently large is a task that no classical algorithm can do in an efficient way. The hardest instance of these problems is with semi-prime numbers: integers product of two primes only. To give an idea of the hardness of these problems, in 2019 a 230-digit number has been factored using approximately 900 core-years of computing power.

Problems belonging to the complexity class P can be solved in polynomial time, meaning that a number m of bits would be factored in time $O(m^c)$ for some constant c . It is believed that a classical solution for the period search in this class does not exist, hence the problem is not considered in class P but is suspected to be of class NP . This second complexity class includes problems which cannot be solved in polynomial time but can be verified in polynomial time, once the solution is given. The fastest classical algorithm for factorisation is the *general number field sieve* (GNFS), decomposing an integer n in sub-exponential time (here and in the following $\log = \log_2$):

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\log n)^{\frac{1}{3}}(\log \log n)^{\frac{2}{3}}\right). \quad (2.1)$$

Shor's algorithm on a quantum computer runs in polylogarithmic time, so polynomial in $\log n$ which is the size of the input. Namely, using fast multiplication:

$$(\log n)^2(\log \log n)(\log \log \log n). \quad (2.2)$$

The complexity class associated with this behaviour is the bounded-error quantum polynomial time (BQP), defined by Bernstein and Vazirani ([Li11]) in a simplified version as follows:

Definition 1 Let's define a language L as a set of encoded-words with letters taken from an alphabet of 0s and 1s. A problem belongs to BQP if and only if there exist a family of quantum circuits built with a QTM $\{Q_n : n \in \mathbb{N}\}$ (with n being the size of Q_n) such that:

- $\forall n \in \mathbb{N}$, Q_n takes n qubits as input and 1 bit as output;

BQP problems are in this context referred to as decision problems; the output can be thought of as composed of two non intersecting classes (yes and no, or in our case fac-

torisable and non-factorisable). Then:

- $\forall x \text{ in } L, \text{Prob}(Q_{|x|}(x) = \text{yes}) \geq 2/3;$
- $\forall x \text{ not in } L, \text{Prob}(Q_{|x|}(x) = \text{no}) < 1/3.$

Rephrasing, a language L belongs to BQP if and only if there exists a polynomial quantum Turing machine that accepts L with an error probability of at most $1/3$ for all single instances. The value of $2/3$ is completely arbitrary as long as it is bigger than $1/2$ and independent from n . Performing the algorithm 100 times and exploiting the technique of *amplitude amplification* will lead to an error of order $(1/3)^{100}$. From a practical point of view, the quantum algorithm has a surprising efficiency but it is not free from all issues when dealing with bigger numbers. In 2001 IBM factored number 15 implementing the algorithm on 7 qubits, then again in 2012 exploiting multi-qubits entanglement of solid-state qubits. Number 21 was factored using photonic qubits followed by number 35 on the 20-qubits IBM Q System One, but the procedure failed because of error accumulation. Despite the limitations for big numbers are still a huge restrictive constraint, Shor's algorithm has been undoubtedly revolutionary and it is a possible threat to the protective cryptography protocols currently used in communication. For this reason, the interest in better understanding its features and limits is extremely high.

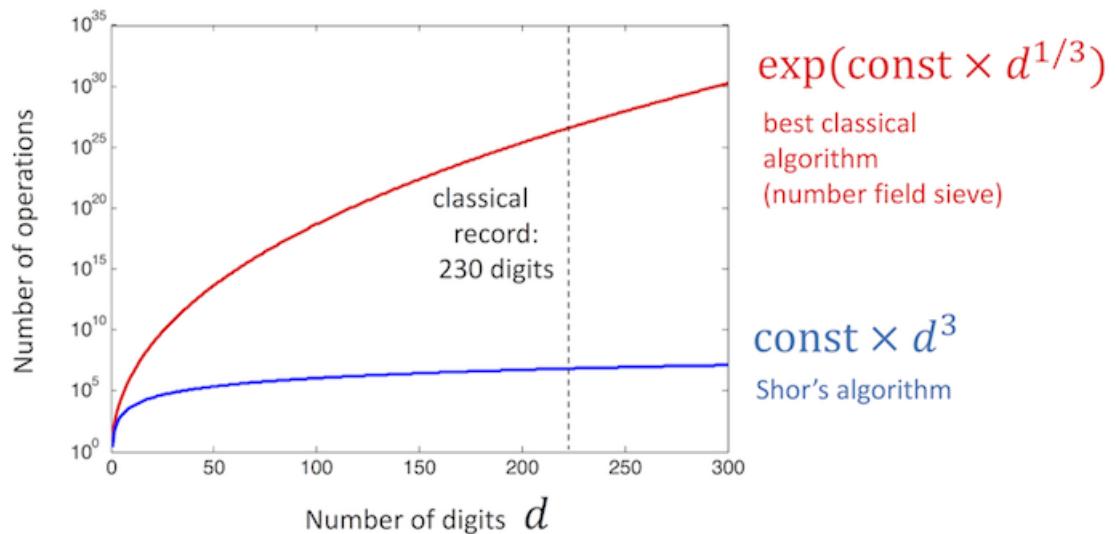


Figure 2.1: Shor's speed-up

2.3 The structure

It has been known to mathematicians since the 1970s that factoring becomes easy if one can solve the problem of finding the period of the *modular exponential function*. Such a period search is defined as follows: given an integer N and an integer y , the goal is to find the smallest positive integer $r > 0$ such that:

$$y^r \equiv 1 \pmod{N}. \quad (2.3)$$

In the original article by Shor [Sho94] the procedure is described in detail: in order to find the two factors of N given a method for computing the order r of $y \pmod{N}$, it is necessary to compute the $\gcd(y^{r/2} \pm 1, N)$, where the $\gcd(A, B)$ is the greatest common divisor of A and B . An efficient way of doing this is described by Euclid's theorem, which solves the problem in polynomial time. Re-writing $(y^{r/2} - 1)(y^{r/2} + 1)$ as $y^r - 1 \equiv 0 \pmod{N}$, $\gcd(y^{r/2} - 1, N)$ fails to be a non-trivial divisor of N only if:

1. r is odd,
2. $y^{r/2} \equiv -1 \pmod{N}$.

If the above conditions are met r is well-defined, and the two factors can be found. The two numbers N and y are chosen as follows: N must be an odd number, otherwise 2 would be a trivial divisor, and must be non-prime nor a prime power, which means $N \neq p^k$ for prime p and $k > 1$. It is also necessary that $\gcd(N, y) = a = 1$, that is to say, that they are co-prime numbers. If $a > 1$, y and N are not *co-prime*, their common divisor already gives a factor of N and the algorithm is not useful, but this a rare situation. A visual representation is given in Section 2.3 for $y = 7$, $N = 15$ and $f(x; r) = y^x \pmod{N}$. The rapidity in finding the right couple of factors p and q strongly depends on the choice of the co-prime. In his work, Shor states that in general, the success probability of the procedure for semiprimes is $\mathcal{P} = 0.5$.

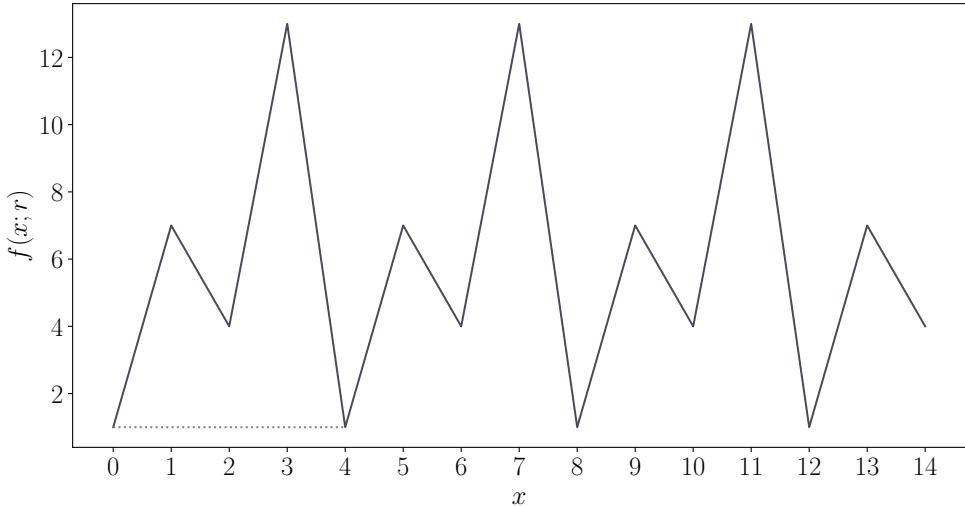


Figure 2.2: The period of this particular $f(x; r)$ is $r = 4$

2.3.1 Initialisation

In its general form, the quantum circuit implementing the algorithm is made of two quantum registers. The first is a control register with $2n$ -qubits ($n = \log N$) used to store the state values from 0 to N and proportional to the precision of the phase estimation procedure. The second one is a target register for the function evaluations and is a classical register of dimension n also necessary for the measurement procedure. As said, the goal is to find the period r and this is pursued with high probability using classical number theory in the post-processing. The starting point, given a number of qubits n , is to initialise the $3n$ system to the state:

$$\psi_0 = |0\rangle_{2n} |1\rangle_n. \quad (2.4)$$

Then, a Hadamard transform on the first register $U_H = H^{\otimes 2n}$ is performed to create a complete superposition of all the states from 0 to $2^{2n} - 1$. The notation imposes that the state $|l\rangle$ has a binary representation resembling the position of 0s and 1s in the control register. The system's state reached after the Hadamard transform is:

$$|\psi_1\rangle = \frac{1}{2^n} \sum_{l=0}^{2^{2n}-1} |l\rangle_{2n} |1\rangle_n. \quad (2.5)$$

2.3.2 Modular exponentiation

The next step is to apply the function $f(l) = y^l \bmod N$ on the smaller register:

$$|\psi_2\rangle = \frac{1}{2^n} \sum_{l=0}^{2^{2n}-1} |l\rangle_{2n} |y^l \bmod N\rangle_n. \quad (2.6)$$

Many suggested papers do implement the modular exponentiation in a very complex way, involving phase estimators, adders, and multiplication operators as does [Bea02]. Wishing to perform the procedure on a variety of N numbers, it would have been computationally too heavy to replicate that setup. To overcome this problem, the unitary matrix for the exponentiation procedure has been implemented directly, calculating its elements according to the needed operation. Being $f : \{0, 1, \dots, N-1\} \rightarrow \{0, 1, \dots, N-1\}$ a bijective function and $\{|x\rangle\}_{x=0}^{N-1}$ an orthonormal basis of a Hilbert space \mathcal{H} , a unitary linear map is defined as

$$U_f : \mathcal{H} \rightarrow \mathcal{H} : U|x\rangle = |f(x)\rangle. \quad (2.7)$$

One can generalise the above definition for any couple y and N , so that given an integer $N' > N$, $f(x)$ is bijective on the set $\{0, 1, \dots, N'-1\}$:

$$f(x) = \begin{cases} xy \bmod N & \text{for } 0 \leq x \leq N-1 \\ x & \text{otherwise} \end{cases} \quad (2.8)$$

Starting with a $2^n \times 2^n$ null matrix and choosing the space base to be the canonical one where the state $|m\rangle$ is represented with a vector with all 0 except a 1 in the m^{th} + 1 position, for each basis state the $xy \bmod N$ is calculated and the transformation matrix is constructed with the resulting state-vectors. Then, given the unitary matrix

$$U_f : \mathcal{H} \rightarrow \mathcal{H}$$

, a controlled- U_f is defined on the Hilbert space $\mathbb{C}^2 \otimes \mathcal{H}$ as seen in the previous chapter:

$$c - U = |0\rangle\langle 0| \otimes \mathbf{1} + |1\rangle\langle 1| \otimes U_f. \quad (2.9)$$

This single operation is iterated for each k step as a controlled unitary gate, taking each qubit $l = 0, 1$ of the control register as control and making it act on the target register:

$$c - U_f^{2^k} |l\rangle |x\rangle = |l\rangle \left| y^{l^{2^k}} \bmod N \right\rangle. \quad (2.10)$$

The proof that U_k is a unitary is connected to the fact that the function $f(x)$ is bijective, and so has a well-defined inverse. Both $x, y \in [1, \dots, N-1]$, so it is enough to show that $f(x)$ is injective, so that if $x \neq x'$, then $yx \bmod N \neq yx' \bmod N$. Let's assume against our thesis that $yx \bmod N \equiv yx' \bmod N$ for $xx' \bmod N$. Since $\gcd(a, y) = 1$, then y must have an inverse $y^{-1} \bmod N$ so:

$$yx \bmod N \equiv yx' \bmod N \quad (2.11)$$

$$y^{-1}yx \bmod N \equiv y^{-1}yx' \bmod N \quad (2.12)$$

$$x \bmod N \equiv x' \bmod N. \quad (2.13)$$

This last statement contradicts our assumption, so we can say that $f(x)$ is a bijection.

2.3.3 Inverse quantum Fourier transform

The next step is to perform an inverse quantum Fourier transform (IQFT), the quantum analogue of the discrete classical Fourier transform (DFT) on the first register, which leaves the system in the state:

$$|\psi_3\rangle = \frac{1}{2^{2n}} \sum_{l=0}^{2^{2n}-1} \sum_{z=0}^{2^{2n}-1} e^{-\frac{2\pi i l z}{2^{2n}}} |z\rangle |y^l \bmod N\rangle. \quad (2.14)$$

The IQFT is a crucial step of the algorithm, it modifies each amplitude through a linear mapping and prepares them for the observation. The difference between the QFT and the classical one is that the QFT acts on a quantum state vector (or a quantum register), while the DFT works on a vector. The former one consists of $O(n^2)$ Hadamards and Phase gates while the latter uses $O(n 2^n)$ gates which is exponentially more.

2.3.4 Measure and post-processing

At last, the machine is observed: the state $|c\rangle$ is obtained by measuring the larger register only in the computational basis.

$$|\psi_4\rangle = |c\rangle. \quad (2.15)$$

The probability to find the $|c\rangle$ state is equal to

$$\mathcal{P}(c, r, n) = \left| \frac{1}{2^{2n}} \sum_{l:y^l=y^k \bmod N} e^{-\frac{2\pi i l z}{2^{2n}}} \right|^2. \quad (2.16)$$

Number c , divided by 2^{2n} , is a good approximation for the fraction k/r with $0 \leq k < r$. Both k and r can be obtained by continued fractions if $c \neq 0$. This final part of the algorithm is performed classically. The continued fraction representation is obtained by an iterative process of representing a number as the sum of its integer part plus the reciprocal with an analogue form. In this way:

$$c = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \dots}} \quad (2.17)$$

The integers a_i are called the *coefficients* or *terms* of the continued fraction. The state can be written equivalently as:

$$c = [a_0; a_1, a_2, \dots]. \quad (2.18)$$

Once the period r is found, this last calculation gives the expected factors:

$$q = \gcd(y^{r/2} - 1, N) \quad p = \gcd(y^{r/2} + 1, N). \quad (2.19)$$

If r is not prime, a possible error source is obtaining $\frac{r}{p}$ instead of r if k shares a factor p with r .

2.4 The implemented circuit

The described algorithm has been implemented in its circuital form on Python with the aid of the dedicated library Qiskit. The code collects as inputs just:

1. the number N to be factor (from which n is evaluated);
2. the co-prime integer y ;

and checks via an apposite function if they are adequate. The modular exponentiation code is here reported for clarity. The matrix performing the transformation is found cycling on all the basis states and their corresponding values are found. In the last line, the matrix is transformed into a controlled gate:

```
def mod_exp(n, y, N, power):
    dim = 2**n
    matrix = np.zeros((dim, dim))
    #MATRIX ELEMENTS
    for i in range(N):
        j = (i*y)%N
        matrix[j][i]=1
    for ii in range(N, dim):
        matrix[ii][ii] = 1
    #EXPONENTIATION
    matrix = linalg.matrix_power(matrix, 2**power)
    #MATRIX TO GATE
    U = UnitaryGate(matrix)
    U.name = "%i^%i mod %i" % (y, 2**power, N)
    c_U = U.control()
    return c_U
```

The structure of the implemented circuit is a series of gates resulting in the structure in Figure 2.3. As an example, if $N = 15$ is the number to factorise and the co-prime integer chosen is $y = 2$. The circuit is implemented with size $n = \lceil \log 15 \rceil = 4$. The Aer

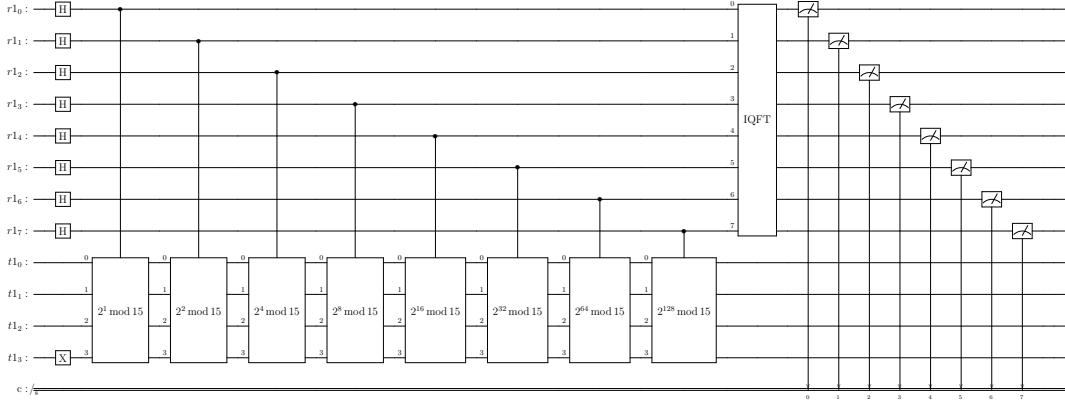


Figure 2.3: The circuit implemented on Qiskit for this type of procedure, the parameters are $N = 15$ which requires $n = 4$ qubits, and $y = 2$

Simulator performs a 1024 shots simulation of it, consisting in measuring the outcome of the procedure more than a thousand times to obtain the statistics of the simulation. The result is shown in Figure 2.4: the four states occur with similar probabilities. The

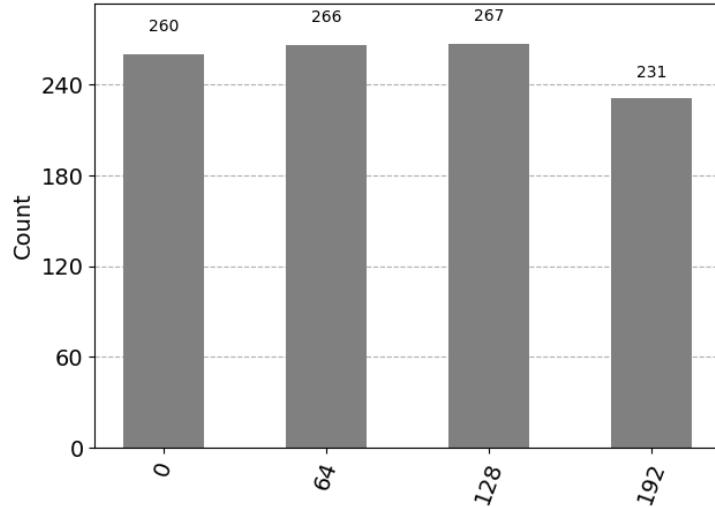


Figure 2.4: The simulative result of the circuit

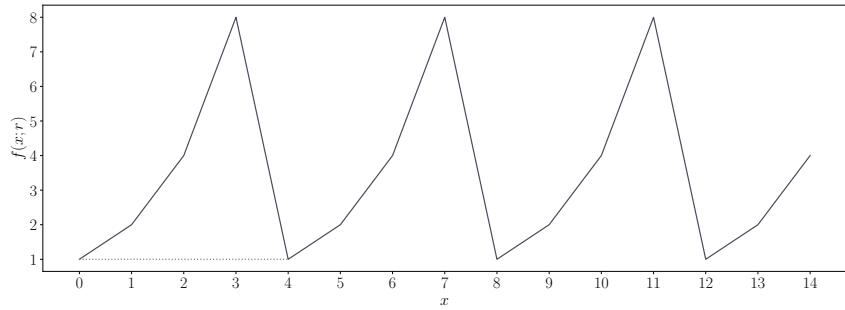
measured states $|c\rangle$ are transformed from binary representation to decimal representation and then divided by 2^{2n} in the form of a phase that approximates k/r :

	Register Output	Phase
0	10000000(bin) = 128(dec)	128/256 = 0.50
1	01000000(bin) = 64(dec)	64/256 = 0.25
2	11000000(bin) = 192(dec)	192/256 = 0.75
3	00000000(bin) = 0(dec)	0/256 = 0.00

Each phase is rewritten via continued fractions and the biggest of the possible periods is chosen as the period r :

	Phase	Fraction	Period
0	0.5	1/2	2
1	0.25	1/4	4
2	0.75	3/4	4
3	0.00	0/1	1

With the current parameters, number 4 is the correct period for $f(x) = 2^x \bmod 15$:



The last classical manipulation leads to the factors p and q that correctly factor number fifteen:

$$\gcd(2^2 - 1, 15) = 3 \quad \gcd(2^2 + 1, 15) = 5 \quad (2.20)$$

Chapter 3

Quantum Entanglement

One of the most interesting features of quantum algorithms and quantum mechanics in general is *entanglement*: a property of quantum systems that has no classical counterpart. The entanglement is the impossibility of particles at subatomic scales to be described independently one from the other, also when separated by large distances. This kind of phenomenon dictates the non-local nature of quantum mechanics. An interesting metaphor to clarify this concept is done by John Preskill (see [Pre12]). According to him, a quantum system can be thought of as a 100 pages book; if its nature was classical, reading 10 pages of the book would mean learning about the 10% of it, but for a quantum one, if we read 10 pages we learn almost nothing about the content of the book: the information is not printed on individual pages, but rather encoded in the correlations between them.

A typical and simple example of such a state is a Bell state, which is actually a maximally entangled one:

$$|\psi_B\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \quad (3.1)$$

For a state of this kind there exist no $|a\rangle$ and $|b\rangle$ such that $|\psi_B\rangle = |a\rangle \otimes |b\rangle$. From this follows the mathematical definition of entangled state written in the density matrix formalism:

Postulate 4. Define $S = S_1 + S_2$ to be a bipartite finite-level system and \mathcal{S} the set of its possible states. Then, the state $\rho \in \mathcal{S}(S_1 + S_2)$ is called entangled if and only if it cannot

be written as the tensor product of the density matrices of the sub-systems:

$$\rho \neq \sum_{i,j} \lambda_{ij} \rho_1^i \otimes \rho_2^j, \quad (3.2)$$

given that $\rho_1^i \in \mathcal{S}(S_1)$ and $\rho_2^j \in \mathcal{S}(S_2)$ with $\lambda_{ij} \geq 0$ and $\sum_{i,j=1}^n \lambda_{ij} = 1$.

Entanglement is strictly linked with the concept of locality: only the action of non-local operations can modify the entanglement pattern of the system, while local ones can both leave it constant (Hadamards) or destroy it (measures). For the purpose of this work, it is enough to think about non-local interactions as two-qubit gates where a joint unitary operation is carried out with the effect of coupling the qubits involved. Performing a classical simulation on a quantum system gives access to the state of the system after each unitary, making it possible to study where and how much entanglement is generated. The purpose of this chapter is to analyse the dynamics of entanglement in all its possible forms in the order-finding algorithm, trying to find patterns linked with the circuit structure and motivating them. Being that the entanglement is responsible for the quantum speed-up of the factorisation, the aim is to discover which of the circuit's components more influences the algorithm efficiency.

3.1 Entanglement between registers

As discussed in the previous chapter, the circuit implementing Shor's algorithm consists of two registers: a control and a target one; this instantly leads to the concept of bipartite system and to consider the $3n$ circuit construction as an object composed of a subsystem A (the $2n$ register) and a second system B (the target register). The analysis of entanglement in the circuit starts by studying the *entropy of entanglement* between A and B , defined as the Von Neumann entropy of either of its reduced states. If the state of the total system is a separable state:

$$|\Psi_{AB}\rangle = |\phi_A\rangle \otimes |\phi_B\rangle, \quad (3.3)$$

then the reduced density matrix also represents a pure state:

$$\rho_A = \text{Tr } \rho_{AB} = \text{Tr } |\Psi_{AB}\rangle \langle \Psi_{AB}| = |\phi_A\rangle \langle \phi_A|. \quad (3.4)$$

Given that a pure state has zero entropy, a non-zero entropy resulting from a reduced density matrix indicates the presence of entanglement between the sub-systems. If the reduced density matrices of the registers are ρ_A and ρ_B , the Von Neumann entropy is defined as the Shannon entropy of the spectrum of one of them generically called ρ :

$$S(\rho) = -\text{Tr} \rho \log(\rho) = -\sum_i r_i \log(r_i) \quad (3.5)$$

where $\rho = \sum_i r_i |r_i\rangle \langle r_i|$ is the spectral representation of ρ with $r_i \geq 0$, $\sum_i r_i = 1$ and $\langle r_i | r_j \rangle = \delta_{ij}$. S has the property of being independent on which of the two registers is traced away:

$$S(\rho_A) = -\text{Tr}[\rho_A \log \rho_A] = -\text{Tr}[\rho_B \log \rho_B] = S(\rho_B). \quad (3.6)$$

This becomes clear when the state $|\psi_{AB}\rangle$ is written through its Schmidt decomposition using the orthonormal bases $|u_i\rangle_A$ and $|v_i\rangle_B$ of the two subsystems and $m = \min\{\dim(\mathcal{H}_1), \dim(\mathcal{H}_2)\}$:

$$|\Psi\rangle = \sum_{i=1}^m \alpha_i |u_i\rangle_A \otimes |v_i\rangle_B. \quad (3.7)$$

Given that, if one traces over subsystem A or B the result in both cases is:

$$S = -\sum_i |\alpha_i|^2 \log(|\alpha_i|^2). \quad (3.8)$$

The Qiskit implementation of the algorithm provides the user with a value of $S(\rho_A)$ at each step, resulting in the following pattern:

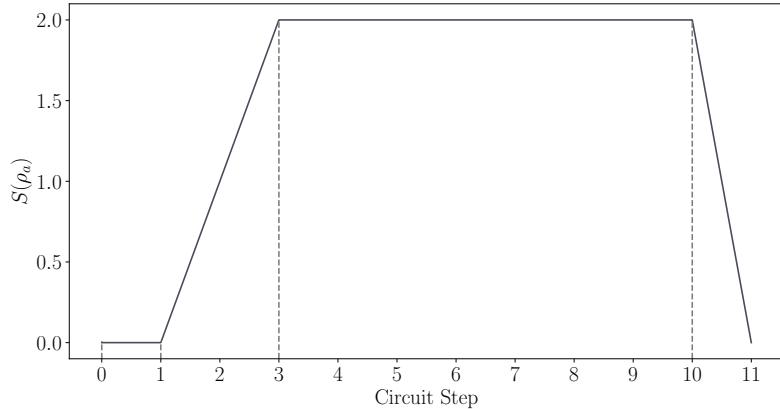


Figure 3.1: The entanglement entropy in the Shor circuit factorising $N = 15$ with co-prime $y = 2$ and $n = 4$ qubits. The target register is the one traced away in the simulation.

The result is linked with the circuit structure as follows:

- Step 0 – 1: pure state initialisation with Hadamards. The full state is pure and the entropy is therefore zero;
- Step 1 – 9: modular exponentiation. The $2n$ controlled unitaries cause the entanglement to build up to a saturation value of 2.0 (see 3.16);
- Step 9–10: IQFT. During the inverse transform the entanglement remains constant because no gates alter the relations *between* the registers;
- Step 10 – 11: measure. The measurement procedure collapses the system into a product state and cancels all the entanglement.

3.2 Entanglement within registers

What happens between registers is not the only source of entanglement present: the focus is now on the quantity’s dynamics *inside* each of the two registers. The states are now mixed and the Von Neumann entropy is no longer the best entanglement witness to consider. The choice falls back on the *negativity* defined as the trace norm $|\cdot|_1$ of the

transposed matrix ρ^T :

$$\mathcal{N}(\rho) = \frac{|\rho^T|_1 - 1}{2}. \quad (3.9)$$

To detect all the entanglement, a register is isolated and its qubits are divided into all the possible subsets to find the mean value on all the possibilities. If \mathcal{N} is not zero for all of them, then entanglement is present. The properties that make negativity a good detection tool are:

1. it is a convex function:

$$\mathcal{N}\left(\sum_i p_i \rho_i\right) \leq \sum_i p_i \mathcal{N}(\rho_i); \quad (3.10)$$

2. it is an entanglement monotone quantity:

$$\mathcal{N}(P(\rho)) \leq \mathcal{N}(\rho) \quad (3.11)$$

where P is a LOCC (local operation and classical communication) over ρ ;

3. the logarithmic negativity is an upper bound to the distillable entanglement and it is defined as:

$$E_{\mathcal{N}}(\rho) \equiv \log_2 |\rho^T|_1 = \log_2 (2\mathcal{N}(\rho) + 1). \quad (3.12)$$

The simulation performed in [KM04] shows that negativity is zero for both registers (pointed line) throughout the algorithm, with the exception that the measurement procedure, indicated by the label t_m in Figure 3.2, leaves the smaller register entangled although this is not useful at all for the remaining classical steps.

3.3 Pairwise entanglement

As demonstrated in the analysis of [KM04], the *entanglement of formation* can be used as a witness to calculate the entanglement between couples of qubits in a mixed state. This is done via the *concurrence*, an entanglement monotone quantity defined by the operator

$$\rho \tilde{\rho} = \rho \sigma_y^A \otimes \sigma_y^B \rho^* \sigma_y^A \otimes \sigma_y^B. \quad (3.13)$$

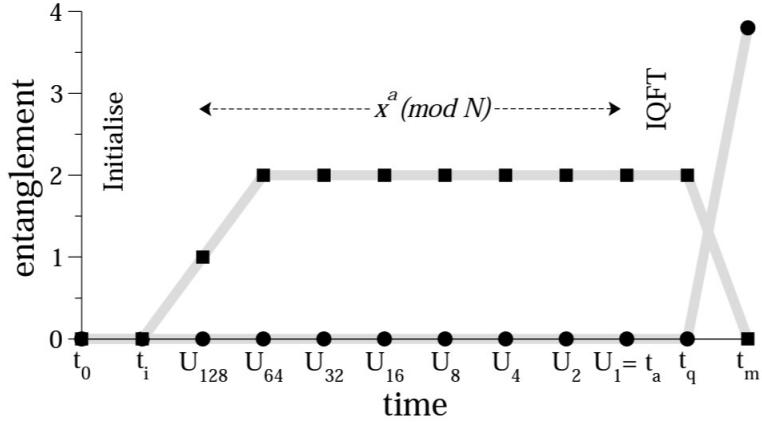


Figure 3.2: Entanglement within registers (squares) and entanglement within the smaller register (circles).

Here, ρ is the density matrix of the system and σ_y is a Pauli matrix. Defining as λ_i the non-negative square root of the eigenvalues of $\rho\tilde{\rho}$, the concurrence is computed as

$$C = \max(\lambda_1 - \lambda_2 - \lambda_3 - \lambda_4, 0). \quad (3.14)$$

The entanglement of formation is then computed as

$$E_f = h \left(\frac{1}{2} + \frac{1}{2} \sqrt{1 - C^2} \right) \quad (3.15)$$

where $h(p)$ is the binary entropy $h(p) = -p \log(p) - (1-p) \log(1-p)$. [KM04] states that there is no pairwise entanglement between any of the points neither between nor within the registers.

3.4 Bipartite entanglement

To spot the remaining entanglement, the negativity is analysed again dividing the *entire structure* into all possible bipartite subsystems, not only a single register at a time as before. The strategy is to divide the circuit in m and $3n - m$ qubits with $m = 1, \dots, 3n/2$. For each repetition and each size value m , qubits are randomly chosen and the results

are averaged to obtain the value at each step. For a circuit factoring number $N = 21$ constituted of $n = 5$ qubits, the entanglement builds up from step 1 to step 10 representing the modular exponentiation procedure, and increases slightly during the IQFT between step 10 and 11; the measuring procedure is not reported.

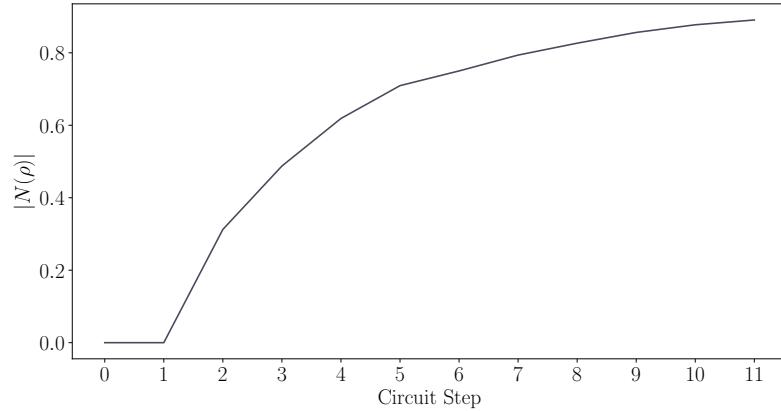


Figure 3.3: The absolute value of the average negativity as a function of the computational step, for $N = 21$, $y = 4$, 20 repetitions with size 4.

The larger the number of the considered qubits, the more the mean negativity curve resembles a logarithmic behaviour; As first shown by Nielsen and Chuang the upper

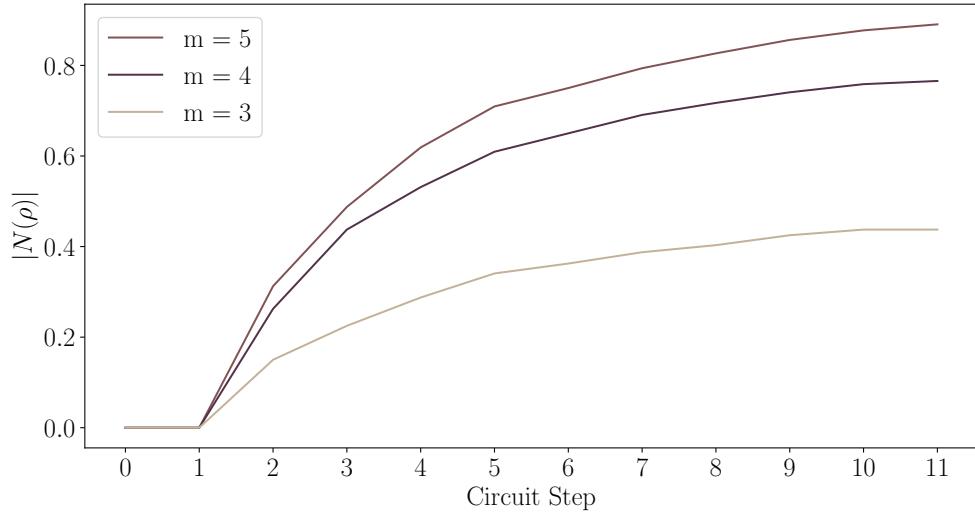


Figure 3.4: The absolute value of the average negativity for $N = 21$, $Y = 4$, 20 random partition choices

bound of the entanglement entropy can be approximated via:

$$S_r \leq \log(r) \quad (3.16)$$

where r is the period; with these parameters the value is $\log(r = 6) \simeq 0.8$

3.5 Entanglement in the ideal IQFT

Since the quantum Fourier transform is a fundamental step of the circuit with a high number of entangling gates inside, the analysis of the entanglement in the corresponding circuit section has been deepened. As said in previous chapters, unitary operations can only alter the entanglement between the gates they are applied to. The entanglement within each register is zero after the modular exponentiation, and during the IQFT, no gates operate between the two registers, where the only entanglement is present. So during the IQFT, the entanglement cannot decrease. Furthermore, inside the Fourier transform, each pair of qubits in the upper register undergoes a possibly entangling gate (the phase gate) only once, so entanglement can only be generated or shifted between qubits of that register, but it cannot decrease. It is possible to demonstrate another particular feature of the quantum Fourier transform, which is pointed out in [KM04]; during a $2n = 8$ qubits IQFT, entanglement migrates from the upper two qubits in the larger register to the lower couple, while the smaller register stays entangled the whole time. To verify this in the implemented circuit, only the entanglement of subsets of dimension two has been computed with the rest of the larger register. Simulating the

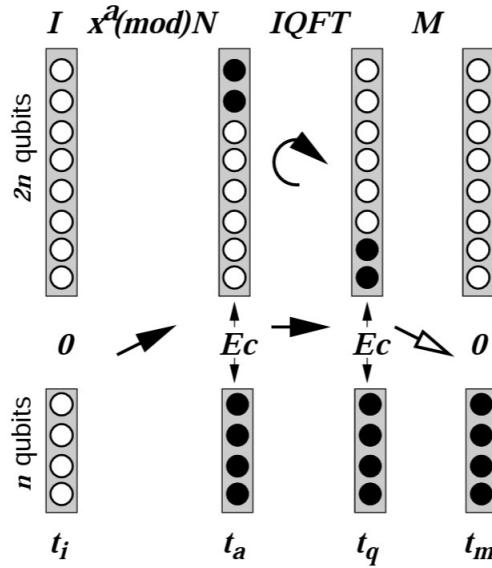


Figure 3.5: Entanglement pattern factoring 15 with co-prime $y = 13$

circuit and tracking the behaviour of the two couples under analysis only in the one step before and one after the transform the shift is clearly present, as seen in Figure 3.6.

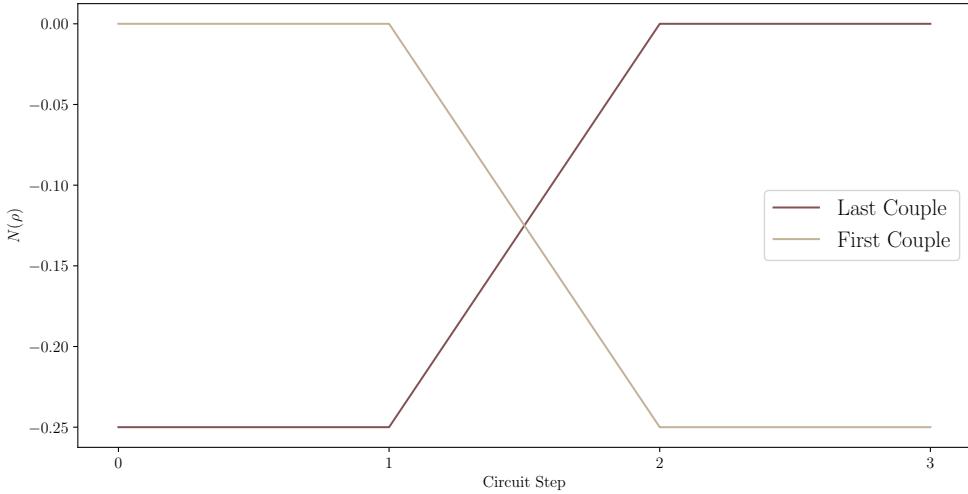


Figure 3.6: Negativity before and after the IQFT

3.6 Single qubit entanglement

In conclusion, the attention is shifted to the entanglement of a single qubit. A reduced density matrix is extracted from the simulation starting from $|0\cdots 0\rangle$ for each one of them and this time the chosen entanglement witness is again the Von Neumann entropy:

$$S_{\text{single qubit}}(\rho) = S(\rho^{(1)}) = -\text{Tr}[\rho^{(1)} \log \rho^{(1)}]. \quad (3.17)$$

The entanglement intensity is almost zero for all eight qubits of the register, so the behaviour in the plot is not extremely meaningful, however, an interesting piece of information is deducible from the image in Figure 3.7. Each qubit entangles in succession from the most significant (number 7) to the least significant (number 0). The fluctuation of their entanglement happens in concurrence with the first phase shift gate they are subject to, the first non-local gate acting on them (qubit 7 has a spike at step 5 probably due to noise only). Qubit 0 does not entangle because no P_θ is ever applied on it being the target.

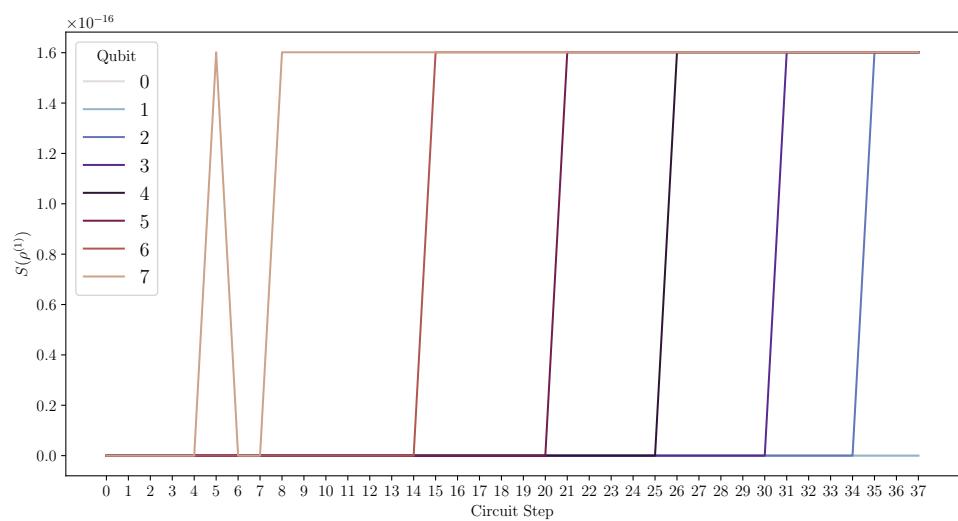


Figure 3.7: Entanglement of single qubits in the inverse quantum Fourier transform with the larger register of dimension $2 \cdot n = 8$, the same used for factoring $N = 15$

Chapter 4

Error Analysis in the QFT

4.1 The quantum Fourier transform

As a fundamental part of the factorisation algorithm, particular attention has been devoted in the previous chapter to the entanglement in the Quantum Fourier Transform. As the authors of [HV08] point out, the Shor algorithm (SA) is critically dependent on the QFT, which is highly sensitive to errors in the input states. The work of Hill and Viamontes shows that the polynomial scaling of the algorithm is destroyed by errors in the QFT, and demonstrates that error-correcting techniques are not always efficient in preventing this to happen. In this work, error mitigation codes are not involved. The point is rather to analyse the stability of the transform despite the insertion of unitary random rotations and the impact on the procedure's effectiveness. Obviously, the considered error types and schemes are dissimilar from the ones occurring in real quantum devices. Errors in quantum computers are not only of two types nor discrete: decoherence and gate noise result far more complicated than gates acting on single qubits in random positions. This work, however, wants to stick to the simplest case and work with only single-qubit inaccuracies in order to provide an overview of how much the simplest errors still have a huge impact on the probability of success of the period finding subroutine (QPF). First of all, the structure of the QFT is analysed in detail. This gate arrangement is constituted by a series of Hadamard and phase shifts as in Figure 4.1. The unitary matrix representation of the QFT in the computational basis shows its action on the

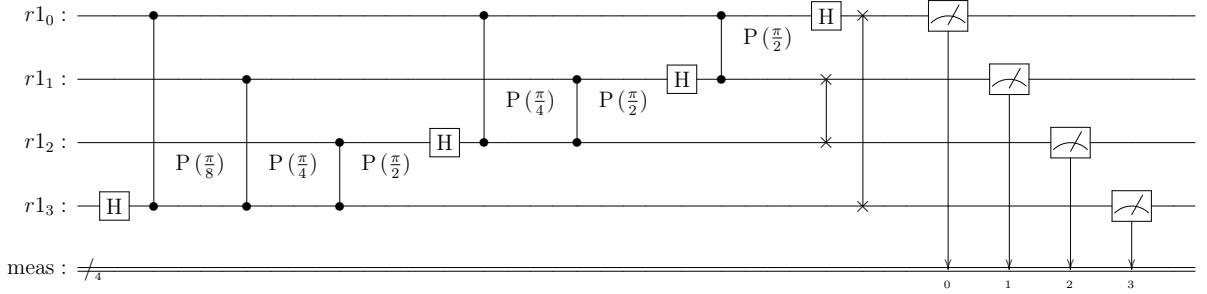


Figure 4.1: The circuit implemented on Qiskit for the QFT

amplitudes of the quantum state.

$$F_n = \frac{1}{\sqrt{2^n}} \sum_{q=0}^{2^n-1} \sum_{q'=0}^{2^n-1} e^{\frac{2i\pi q q'}{2^n}} |q\rangle \langle q'| \quad (4.1)$$

In Equation (4.1), $q = q_1 q_2 \dots q_n$ with $q_i \in \{0, 1\}$ is the decimal representation of the bits q_1, q_2 etc. and the same happens for q' . In a more convenient linear map representation, the action of F_n on a product state $|q_1 q_2 \dots q_n\rangle$ is usually defined as:

$$F_n |q_1 q_2 \dots q_n\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2i\pi 0.q_n} |1\rangle) \otimes (|0\rangle + e^{2i\pi 0.q_{n-1}q_n} |1\rangle) \otimes \dots |0\rangle + e^{2i\pi 0.q_1q_2\dots q_n} |1\rangle) \quad (4.2)$$

Here the binary fractional representation is used and $0.q_1 q_2 \dots q_n$ defines $\frac{q_1}{2} + \frac{q_2}{2^2} + \dots + \frac{q_n}{2^n}$. A natural way to construct the quantum circuit of the QFT is by decomposing the transform in:

$$F_n = R_n Q_n. \quad (4.3)$$

The operation denoted as R_n is the *bit reversal operator* acting as follows

$$R_n |q_1 q_2 \dots q_n\rangle = |q_n q_{n-1} \dots q_1\rangle. \quad (4.4)$$

The second operation Q_n is the core of the Quantum Fourier Transform and acts on the state in the following way:

$$Q_n |q_1 q_2 \dots q_n\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2i\pi 0.q_1 q_2 \dots q_n} |1\rangle) \otimes (|0\rangle + e^{2i\pi 0.q_2 \dots q_n} |1\rangle) \otimes \dots |0\rangle + e^{2i\pi 0.q_n} |1\rangle) \quad (4.5)$$

In the simulation, each operation is described by a unitary gate as described in Section 1.4. The swap procedure R_n is structured as a sequence of $n/2$ swap gates, and is defined by the matrix:

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.6)$$

The Q_n part is obtainable as a series of Hadamards on each qubit of the bigger register, followed by a number of and phase gates:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (4.7)$$

$$P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \quad (4.8)$$

4.2 Quantum errors

As said, there exists no such thing as an *ideal (or logic) qubit*. A realistic circuital simulation always has to take care of possible imperfections, which can never be totally removed. It is very difficult to sufficiently isolate the qubits from external noise, despite the rigorous conditions in which they are maintained. Each unit of the system is in fact a *physical qubit*, on which a procedure of error mitigation or correction has to be taken into account. Modern error correction technology can achieve error rates of about one in a thousand, but for large-scale quantum computers, an error rate of one in a million is needed. In classical information bits are either '0' or '1' states, therefore the only error type considered is the bit-flip, which takes $0 \rightarrow 1$ or vice-versa. On the other hand, the general qubit state can assume a continuum of values between its basis states and is subjected to an infinite number of error combinations. As anticipated, the general qubit state is written as a superposition of states

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \quad (4.9)$$

The coefficients α and β are complex numbers that satisfy the normalisation condition $|\alpha|^2 + |\beta|^2 = 1$. In the Bloch sphere formalism, specifying the two angles θ and ϕ , the general qubit representation becomes

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle. \quad (4.10)$$

The normalisation condition now becomes

$$\left|\cos\left(\frac{\theta}{2}\right)\right|^2 + \left|e^{i\phi}\sin\left(\frac{\theta}{2}\right)\right|^2 = 1. \quad (4.11)$$

The Bloch sphere is a useful concept because the most common errors are the ones in which the qubit is led to coherently rotate from one point of the surface to another. In this context, the effect of a unitary operation $U(\delta\theta, \delta\phi)$ is described as

$$U(\delta\theta, \delta\phi)|\psi\rangle = \cos\left(\frac{\theta + \delta\theta}{2}\right)|0\rangle + e^{i(\phi + \delta\phi)}\sin\left(\frac{\theta + \delta\theta}{2}\right)|1\rangle. \quad (4.12)$$

The parameters can vary continuously, so the error possibilities are endless. Usually, single-qubit coherent noise processes are described by matrices that can be expanded in terms of Pauli's ones. The basis set for two-dimensional matrices as the ones used here is composed of four objects:

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbb{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \mathbb{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \mathbb{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (4.13)$$

In this framework, every single-qubit error can be expanded on this basis using the appropriate expansion coefficients $\alpha_{\mathbb{1}, \mathbb{X}, \mathbb{Y}, \mathbb{Z}}$. This process is known as the *digitalisation* of the error in the Pauli set $\{\mathbb{1}, \mathbb{X}, \mathbb{Y}, \mathbb{Z}\}$ resulting in:

$$U(\delta\theta, \delta\phi) = \alpha_{\mathbb{1}}\mathbb{1}|\psi\rangle + \alpha_{\mathbb{X}}\mathbb{X}|\psi\rangle + \alpha_{\mathbb{Y}}\mathbb{Y}|\psi\rangle + \alpha_{\mathbb{Z}}\mathbb{Z}|\psi\rangle. \quad (4.14)$$

Recalling that $\mathbb{Y} \equiv \mathbb{X}\mathbb{Z}$ up to a phase, the description of each kind of discrete quantum noise can be reduced to a mathematical description involving Pauli- X and Pauli- Z .

4.2.1 Bit flips and phase flips

After digitalisation, only two fundamental quantum error types need to be accounted for: bit-flips and phase-flips. Quantum X -errors can be thought of as bit flips that map $|0\rangle$ in $|1\rangle$ and vice-versa. The general action is:

$$\mathbb{X}|\psi\rangle = \alpha\mathbb{X}|0\rangle + \beta\mathbb{X}|1\rangle = \alpha|1\rangle + \beta|0\rangle. \quad (4.15)$$

The second type, the Z -errors, represents phase flips. It has no classical analogue and maps qubits in the following way: $\mathbb{Z}|0\rangle = |0\rangle$ and $\mathbb{Z}|1\rangle = -|1\rangle$. On a general qubit state:

$$\mathbb{Z}|\psi\rangle = \alpha\mathbb{Z}|0\rangle + \beta\mathbb{Z}|1\rangle = \alpha|0\rangle - \beta|1\rangle. \quad (4.16)$$

4.2.2 Implementation of errors

Qiskit Aer provides an ad-hoc module to perform customised noisy model simulations via the class `pauli_error`. However, in this work, the elementary gates have been used with the direct introduction of Pauli X or Z errors in the circuit in order to increase the freedom in the error scheme's creation. From now on, the only rotations considered are

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (4.17)$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (4.18)$$

Each error is associated with a probability \mathcal{P} of occurring, and the complementary $1 - \mathcal{P}$ is the probability of not observing it. The QFT is implemented in Qiskit with the steps described above with the manual addition of error gates. Each one of them is implementing the operation:

$$U_{\sigma_{X,Z}} = \mathcal{P} \cdot \sigma_{X,Z} - (1 - \mathcal{P}) \cdot \mathbb{1}. \quad (4.19)$$

The circuit has been designed to allow the user to decide p as well as on which qubit or qubits place the error and at which steps of the circuit. In each case, the density matrix is calculated after the swap procedure, before the measures. After the simulation, the

density matrix corresponding to each noisy circuit is compared with the density matrix of the ideal QFT circuit (without noise). The quantity chosen to compare the two is the *trace distance*: a measure of the distinguishability between two states.

$$d(\rho, \sigma) = \frac{1}{2} \|\rho - \sigma\|_1 = \frac{1}{2} \text{Tr} \left[\sqrt{(\rho - \sigma)^\dagger (\rho - \sigma)} \right] \quad (4.20)$$

The trace distance d is indeed a metric on the space of density matrices and therefore is always non-negative, symmetric and satisfies both the triangle inequality and

$$d(\rho - \sigma) = 0 \Leftrightarrow \rho = \sigma \quad (4.21)$$

4.3 Adding a single error to the QFT

Several experiments were conducted with different circuit parameters. For each configuration, the experiment was repeated a number of times (defined by the parameter `runs`) and each time the density matrix and therefore the distance of it from the ideal one was calculated. Data are presented through the histogram of distances in the density matrix space. The parameters are the error probability \mathcal{P} , the total number of qubits n , the error type, and the position between 0 and n . As regards the error insertion, five different configurations were considered via the parameter `case`:

1. One error at the beginning before the first Hadamard,
2. One error before the swapping procedure,
3. One error after the swapping procedure,
4. n errors, one before each Hadamard,
5. n errors, one after each Hadamard.

A possible circuit with one error (`case= 1`) is the following: The distance value for a specific set of parameters is

$$d = \frac{1}{2} \|\rho_{\text{simulation}} - \rho_{\text{ideal}}\|_1. \quad (4.22)$$

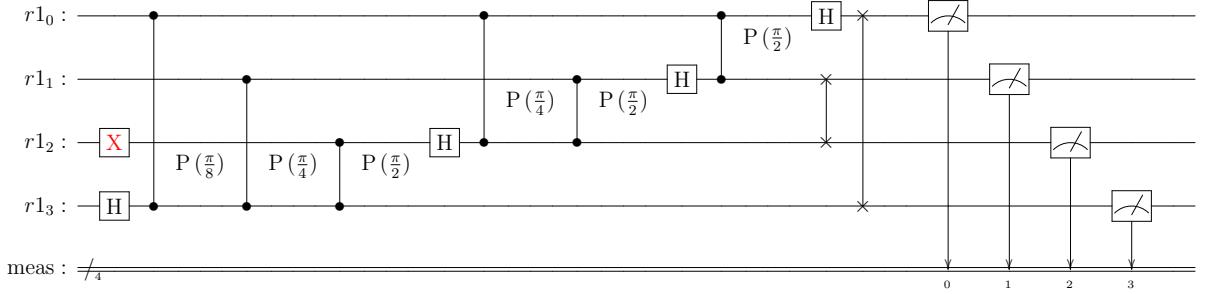


Figure 4.2: The circuit implemented on Qiskit for a Noisy - QFT

Most of the resulting histograms show an oscillation of the trace distance between two or three values, the percentage of runs that end up with $d \neq 0$ increase a higher value of the probability of error \mathcal{P} . Figure 4.2 shows the result for the simulation on the circuit in Figure 4.2. All the previous arguments are a benchmark for when the initial state is $|0\rangle$.

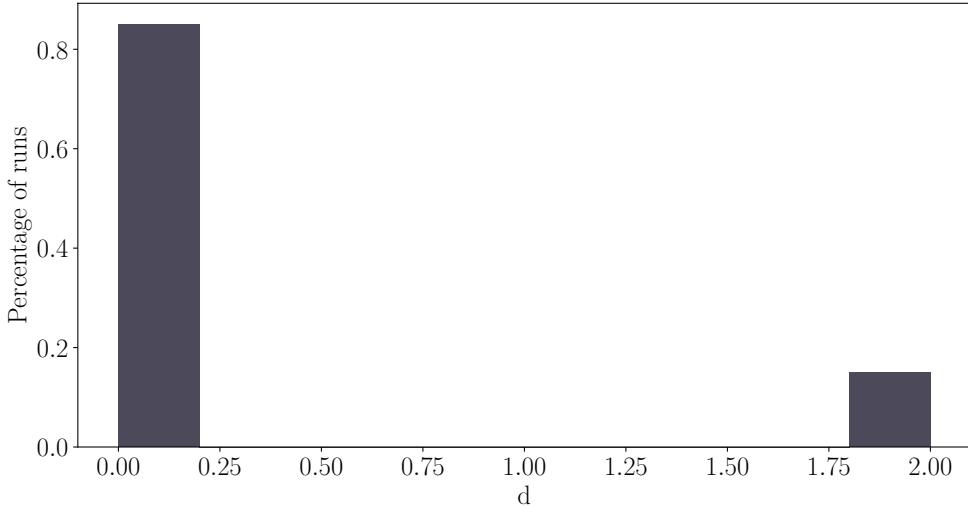


Figure 4.3: This histogram corresponds to a system of $n = 4$ total qubits where X -errors are applied on qubit = 2 with probability $\mathcal{P} = 0.2$ in case = 1. The circuit is performed runs = 200 times and the result is coherent with the probability of error of the 20%

There exist some particular error configurations for which $d = 0$, which are particular error set-ups in which the system is not modified by the unitary rotations. This happens

for example when a X -error acts after a Hadamard on zero

$$|\psi\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \mathbf{X} |\psi\rangle = \frac{|1\rangle + |0\rangle}{\sqrt{2}}, \quad (4.23)$$

or when a Z gate acts on a qubit with no phase (in case 1 in each position). The analysis is then naturally extended considering all possible initial states.

4.4 Single error generalisation

In this section, the number of qubits $n = 4$ and the error probability $\mathcal{P} = 1$ are fixed parameters. The value `runs= 1` is chosen because, with the $\mathcal{P} = 1$, the error will always be there and is not necessary to average over different simulations. The aim is to investigate the patterns as the error types and position change. Referring to the previous case distinction, only `case= 0` is inserted here, see B for the other four. In each plot, the y -axis corresponds to the distance value, while on the x -axis the states are represented in the following notation:

$$\begin{cases} 0 = |10000000\rangle \\ 1 = |01000000\rangle \\ \dots \\ 15 = |00000001\rangle \end{cases} \quad (4.24)$$

Case 1: single error before first Hadamard

As said before, the Z -error leaves the ρ identical because each initial state has zero phase at the beginning, for this reason, the graph is independent of the choice of the qubits on which the error is applied.

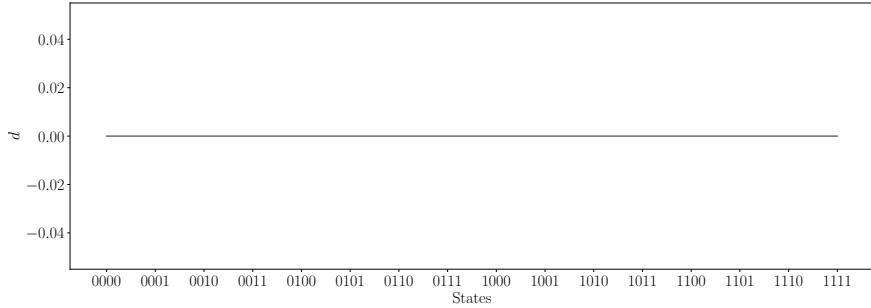


Figure 4.4: Z -error on qubit = 0, 1, 2, 3

As regards the X -error plots, there are no striking similarities in changing the position of errors: the intensities and positions of the peaks are different in each image. The only remarkable property to be noticed is that symmetry exhibits between the two halves of each plot.

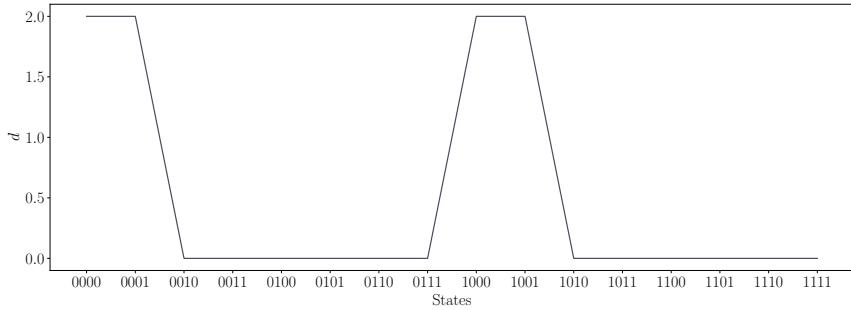


Figure 4.5: X -error on qubit = 0

In Table 4.1, data for all cases and positions are collected for both error types. The *mean* distance $\langle d \rangle_{states}$ reported is the mean value over all the base's states for each type and position. The label *noise* means that $\langle d \rangle_{states} \simeq 0$, this is a different situation from $d = 0$. In the second case, the value tells us that the qubit parameters are never flipped by any error, the first one means that the qubit is changed and then restored to its previous value, leaving behind some noise. Red values are associative with the configurations that maximise the distance from the ideal density matrix. No particular pattern seems to emerge from this data.

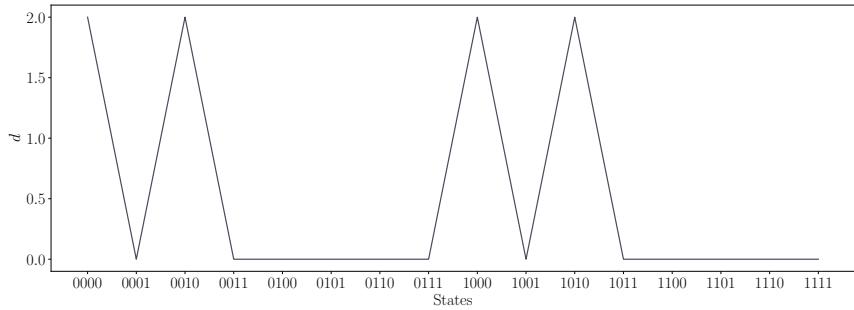


Figure 4.6: X -error on qubit = 1

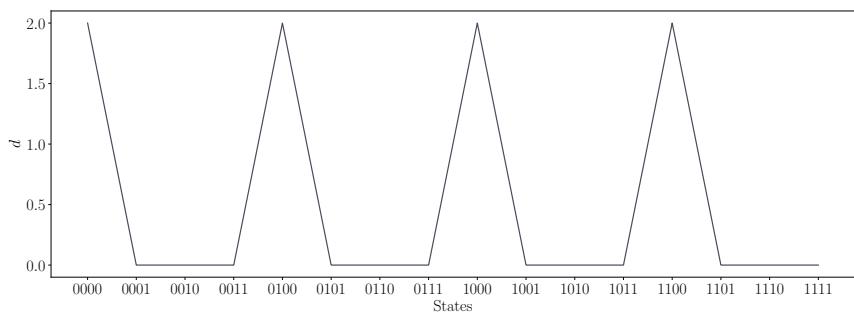


Figure 4.7: X -error on qubit = 2

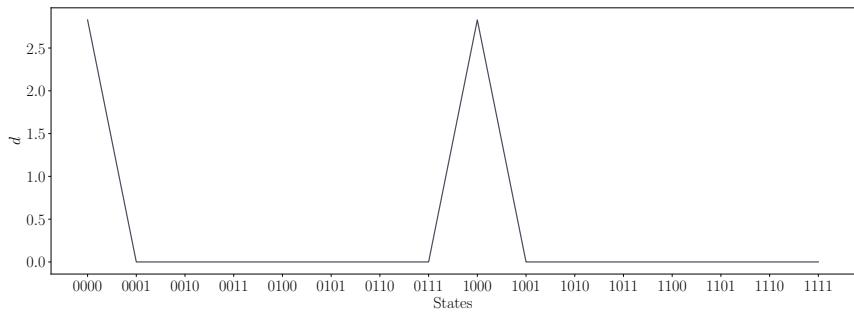


Figure 4.8: X -error on qubit = 3

	Case 1		Case 2		Case 3		Case 4		Case 5	
Error Type	X	Z	X	Z	X	Z	X	Z	X	Z
pos = 0	0.50	0.00	<i>noise</i>	0.35	<i>noise</i>	0.35	0.67	<i>noise</i>	0.75	0.35
pos = 1	0.50	0.00	0.40	0.76	0.23	0.58	0.50	0.76	0.53	<i>noise</i>
pos = 2	0.50	0.00	0.23	0.58	0.30	0.76	0.35	<i>noise</i>	0.50	0.58
pos = 3	0.35	0.00	<i>noise</i>	0.35	0.00	0.35	0.35	0.35	<i>noise</i>	<i>noise</i>

Table 4.1: $\langle d \rangle_{states}$ with $n = 4$

What is clear is that different states show different behaviour when afflicted by errors: some of them do not change, but the majority is not immune to rotations even when only one single-qubit rotation is inserted. Following this reasoning, another more informative perspective is taken into consideration.

4.5 Return probability

From the previous section is clear that a single error is enough to alter the transform's correct behaviour. Nevertheless, also the qubit and the position at which the errors are placed play an important role in determining its effect on the outcome. This section wants to put light on the *depth* parameter, to see if some kind of property is dependent on how deep the error is inserted. The procedure will again consist of a circuit which asks in input where to place the error (how deep in the QFT), and computes as output the *overlap*, defined by the following amplitude:

$$\langle \psi_0 | U_\varepsilon^\dagger U | \psi_0 \rangle^2. \quad (4.25)$$

In the equation, U is the unitary matrix of the circuit up to the point where the error is inserted and U_ε is another matrix representing the same circuit but with the error at the beginning and reversed. The idea is to cut the circuit at the point where the error appears, then bring the system back to the initial point and see how frequently initial states after the evolution return to themselves and if they do, with which probability. The overall purpose is to have a hint on how much the noisy quantum Fourier transform is 'chaotic' and in which ways the depth of the error is influential to determine the produced chaos. As a reasonable hypothesis, it is expected that the deeper (far from the first gate)

the error is inserted, the bigger the probability to end in a state orthogonal to the initial ψ_0 . The user can specify the value of `depth` as an integer number in the interval $[0, n+1]$ linearly proportional to the depth of the insertion. Then also the qubit where to insert the error via the parameter `position` and obviously the number of qubits n and the type of the error. Figure 4.9 shows a typical circuit with the pointed-out characteristics and in Figure 4.10 each position for a circuit of four qubits is linked to the corresponding parameter. In the related notebooks, `depth= 0` is designed to be the circuit without

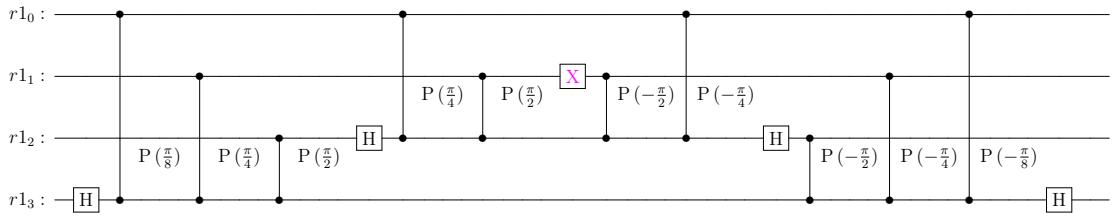


Figure 4.9: Circuit representing depth = 3, qubit = 1, $n = 4$

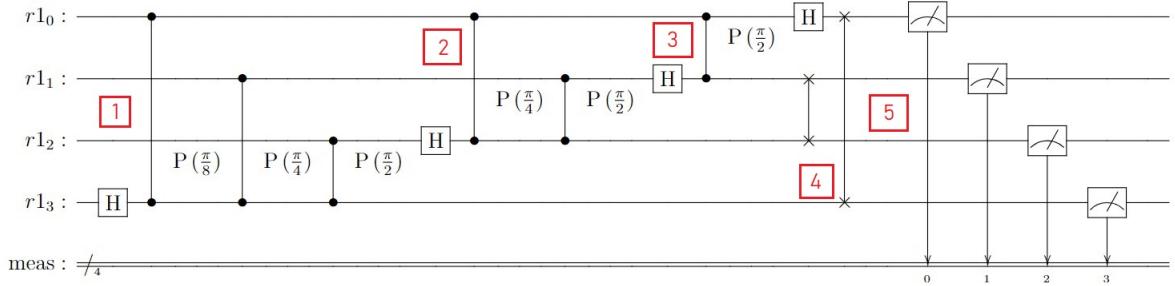


Figure 4.10: Error positions corresponding to each depth value

errors, in a way that makes an error-free circuit easily accessible at any time. The *unitary simulator* accessible in the Qiskit open-source package allows the extraction of the unitary matrix that models the implemented circuit. These are square matrices with dimensions $2^n \times 2^n$ with complex numbers as elements. The matrix for `depth= 0` when the error is absent coincides with the identity. For each one of the 2^n states of the basis, $\psi_{i=1\dots N}$ is evolved through the matrix operator $U_\varepsilon^\dagger U$ and from the resulting state-vector, the probability to re-obtain the i^{th} -state is calculated as a squared self-amplitude. Basically, the procedure extracts the diagonal elements of $M = U_\varepsilon^\dagger U$. If the computed value is 0, the error configuration erases the possibility of the state evolving in itself. Otherwise, the expected value of probability lies between 0 and 1, hopefully bigger than 0.5. The

circuit in Figure 4.9 is synthesised in the matrix M : The diagonal elements demonstrate

$$\begin{pmatrix} 0 & 0 & \frac{1}{4}+0.604i & 0 & \dots & 0 & \frac{1}{4}-0.604i & 0 \\ 0 & 0 & 0 & \frac{1}{4}+0.604i & \dots & 0 & 0 & \frac{1}{4}-0.604i \\ \frac{1}{4}-0.604i & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{1}{4}-0.604i & 0 & 0 & \dots & \frac{1}{4}-0.104i & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \frac{1}{4}+0.104i & \dots & 0 & 0 & \frac{1}{4}+0.604i \\ \frac{1}{4}+0.604i & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{1}{4}+0.604i & 0 & 0 & \dots & \frac{1}{4}-0.604i & 0 & 0 \end{pmatrix}$$

Figure 4.11: Matrix representing case= 3, $n = 4$, qubit = 1

that with this parameter setting, no state returns on itself.

4.6 Single error results

First, the circuit with the addition of one error with probability one only is checked. For a fixed number of qubits n , the amplitude is mediated over the possible error positions and over the states to find a mean return probability (RP) for each depth case. As said, the expectation is that this probability will become smaller, with a higher depth value. As regard the Z -errors, the behaviour in Figure 4.12 seems to resemble the expected one: RP= 1 with depth = 0 which means no error, then a linear decrease in amplitude up to zero probability of return for depth= 4, 5. A completely different scenario emerges in the case of X -errors in Figure 4.13. It appears that as errors are inserted deeper into the system, there is a tendency for the states to return to their initial configuration. A plausible explanation is that with longer circuits bit flips are masked and less influential. For instance, introducing one error in the fifth depth position leads to a situation where the average return probability exceeds one-half. After that, attention is shifted to the behaviour of single states to check if some of them have higher RP. In Figure 4.14 for X -errors the symmetry is back: first and eighth states behave similarly and the same happens for all the other state couples.

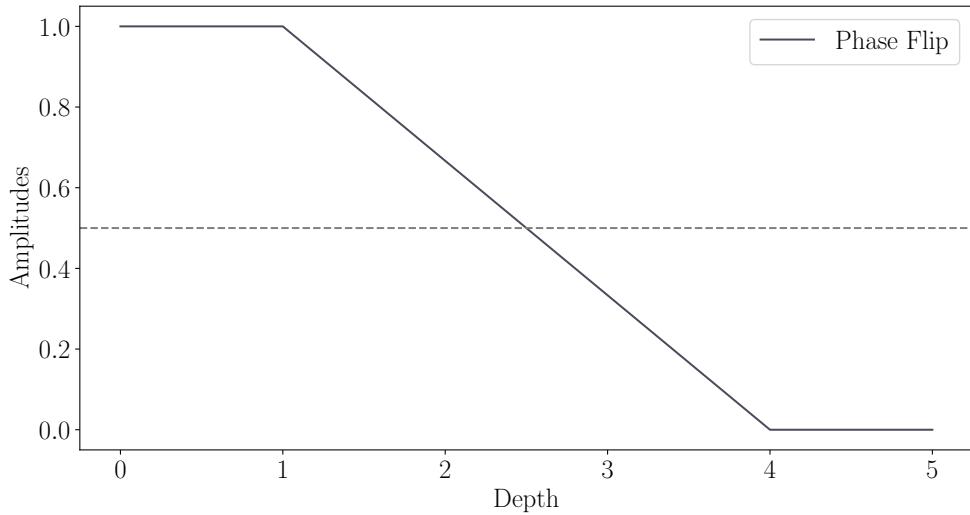


Figure 4.12: Mean amplitude for case with Z -errors, $n = 3$, fixed error position. For depth zero the unit value is a sign of the absence of error.

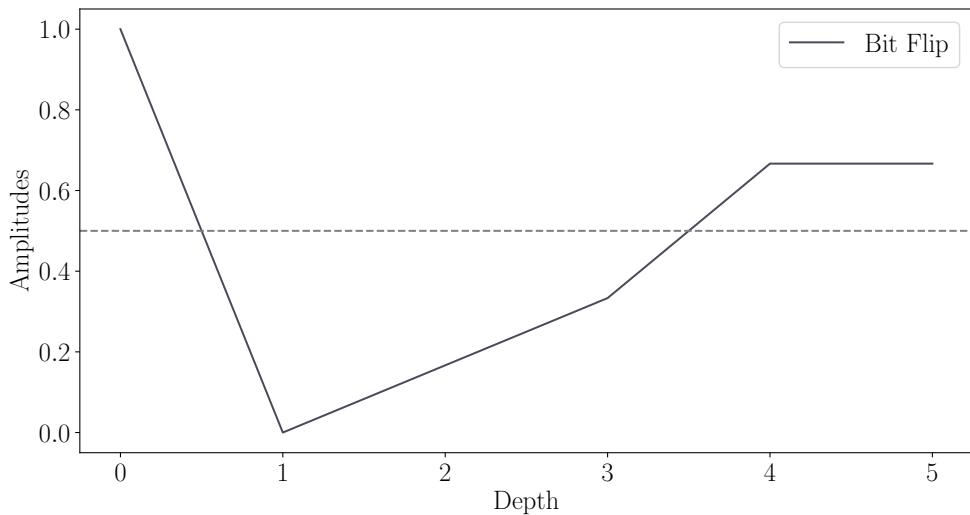


Figure 4.13: Mean amplitude for different insertions with X -errors, $n = 3$, fixed error position.

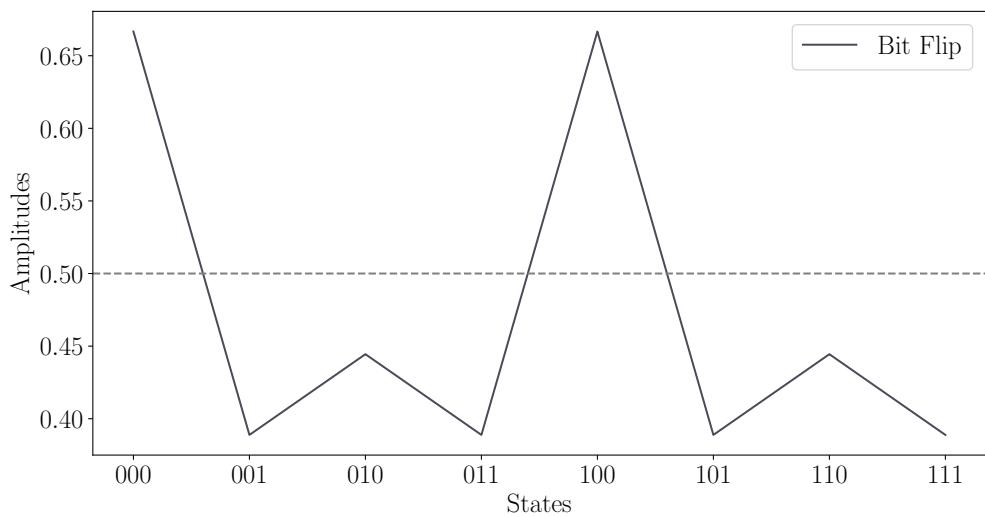


Figure 4.14: Mean amplitude for state with X -errors, $n = 3$, fixed error position

For Z -errors (Figure 4.15) all the states have the same mean probability to go back to themselves after the procedure equal to $RP = 0.5$. Combining all previous statements,

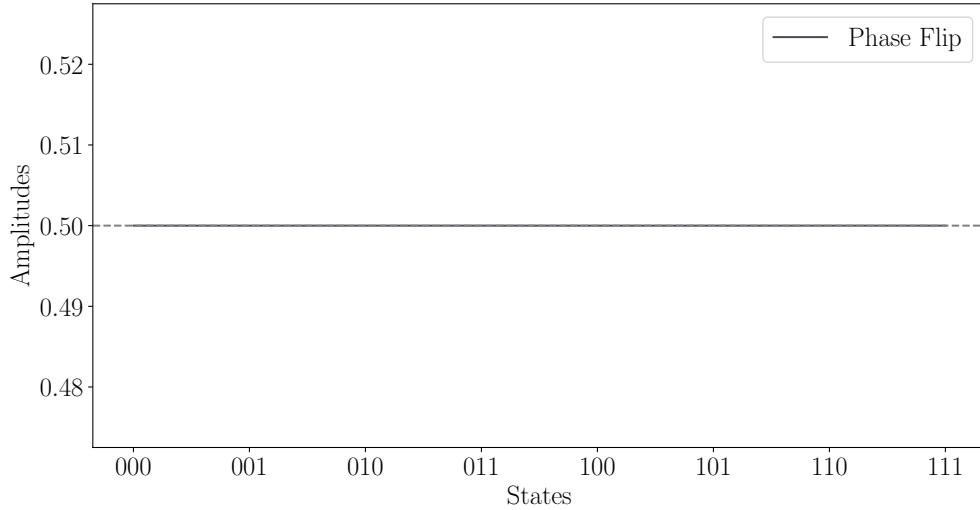


Figure 4.15: Mean amplitude for state with Z -errors, $n = 3$, fixed error position

the two distinct types of flips are integrated into the circuit with equal probabilities $\mathcal{P}_X = \mathcal{P}_Z = 0.5$. Now, the QFT behaviour aligns with the expectations and the depth of the error insertion is linearly proportional to the amplitude intensity.

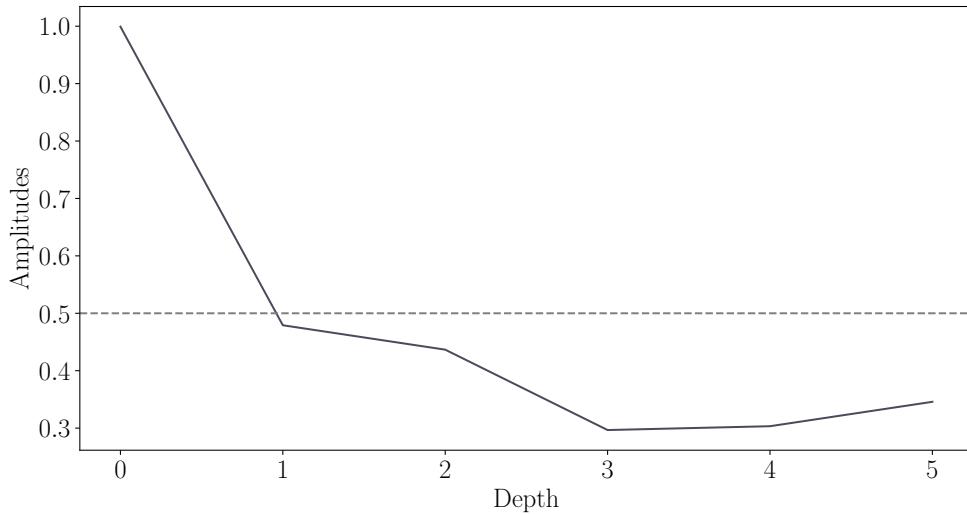


Figure 4.16: Amplitudes with random position and error type, $n = 3$

4.7 Why the symmetry?

Trying to understand why the first and the central state always seem to return more than the other ones begins with the observation of their tensor structure. For three qubits the states under examination are $|000\rangle$ and state number 8 which is $|100\rangle$: the only difference between the two is the value of the first qubit on the left. Searching for meaning leads to looking deeply into the Qiskit representation of states; the library uses the so-called *LittleEndian* formalism where the states are in the form $|q_{n-1}...q_2\ q_1\ q_0\rangle$ with the most significant bit on the left (MSB) and the least significant (LSB) one on the right, a notion opposite to the usual textbook formalism. The circuital transposition follows: The intuition is that the value of the most significant qubit in the QFT circuit does

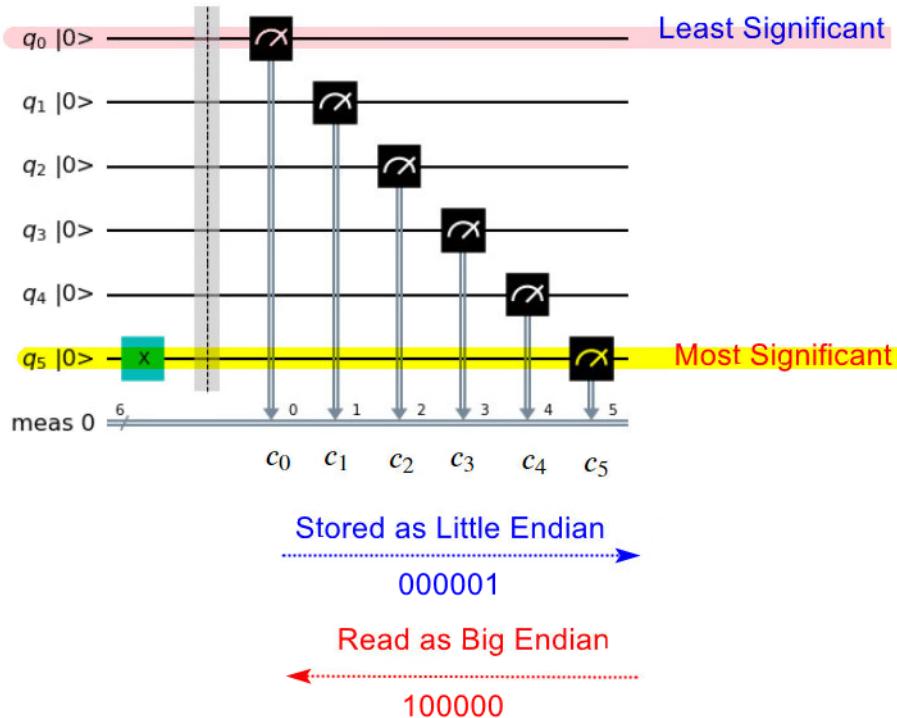


Figure 4.17: Qiskit state formalism

not influence the return probability. Moreover, looking at the structure of the controlled phases, Qiskit represents them as follows:

$$|q_1\ q_0\rangle = |\text{target control}\rangle . \quad (4.26)$$

Looking at the circuit of the quantum Fourier transform is clear that the $(n - 1)^{th}$ qubit is always a target and never a control, therefore errors on that qubit should be irrelevant for the correct performance of controlled gates. However, this does not explain *why* states like this return more. Looking at plots with enough repetitions as Figure 4.14, it is clear that the behaviour of states under the quantum Fourier transform is a broken line with two maxima at the centre and the beginning and two or more minima. Demonstrating why $|000\rangle$ and $|100\rangle$ return more starts with the previous definition of QFT:

$$F_n |q_1 q_2 \dots q_n\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2i\pi 0.q_n} |1\rangle) \otimes (|0\rangle + e^{2i\pi 0.q_{n-1}q_n} |1\rangle) \otimes \dots (|0\rangle + e^{2i\pi 0.q_1q_2\dots q_n} |1\rangle) \quad (4.27)$$

With the two states of a three-qubit quantum system under examination, the action is the following:

$$F_n |000\rangle = \frac{1}{\sqrt{8}} (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle), \quad (4.28)$$

and

$$F_n |100\rangle = \frac{1}{\sqrt{8}} (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle). \quad (4.29)$$

The distinction is a sign in the last term, however, the states behave in the same way concerning errors. Both of them, looking at several error combinations, have the highest probability of being left unchanged. They do not return ($RP = 0$) for phase shifts and always return ($RP = 1$) with X -errors. It is therefore possible to say that the unitary matrix of the QFT commutes with the bit flip operator on the last qubit:

$$[M, X_{n-1}] = 0. \quad (4.30)$$

4.8 Multiple error results

In this section, a similar simulation is performed this time including more than one error for each circuit: one *before* each Hadamard and one after the swapping procedure. The previous `depth` parameter becomes now a measure of the number of unitary single-qubit rotations inserted. In this sense `case=0` means no errors are inserted, `case=1` that there is one single error and so on. With fixed n and $\mathcal{P}(\mathbb{X}) = \mathcal{P}(\mathbb{Z}) = 0.5$ return probabilities for each state and an increasing number of errors are computed in two distinct set-ups:

1. all errors positioned on the same qubit;
2. each error on a randomly chosen qubit.

As in the previous section, circuits are constructed to be truncated after the last error and then reversed without any X or Z gates as in Section 4.8. The first result is the heat map

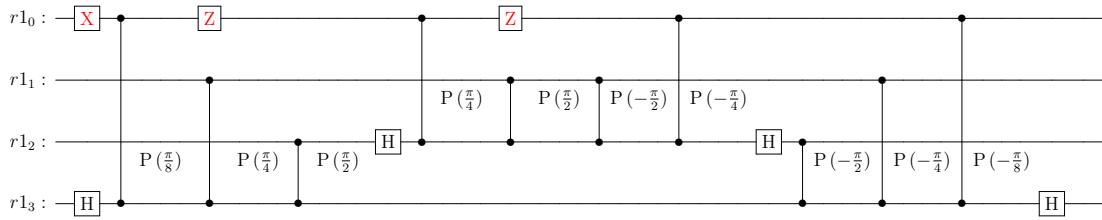


Figure 4.18: Number of qubits $n = 4$, three errors, position = 0

in Figure 4.19 where the colour intensity is proportional to the return probability and a symmetric pattern is highlighted. The same data are linearized in Figure 4.20 where each line corresponds to a row of the heatmap. For each case k there are k errors up until a maximum of $n + 1$. They are either \mathbb{X} or \mathbb{Z} rotations in n possible positions, therefore, the total number of possibilities is $2^{(n+1)\cdot n}$. For $n = 3$ the total number of configurations is $2^{12} = 4096$ and for $n = 4$ is 2^{20} . These values should represent a threshold for the number of repetitions. For computational reasons, it was not possible to perform so many circuits; therefore the parameter, particularly for $n = 4$, was lowered to simplify the simulations. For three qubits, zero error implies $\mathcal{P} = 1$ of returning at the initial state after the circuit execution. The rest of the cases suggest that the first and middle states return more than others, confirming the previous assumptions. Case 1, where a single error is inserted with 50/50 probability of being a X or Z error, is $\mathcal{P} = 0.5$ as expected. Only the X -errors affect the return probability of the state if placed before the first Hadamard.

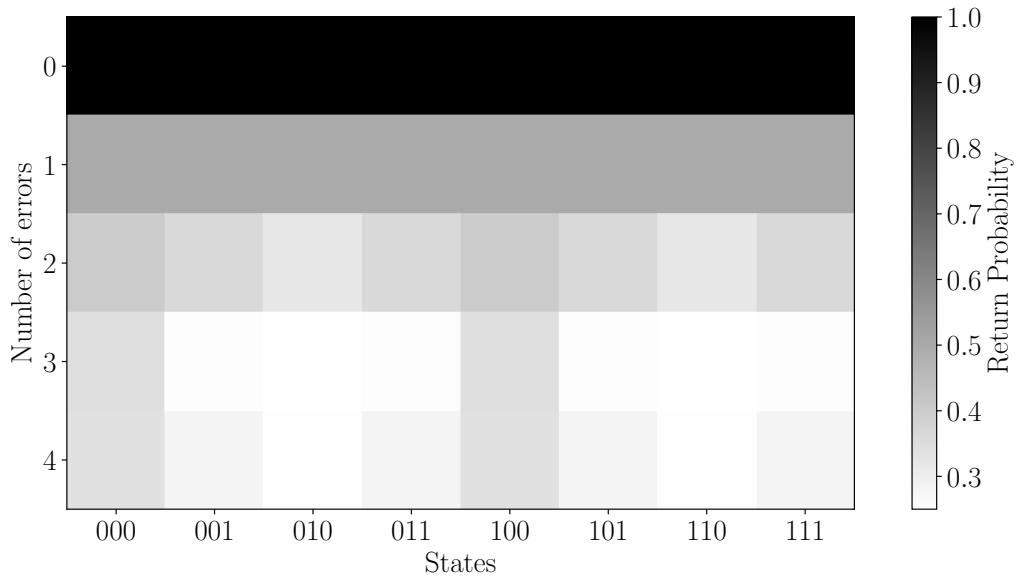


Figure 4.19: Return Probability for a system of $n = 3$ qubits, 200 runs

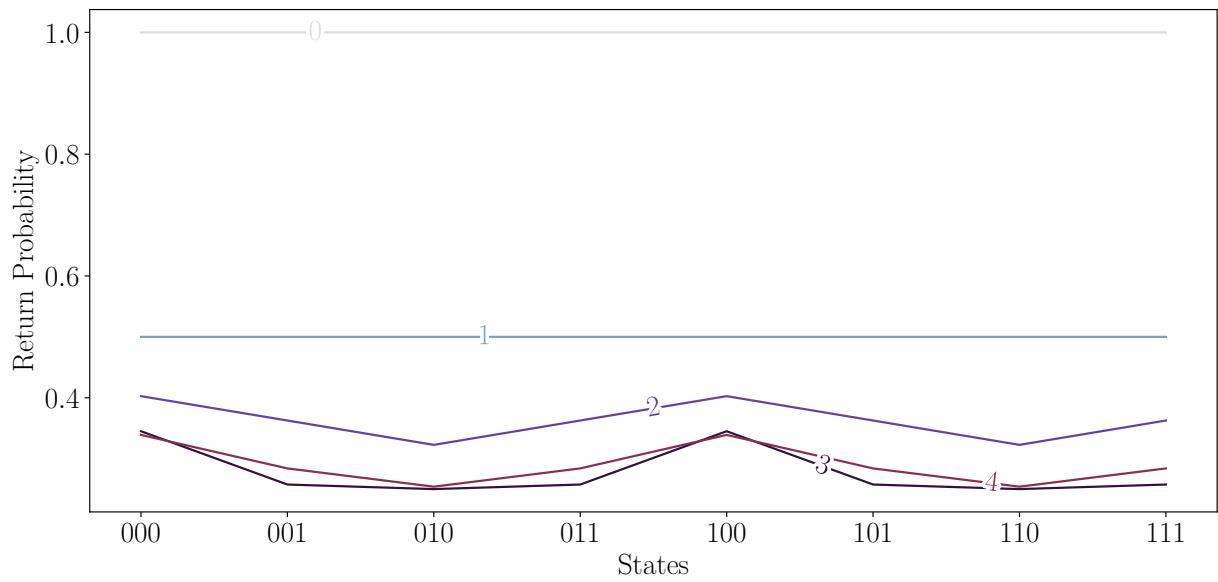


Figure 4.20: Return Probability for a system of $n = 3$ qubits, 200 runs

With $n = 4$ qubits results shown in Figure 4.21 and Figure 4.22 are similar. The independence from the most significant qubit is still present also if the number of repetitions is not high enough.

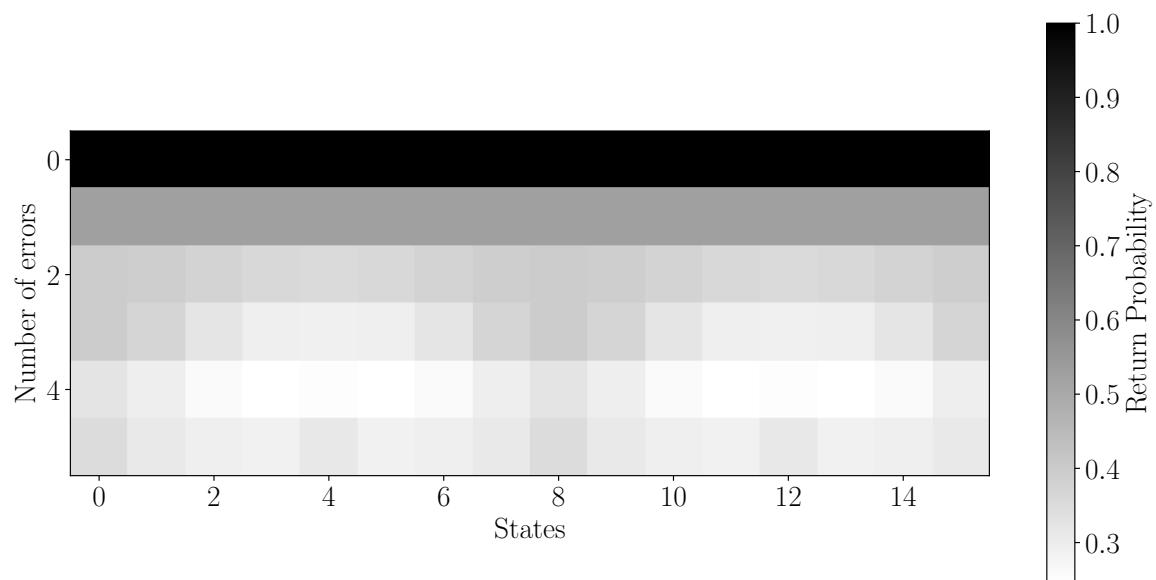


Figure 4.21: Return Probability for a system of $n = 4$ qubits, 500 runs

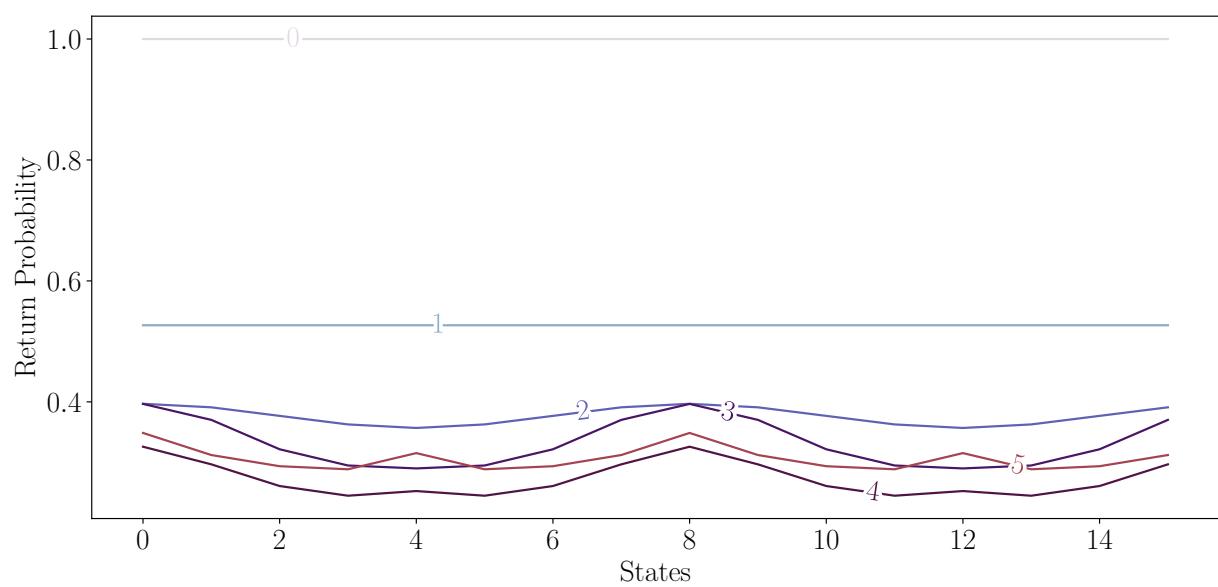


Figure 4.22: Return Probability for a system of $n = 4$ qubits, 500 runs

The analysis is repeated with the same parameters, but positioning the errors on random qubits. Section 4.8 is an example of the insertion of two errors. Linearising the heat map

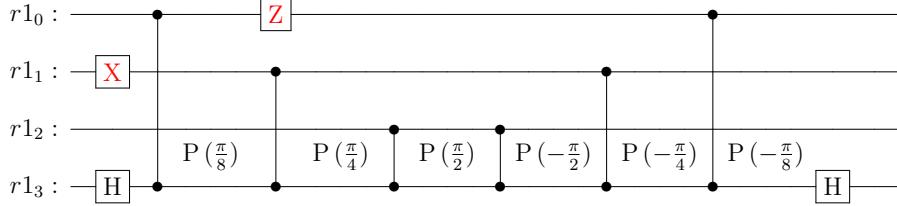


Figure 4.23: $n = 4$, case = 2

as in Figure 4.24 for $n = 3$ qubits is possible to see again the same pattern for 0 and 1 unitary rotation, with $\mathcal{P} \simeq 0.5$ in the second case. Confronting with Figure 4.20 it is clear that inserting errors on random qubits and not all on the same one, lowers the return probability intensity by a small amount. The same thing happens for $n = 4$

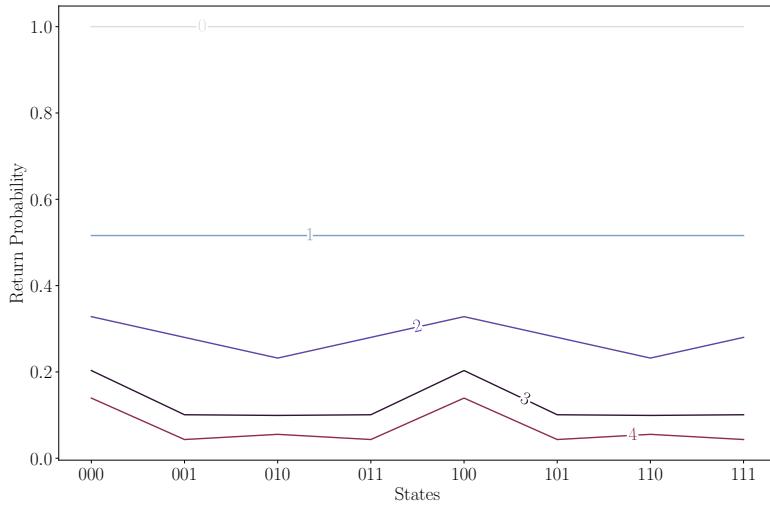


Figure 4.24: Return Probability for a system of $n = 3$ qubits, 500 runs

qubits in Figure 4.25, in particular for more than 3 errors. In conclusion, $n = 3$ is taken as a reference to comprehend what happens if only one type of error gate is used. In Figure 4.26 only Z -errors are used, while in Figure 4.27 only bit flips are inserted. The impact of X -errors seems to be more decisive in this particular construction in lowering the return probability.

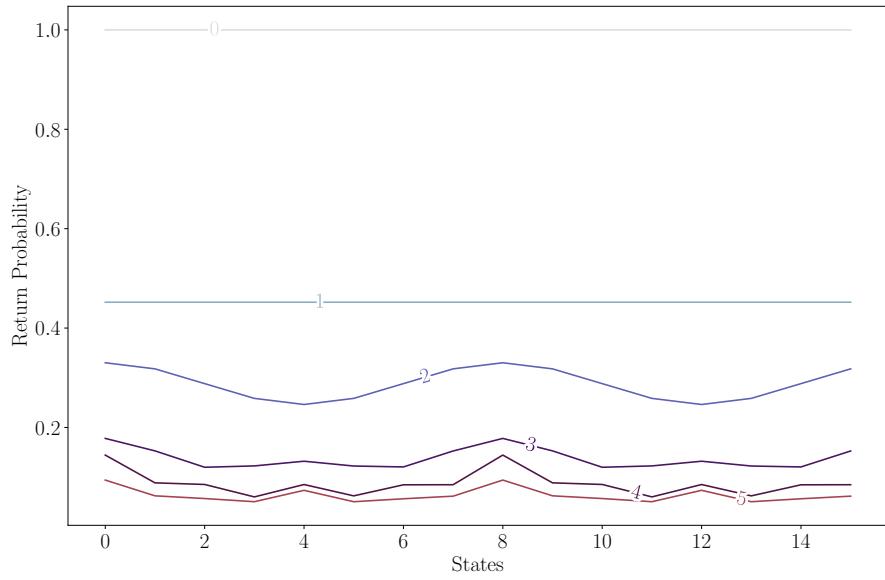


Figure 4.25: Return Probability for a system of $n = 4$ qubits, 500 runs

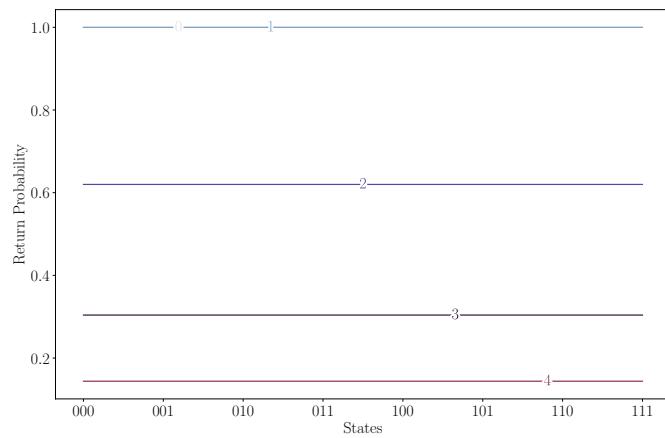


Figure 4.26: Return Probability for a system of $n = 3$ qubits, 250 runs, only Z -errors are applied with probability one in different positions, then results are mediated over all the repetitions.

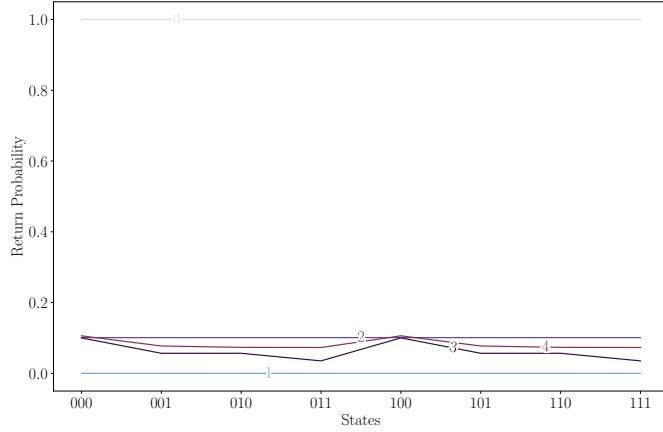


Figure 4.27: Return Probability for a system of $n = 3$ qubits, 250 runs, only X -errors are applied with probability one in different positions, then results are mediated over all the repetitions.

4.9 Amplitude attenuation

The concept of *attenuation* in general, is defined as a measure of flux intensity decreasing within a medium: it is of common use in waves and signal propagation phenomena. Rephrasing this in the context of this work, the attenuation can be interpreted as a quantity proportional to the self-overlap of a state after its transmission through the circuit (the medium). Two kinds of mean values are considered. In the following plots: ε is the self-overlap, $n_{\text{errors}} \in [0, n_{\max}]$ and $n_{\max} = n + 1$ is the maximum number of errors for a circuit of n -qubits. Each amplitude ε_k is a value in the interval $[0, 1]$ therefore, $\log(\varepsilon_k) \leq 0$: this explains the minus sign in both the y -axis. The logarithm is a concave function and the inequality for the logarithm of the arithmetic-mean/geometric-mean is satisfied:

$$\log\left(\frac{\sum_i x_i}{n}\right) \geq \sum_i \frac{\log(x_i)}{n}. \quad (4.31)$$

Given the property of logarithms

$$\sum_i \log(x_i) = \log\left(\prod_i x_i\right), \quad (4.32)$$

the right side of the inequality in Equation (4.31) represents the logarithm of the geometric mean and the left side is the logarithm of the arithmetic mean (correctly $\text{AM} \geq \text{GM}$). The

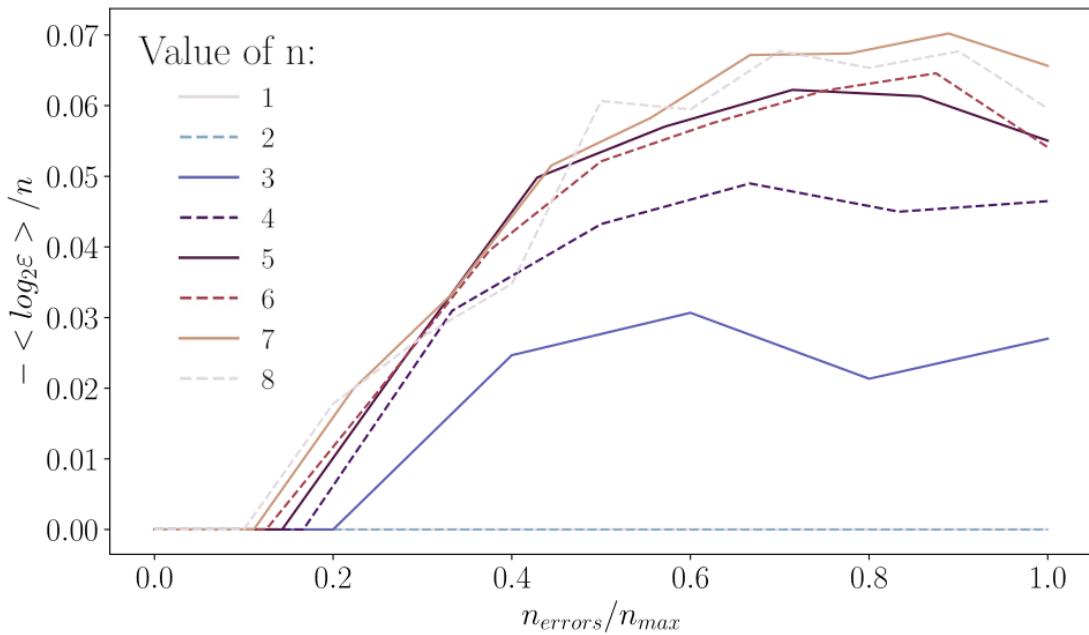


Figure 4.28: Mean of the logarithm of the amplitude over 500 iterations

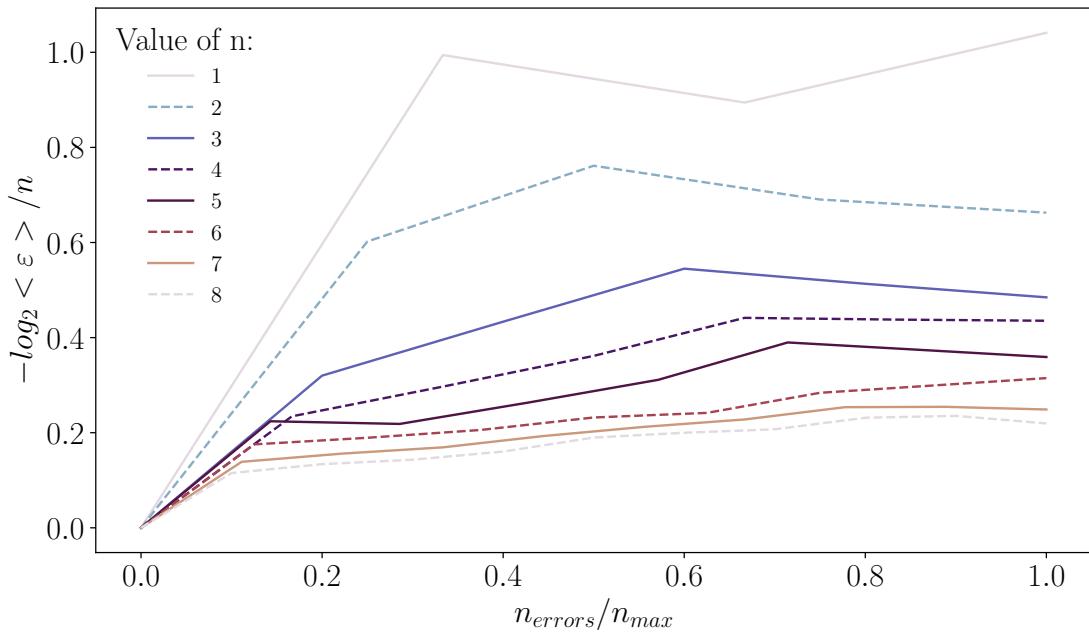


Figure 4.29: Logarithm of the mean amplitude over 500 iterations

geometric mean represented in Figure 4.28 is a measure of the average growth rate of the attenuation. The results show that the bigger the system, the more the error rate grows. This is obvious because the more qubits we use to build the circuit, the more errors can be added. On the other hand, Figure 4.29 refers to the arithmetic mean and shows the mean value of the error amplitude for each n ; here is clear that as the circuit grows, the error rate decreases. Adding up information the pictures demonstrate that with a growing number of qubits, the total error grows more because the length of the circuit allows more fallacies, but overall the length is directly proportional to the probability of the circuit to auto-delete the error and ending up in the correct state's configuration.

Chapter 5

Noisy-QFT for Factoring Numbers

In this chapter, previous arguments about the random rotations and the quantum Fourier transform are combined, and a noisy-QFT is integrated into the circuit for the integer factorisation described in 2. A series of simulations are performed in order to verify if an increasing number of unitary X and Z rotations in the QFT impacts the outcome to the point where there is no longer any advantage in using Shor's algorithm over a classical one.

5.1 Selecting the right witness quantity

At first, a noisy QFT is inserted in circuit for factoring number 15; this time the number of rotations in the routine goes from 0 to $n + 1$, one before each Hadamard and one after the swaps. The factorisation is performed 350 times where each of them has a fixed number of errors on random qubits with a 50/50 probability to flip a phase or a bit. In the ideal scenario, without any discrete errors, the factorisation generates the statistics in Figure 5.1. The result may not appear extremely satisfactory being highly dependent on the choice of the co-prime y ; with $y = 8$ however, at least correct factors is found with probability $\mathcal{P} \simeq 0.8$. On the other hand, in Figure 5.2 the same experiment is repeated with equal parameters except for the number of errors, raised to the maximum value of 5. Surprisingly, outcomes are not in contrast as one would think: the probability of success

is again $\mathcal{P} \simeq 0.8$.

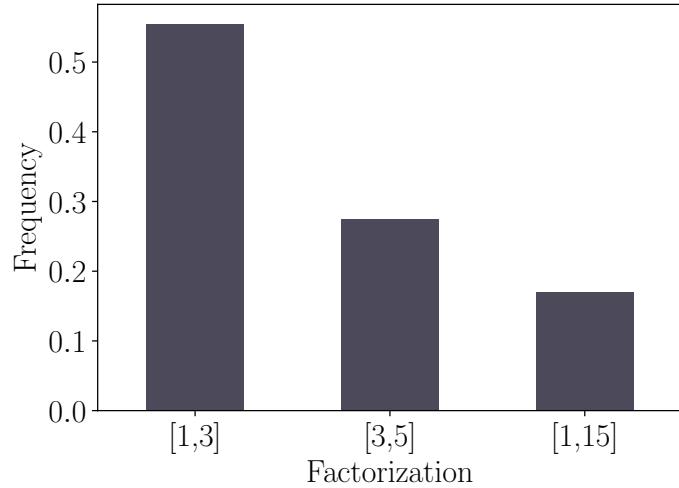


Figure 5.1: Factorisation of $N = 15$ with $y = 8$, 350 runs and 0 errors

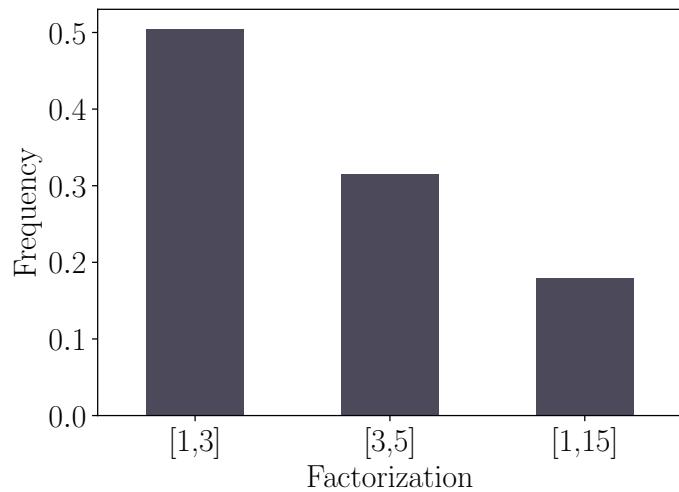


Figure 5.2: Factorisation of $N = 15$ with $y = 8$, 350 runs and 5 errors

To eliminate the dependency on the co-prime choice, a possible reason for this unanticipated outcome, the same integer is factored a number of times with random y each time.

Errors	1×3	3×5	1×15
0	26%	30%	44%
1	28%	31%	41%
2	28%	30%	42%
3	31%	29%	40%
4	33%	29%	38%
5	30%	27%	43%
Mean	29.3%	29.3%	41.3%

Figure 5.3: Output statistics for mixed X and Z rotations

The co-prime integer is chosen from the set of possible numbers such that $\gcd(y, N) = 1$:

$$y(N = 15) = [2, 4, 7, 8, 11, 13] \quad (5.1)$$

Results are collected in Table 5.3, demonstrating that the three outputs, regardless of the number of errors, are more or less equally likely, but 1×5 has a slightly higher probability. Representing data from the table as lines in Figure 5.4, the proportions between them are much clearer: the correct outcome is not preponderant. Moreover, isolating the effect

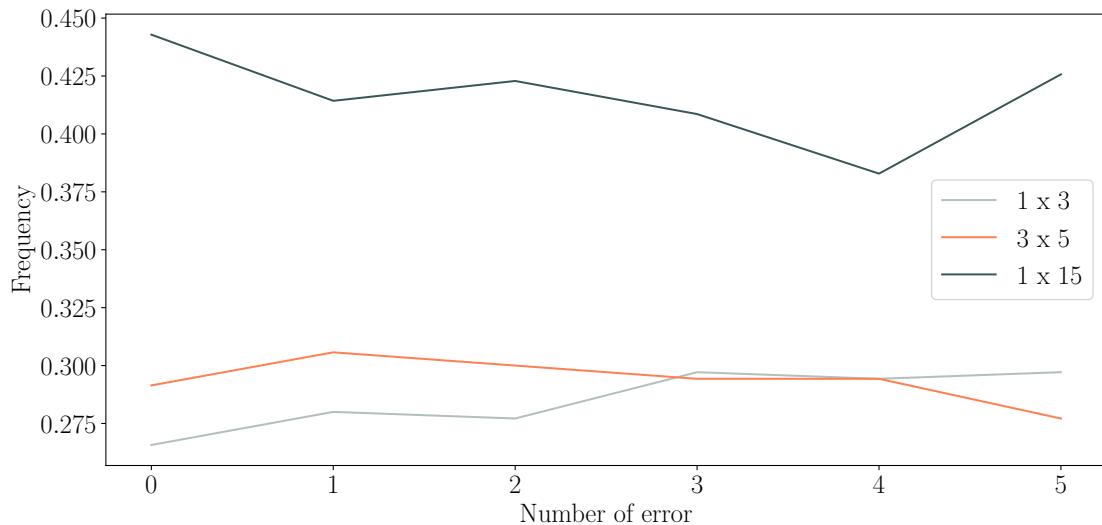


Figure 5.4: Linearized representation of data

of X -errors only (see Figure 5.5) still shows results different from the expected ones.

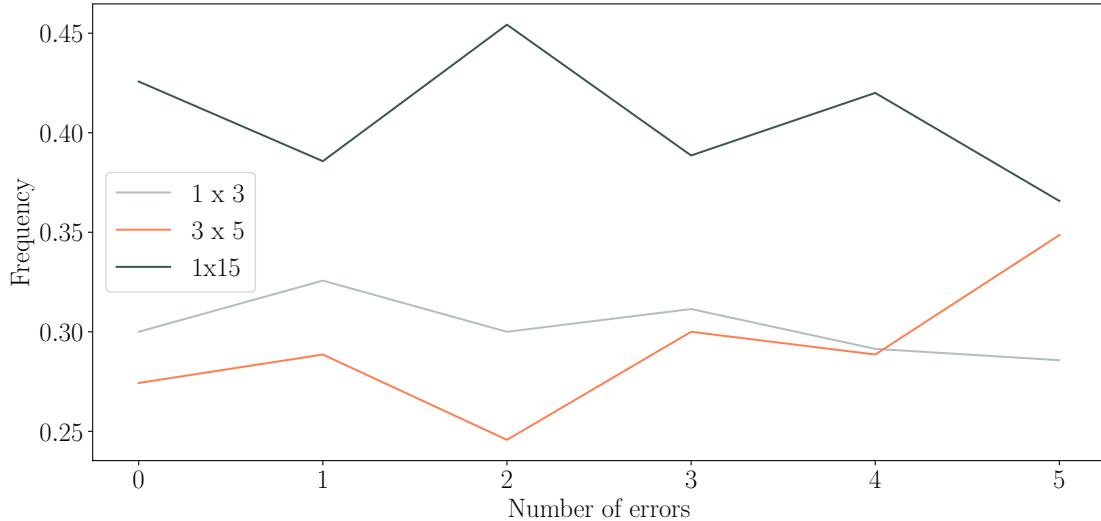


Figure 5.5: Linearized representation of data with bit flips only

The main problem is that adding errors to the algorithm seems not to change its performance, if one only looks at the values of p and q . The fact that 15 is one of the simplest possible numbers to factor with the Shor algorithm, does not justify that no errors can modify its efficiency. The solution arises when the subroutine for finding the two factors is tested with fixed co-prime.

```
#COMPUTATION OF p AND q
for r in range(15):
    print(gcd(y**((r//2)-1, N), 'X', gcd(y**((r//2)+1, N)))
```

The result of these code lines is the following:

15 X 1	15 X 1	1 X 3	1 X 3	3 X 5	3 X 5	1 X 3	1 X 3
15 X 1	15 X 1	1 X 3	1 X 3	3 X 5	3 X 5	1 X 3	

Whichever the period r , the classical number theory behind the algorithm is so well-designed that it *always* finds the correct factors, or at least one of them. Therefore, based

on the preceding arguments, it has been demonstrated that looking at the final values of the factors is not an efficient approach to determining whether and how the quantum algorithm of Shor is working properly. An alternative is needed for this purpose.

5.2 Useful states and robustness

In this section, a new strategy to challenge Shor's behaviour under the effect of discrete errors is defined. First, this method will be performed on $N = 15$, to check that everything is consistent with the previous chapters and to define a benchmark for $n = 4$ qubits. Next, the analysis will shift to $n = 5$ and $n = 6$ qubits with higher values of N , as far as the computational power allows going. The analysis will consist of a deep investigation into the period-finding procedure in search of the easiest way to discriminate between a right or wrong factorisation outcome with the smallest number of operations.

Simon Dewitt and Austin Fowler in their work [DFH06] suggested that an efficient way of studying the robustness of Shor by looking at the quantum procedure only, without considering successive classical operations to find p and q . In Chapter 2 the algorithm was described as composed of a quantum part dedicated to finding the period $r > 0$ of $f(x) = x^k \text{mod} N$ (the integer $r \mid f(r) = 1$), and the classical computation of the gcd. The period search exploited the continued fractions representation of the number c measured with a single shot simulation on the circuit. As seen in 2.15, after the measurement, there is a correspondence between the outcome and a phase:

$$\frac{c}{2^{2n}} \approx \frac{k}{r}. \quad (5.2)$$

Here $n = \log(N)$ is the necessary number of qubits and k is a constant such that $0 \leq k < r$. Going back to the investigation of the robustness of the circuit, the problem is rephrased as follows: the new witness for the algorithm's efficiency is the frequency of measuring a specific state:

$$c^* = \frac{2^{2n}}{r}. \quad (5.3)$$

The problem is somehow reversed. Now, assuming the knowledge from classical calculations of the value of the correct period, and fixing $k = 1$, the aim is to determine how many times the algorithm successfully measures the corresponding state. This kind of

state c^* will certainly lead to a correct factorisation: its frequency as the error number increases is the target of the following dissertation.

Before showing the results, an insight into the error system is necessary. In the Python notebooks, error insertions up to this point were described by the function:

```
#ERROR DEFINITION
def errori(circuit, position, n):
    epsilon = 0.5
    extracted = random.uniform(0, 1)
    position = random.randint(0, n)

    if (extracted < epsilon):
        circuit.x(position)
    else:
        circuit.z(position)
    return circuit
```

The insertions of a bit flip in a predefined random position happen with frequency $\mathbf{f} = \epsilon$ while the phase flip has $\mathbf{f} = 1 - \epsilon$. From now on, this function continues to be employed for each error gate; however, the difference is that errors are now inserted *after each gate* (both Hadamard and Phase Gate), thus enabling a significantly greater number of insertions. In general, the total number of errors (gates) per n is:

$$\#_{err} = 2n + \frac{2n \cdot (2n - 1)}{2} \quad (5.4)$$

$N = 15$ is the simplest and fastest case; so, for this integer only, it was possible not to use a fixed y but choose a random co-prime between the possible ones. The controlled matrices of the modular exponentiation have been calculated from scratch each time, although being the computationally most expensive part. On the contrary, for bigger N 's, y is fixed and the $2n$ controlled unitaries are computed separately in advance to speed up the computation as much as possible. For $N = 15$, the possible error insertions are $\#_{err} = 36$, each of which on $2n = 8$ possible qubits, and of two different natures (X

and Z). With the chosen parameters saw in Section 2.4 the obtained state is:

$$c^*(N = 15, r = 4) = \frac{2^8}{4} = \frac{256}{4} = 64. \quad (5.5)$$

An important observation is that the value of c^* is independent of the co-prime y , in contrast to what happened with the analysis of ps and qs . The simulation is performed a number of times for each error addition, and the success probability is calculated in the following frequentist way:

$$P(c = c^* | \text{fixed number of errors}, N, r) = \frac{\text{runs whose outcome is } c^*}{\text{number of runs}} \quad (5.6)$$

In Figure 5.6 results are reported and each point represents the frequency over 50 runs. After more or less ten errors the output becomes random and the algorithm performs only slightly better than randomly choosing a value of c in the range $[0, 2^{2n}]$ (eventually highlighted by the horizontal line). The probability of finding the right period $r = 4$ tends to decrease exponentially, however approaching a non-zero value: as seen before for this simple N the algorithm demonstrates a high level of performance.

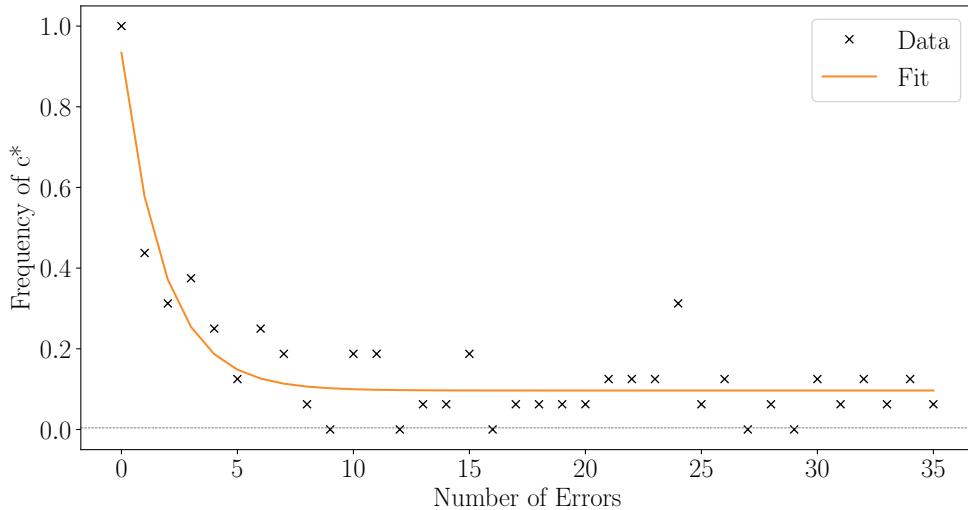


Figure 5.6: Results for $N = 15$

The expectation is that the behaviour of the system for a bigger number of qubits will be similar to this one. Going to more significant simulations, the procedure starts with the choice of the numbers N_i to factorise and their corresponding y_i . In order to

do so, a period is chosen and maintained fixed, here $r = 6$, and couples (N_i, y_i) are chosen consequently. In this way, the value of c^* is dependent only on the number of qubits $n_i = \log_2(N_i)$. In Table 5.18, chosen parameters for the following simulations are reported:

N	y	n	circuit n	$\#_{err}$
15	8	4	12	36
27	8	5	15	55
63	31	6	18	78
247	27	8	24	171

Figure 5.7: Simulation parameters for having $r = 6$

Usually, for large-scale quantum algorithms, the general precision for a single component of the circuit is considered to be: $n_p^{-1} = (K \cdot Q)^{-1}$. This value represents the number of positions where an error can occur given a circuit of Q qubits and K steps (depth of the circuit). In the previous pages, this has been defined as $\#_{err}$. As an example, for a circuit of $n = 8$ qubits and $\#_{err} = 171$ the total number of configurations will be $\simeq 2^{2n \cdot 171}$. Given the limitations of a Python simulation, conducting and analysing beyond a certain point is unfeasible. Hence, in this study, the factorisation of only $N = 27$ and $N = 63$ will be conducted and examined. A complete analysis of the numbers in Table 5.18 would be impossible mainly due to limitations in time and computational resources. It is remembered that the pre- and post-processing operations can always be performed in polynomial time using classical techniques, so the computation time only depends on the quantum period finding now under investigation.

A consideration is fundamental before the data observations. In calculating c^* , the value of k has been arbitrarily set equal to one, so that only a single state was counted as useful. Looking more carefully at the simulation results there is a simplification worth making. Having to do heavy simulations with bigger N s, is reasonable to find a way to consider the maximum number of states as correct, and not limit the analysis to $k = 1$ reducing the already few data available. Given that $0 \leq k < r$, c^* can be generalised to the *set of states* associated with r :

$$\frac{c}{2^{2n}} \approx \frac{k}{r}$$

Factor correctly here means that the algorithm finds at least one of the two factors. Going back to $N = 15$ $k = 0, 1, 2, 3$ and therefore the possible values are:

$$c* = [0, 64, 128, 192]. \quad (5.7)$$

This set is obviously dependent on the length of the chosen period, the reason for which $r = 6$ will be a constant in the two simulations in merely a choice to reduce the number of simulative computations.

A last approximation is done. Figure 5.8 shows the result of a 2048 shots simulation. It is seen that if 2^{2n} is perfectly divisible by r , the simulation will give as a result a peaked histogram for the final statistics: While this always happens with $N = 15$, going

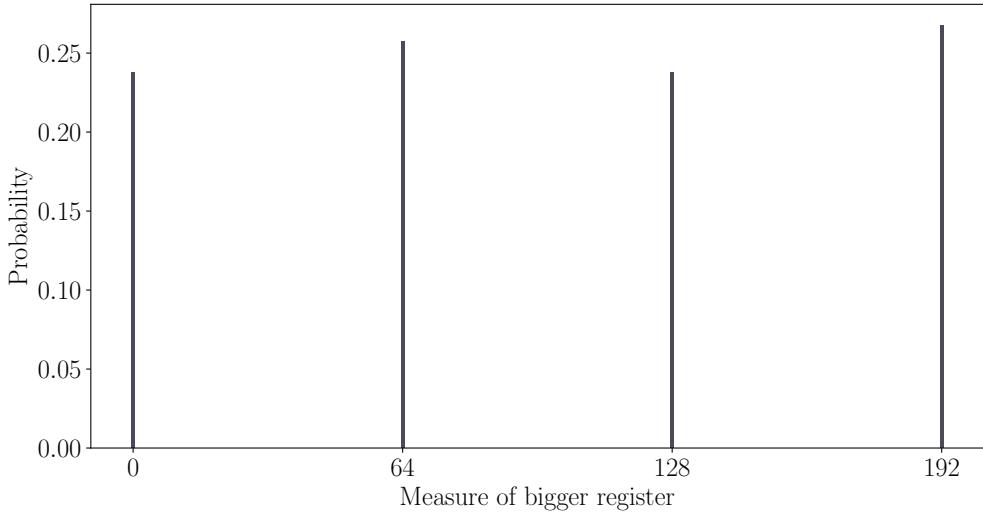


Figure 5.8: Zero errors, $N = 15$, $r = 4$

up to higher numbers leads to a different situation where $2^{2n}/r$ is a real number and the distribution is broader as seen in Figure 5.9.

Clearly, the simulation has to distinguish between the two situations. For doing so, the final set of states considered useful for an effective factorisation is defined as:

$$c_{useful}^* = \lfloor k2^{2L}/r \rfloor, \lceil k2^{2L}/r \rceil \quad \forall k \mid 0 < k < r \quad (5.8)$$

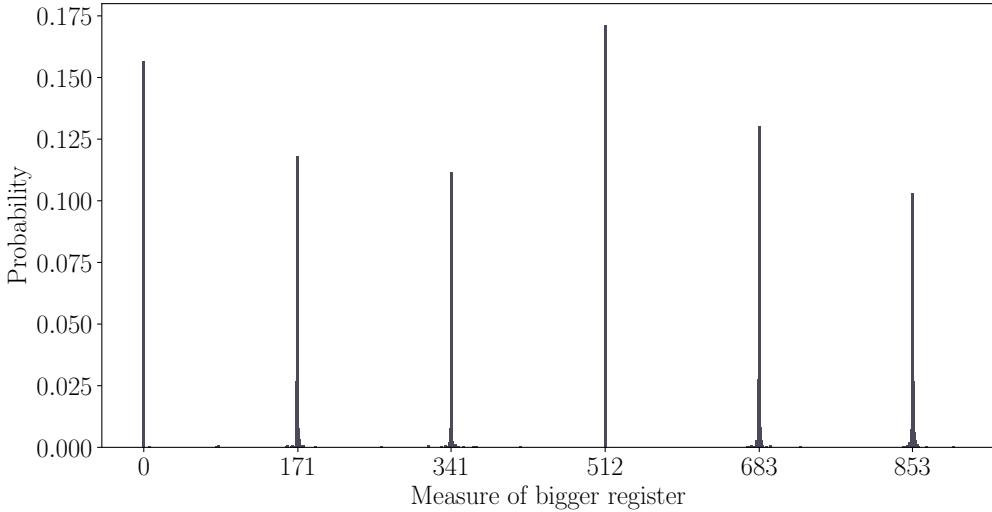


Figure 5.9: Zero errors, $N = 27$, $y = 8$, $r = 16$

This set finally allows the definition of Shor's success rate as:

$$s(n, r) = \sum_{c_i^* \text{ useful}} p(c_i^*, n, r) \quad (5.9)$$

where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the rounding down and up respectively, and the probability $p(c_i^*, n, r)$ is simply the probability of measuring the state c_i^* given the circuit.

5.3 Considerations on the error model

The considered error pattern for the simulations for $N = 27$ and $N = 63$ is again composed of discrete errors: after each operational time step, each bit can experience with 0.5 probability a bit flip $X |\psi\rangle = \alpha |1\rangle + \beta |0\rangle$ or a phase flip $Z |\psi\rangle = \alpha |0\rangle - \beta |1\rangle$. The situation in which they occur simultaneously is not contemplated. As described in the article [DFH06], this error model represents the most common set-up used in the quantum error correction analysis, although it oversimplifies the reality of things in several ways:

- The errors are uncorrelated and equiprobable, but real architectures can be more

vulnerable to one error type than another;

- the model does not examine systematic errors caused by inaccurate gate design or external influence;
- memory errors such as incorrect recall or complete loss of information for a specific qubit and also decoherence errors are not considered.

With this in mind, this error set-up is chosen because in general, more realistic continuous errors are a linear combination of discrete ones considered. To better understand the impact of error introduction on the circuit, the state prior to the modular exponentiation is examined:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_{master} + |1\rangle_{master}) \sum_{m=0}^{2^{2n}-1} \alpha_m |m\rangle_{computer}. \quad (5.10)$$

After the controlled multiplication gates, the state of the second register changes:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle \sum_{m=0}^{2^{2n}-1} \alpha_m |m\rangle + \frac{1}{\sqrt{2}}|1\rangle \sum_{m=0}^{2^{2n}-1} \beta_m |m\rangle. \quad (5.11)$$

And again the IQFT with Hadamards and Phase Gates take the state to:

$$|\psi\rangle = \frac{1}{2}|0\rangle \sum_{m=0}^{2^{2n}-1} (\alpha_m + e^{i\theta}\beta_m) |m\rangle + \frac{1}{2}|1\rangle \sum_{m=0}^{2^{2n}-1} (\alpha_m - e^{i\theta}\beta_m) |m\rangle. \quad (5.12)$$

The probability of measuring 0 or 1 is:

$$P\left(\frac{1}{2} \mp \frac{1}{2}\right) = \frac{1}{2} \pm \frac{1}{4} \sum_{m=0}^{2^{2n}-1} (e^{i\theta}\alpha_m^*\beta_m + e^{-i\theta}\alpha_m\beta_m^*) \quad (5.13)$$

Imposing the normalisation of the coefficients

$$\sum_{m=0}^{2^{2n}-1} |\alpha_m|^2 = \sum_{m=0}^{2^{2n}-1} |\beta_m|^2 = 1, \quad (5.14)$$

the errors cause the summation of the previous formula to 0 resulting in uniform probability $p = (0.5)^{2n}$ for all the states.

5.4 Factoring 27

The simulation for $N = 27$ is now considered; the correct factorisation is obviously 3×9 and the parameters are chosen accordingly to the previous Table 5.18:

N	y	n	circuit n	$\#_{err}$
27	8	5	15	55
63	31	6	18	78
77	77	7	21	105
247	27	8	24	171

Figure 5.10: Simulation parameters for $r = 6$

The process searches for states belonging to the set:

$$c* = [0, 171, 341, 512, 683, 853]. \quad (5.15)$$

The circuit is run 50 times for each error addition, for a total of 2750 circuits implemented (50 repetitions multiplied by 55 error additions). In each of the 50 repetitions, the number of errors is fixed, but their positions and types are randomised. Naturally, the potential number of combinations far exceeds 50; however, the choice of `runs= 50` is primarily due to restricting the computational time. With this parameter setting the program's execution lasted for about 75h. A higher number of repetitions would have been possible but quite expensive.

First, the factorisation is checked. As mentioned before, this is not an accurate witness of the algorithm's performance in the presence of errors. It is evident that the simulation always finds compatible factors with the expected ones although in the case of this co-prime number, it never observes the factor 3. Output $[0, 0]$ results in measuring the state $|00000\rangle$, which corresponds to zero phase. Looking at the values of measured states the idea of how much the algorithm is influenced by errors becomes clear. Figure 5.12 shows all the simulations from 0 to 55 errors superposed: and although the peaks are still clear, noise is preponderant. If plots for the minimum and the maximum $\#_{err}$ are isolated, the difference stands out: for zero errors (Figure 5.13) the ideal situation is found, for 55 errors (Figure 5.14) noise obscures the simulation's data. With an even bigger number

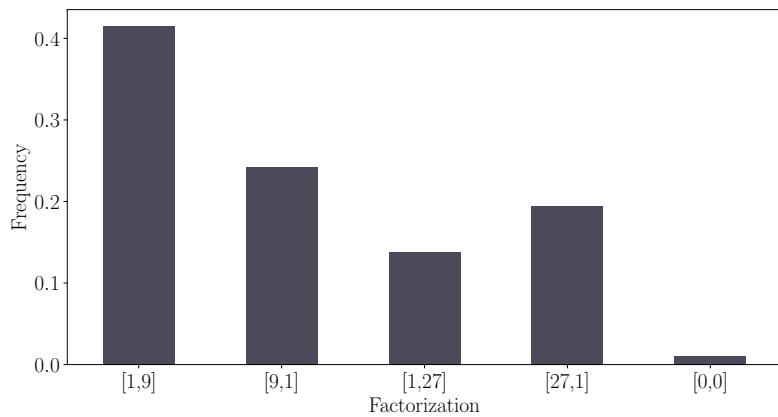


Figure 5.11: Simulation for the factorisation of $N = 27$, $p = 3$, $q = 9$ with errors in random positions

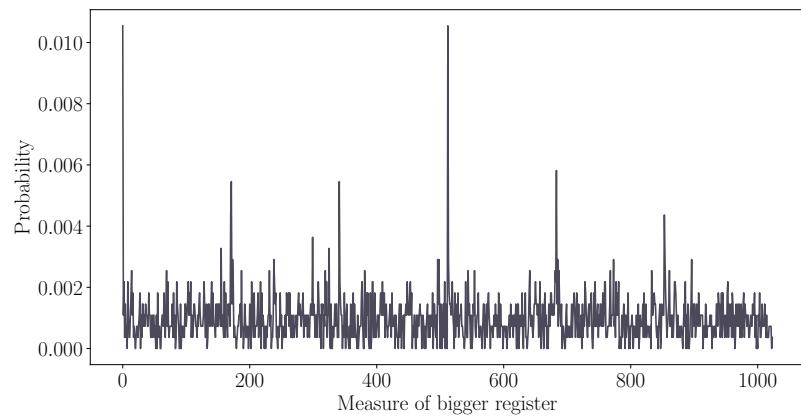


Figure 5.12: Useful states of the total simulation

of repetitions, the second plot would have become a uniform distribution over all the base's states. Consequently, the measure would become a random sampling destroying completely the constructive interference. The left plot with no unitary errors has a

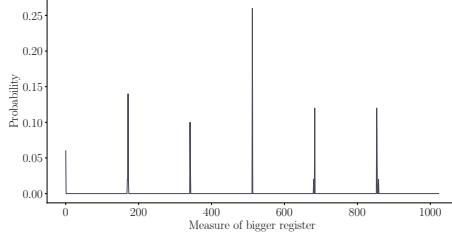


Figure 5.13: Measured output for 0 errors

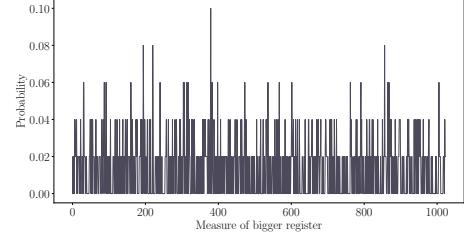


Figure 5.14: Measured output for 55 errors

degree of noise of the 0.08% and the one on the right shows about the 92% of non-useful states values in the outcome. Noise increases exponentially leading the system from a sharpened distribution to a uniform one after less than 15 random rotations: As regards

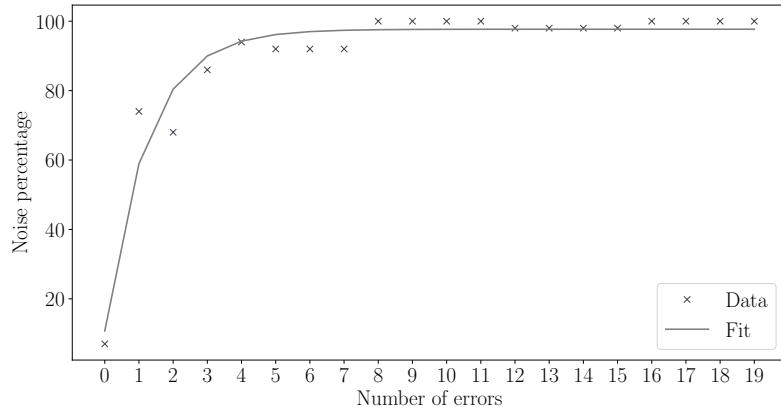


Figure 5.15: Percentage of noisy data per error number zoomed on the first 20 errors

, the distribution of periods for the total simulation shown in Figure 5.16 indicates that the correct value $r = 6$ is as frequent, if not less, than all the other values in the period domain. Again a symptom of noise and inefficiency. Finally the most important result: for an increasing number of errors, the frequencies of the useful states c^* are reported in Figure 5.17. The horizontal line highlights the value of uniform probability $1/2^{10} = 0.001$. A single error reduces the efficiency of the algorithm by half, and for fewer than ten errors, the measured state becomes essentially random.

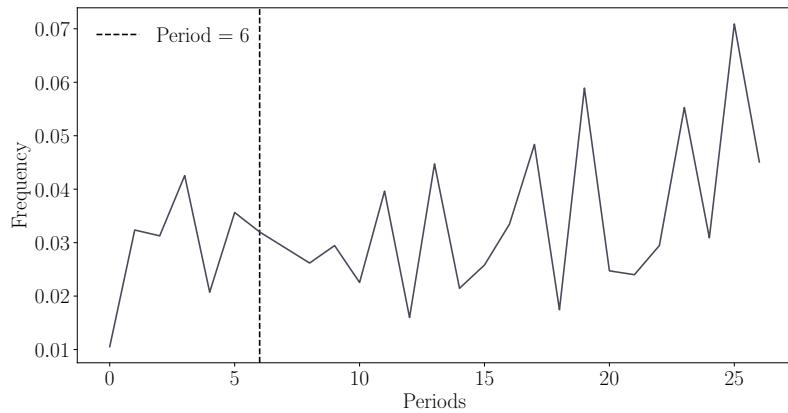


Figure 5.16: Periods distribution in the simulation with the expected value pointed out by the dashed line

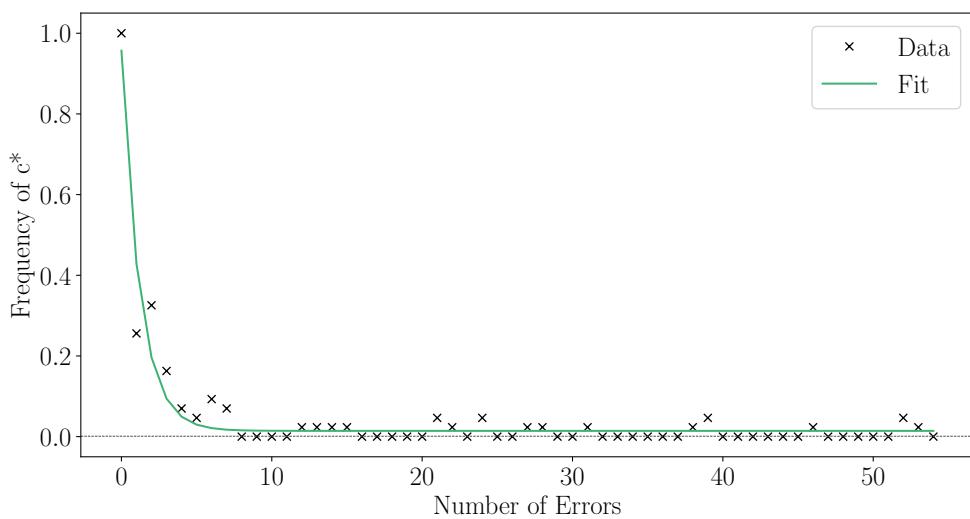


Figure 5.17: $N = 27$, $y = 8$, $n = 5$, reps= 50

5.5 Factoring 63

Next, the focus is shifted to $N = 63$ and the relative parameters are in Table 5.18 as before. This time the simulation would have been impossible without using a cluster to

N	y	n	circuit n	$\#_{err}$
27	8	5	15	55
63	31	6	18	78
77	77	7	21	105
247	27	8	24	171

Figure 5.18: Simulation parameters for $r = 6$

execute the file; with this tool, the procedure was split on 72 nodes, each of which lasted 2.5h (a week worth of computational time). Once again, each data point on the plot corresponds to the frequency on 50 runs and the set of useful values is the following:

$$c^* = [0, 682, 683, 1365, 1366, 2048, 2730, 2731, 3413, 3414]. \quad (5.16)$$

Considerations on the noise are comparable to the one for $N = 27$ have not been included. After < 10 errors the random output is reached and results approach an asymptotic behaviour at $1/2^{12} = 0.0002$.

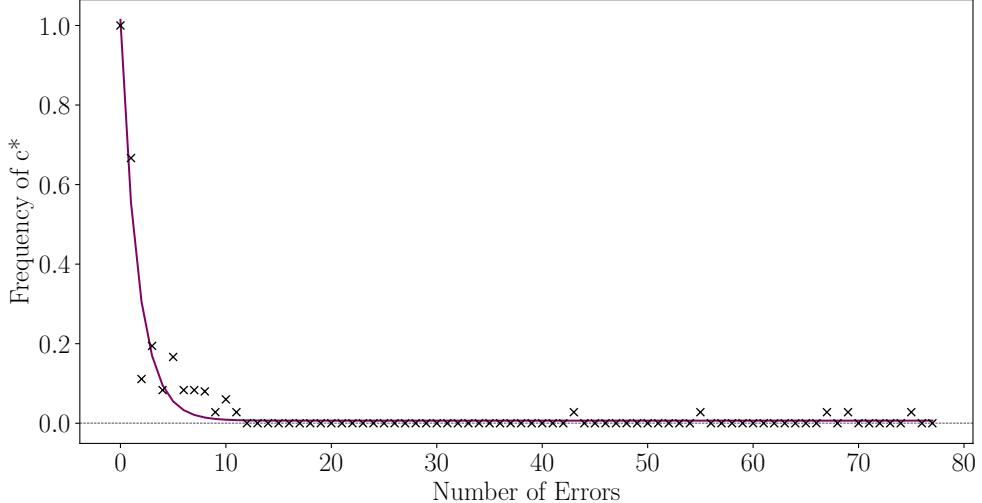


Figure 5.19: Factorisation of $N = 63$, $y = 31$, $n = 6$, reps= 50

5.6 Fitting the curves

Some final considerations about the exponential decay shown by all three simulations are worth making. The question to be answered is "How many errors does the Shor algorithm tolerate before becoming inefficient?". First of all, a linear fit is performed for the three decays in Figure 5.6, Figure 5.17 and Figure 5.19. The analysis comprehends just points up to 8 errors approximately before the asymptotic behaviour. Data are normalised in order to have a frequency of 1 with zero errors. The fitting function used here is:

$$y = a \cdot x + b \quad (5.17)$$

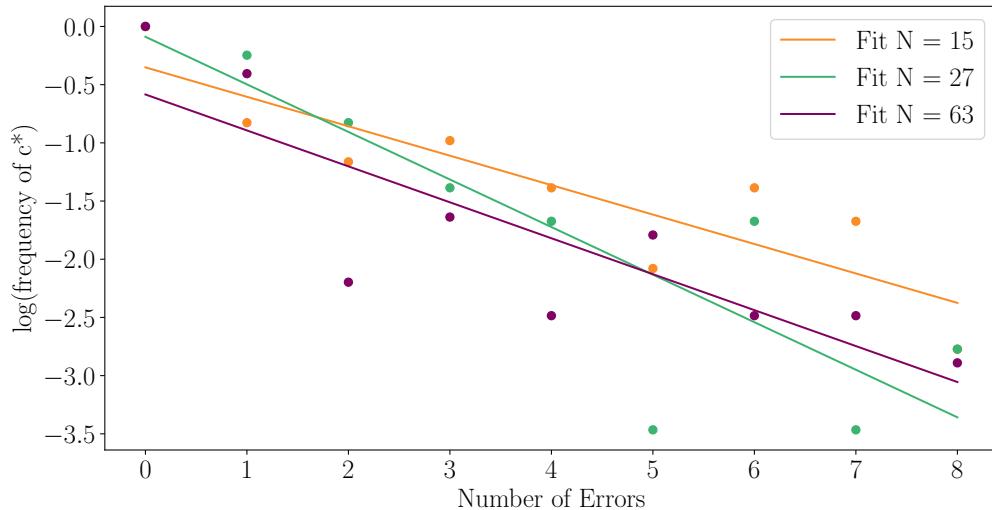


Figure 5.20: Linear fit on the logarithm of the data

Being that Figure 5.20 is not really meaningful and lines do not exhibit any common behaviour, an exponential fit is performed to try to extract more useful pieces of information:

$$y = a \cdot e^{-b \cdot x} + c. \quad (5.18)$$

The fitting function corresponds to the following behaviour, typical of a negative exponential:

$$\Pr(n) = \Pr(0)e^{-\frac{n}{n_0}}. \quad (5.19)$$

Here, $\Pr(i)$ is the probability of success with i errors, and n/n_0 is the *damping factor*.

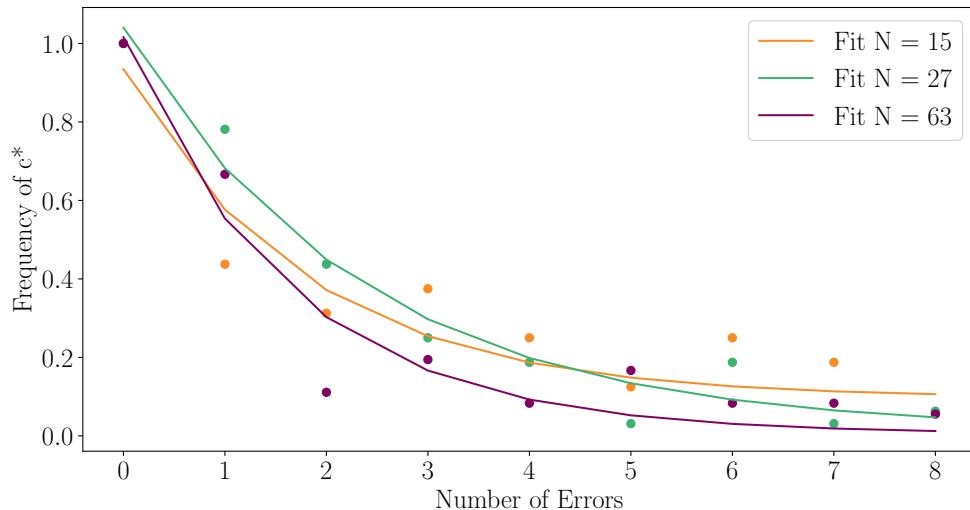


Figure 5.21: Exponential fit. Note that the three decays approach an asymptote for the corresponding value of $1/2^{2n}$.

From the fitting procedure, the maximum number of tolerated errors $n_0 = 1/b$ is extracted: The statement is therefore that the algorithm can still work, although with a

N	b	n_0
15	0.56	1.81
27	0.82	1.22
63	0.61	1.63

Table 5.1: Fitting parameters and maximum number of errors tolerated

lowered efficiency, when the number of errors goes from 1 up to a maximum of 2. On the other hand, regarding the number of unitary rotations leading the algorithm to a *total* loss of efficiency, it is now demonstrated that this number is almost linearly proportional to the size of the control register. Plotting the number of errors corresponding to the first value of $\text{Pr}(i) = 0$ versus two times the number of qubits n , Figure 5.22 is the result. The plot indicates that as the circuit size increases, a greater number of errors is required

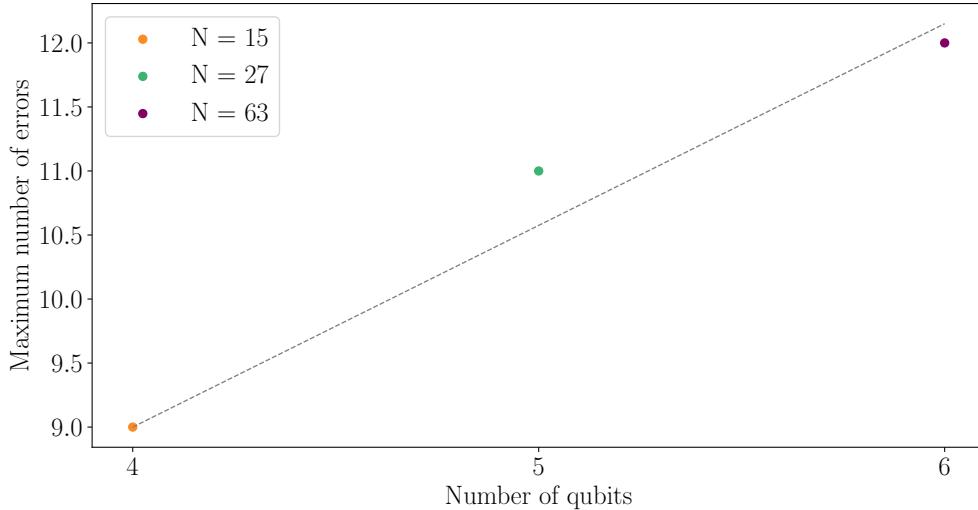


Figure 5.22: Maximum error number increasing with the system size

to cancel the advantage of the quantum algorithm and reach a point where constructive interference is lost due to the presence of errors.

Conclusions

Summarising, this thesis has investigated the structure and entanglement dynamics of Shor’s algorithm, focusing on the effects of errors on factorisation. Several key findings have emerged through the analysis of the algorithm’s performance and vulnerabilities. Firstly, it has been observed that Shor’s algorithm demonstrates good resilience in tackling the prime factorisation problem when the number of errors is below two. The errors considered were just a subset of the possible ones, considering the wide range of inaccuracies to which real quantum computers are subjected, but nevertheless have shown very interesting behaviours.

Secondly, it has been demonstrated that the algorithm exhibits a total loss of efficiency with a number of errors approximately equal to $2 \cdot n$ (two times the number of qubits corresponding to the number to factorise). Although the available data in this work were limited due to the increasing computational power needed with increasing N , the outcomes produced satisfactory results. The QFT revealed itself as a valid tool in the study of the algorithm’s performance, being the more influential part of the circuit because of its gate structure.

The proposed study has clarified the necessary level of accuracy for a successful implementation of the algorithm on real quantum devices. Additionally, a threshold of efficiency is established for error mitigation techniques employed in the context of the Shor algorithm. Further research would comprehend conducting larger factorisations and exploring techniques for reducing the number of necessary qubits, such as alternative QFT structures. Moreover, investigating how much quantum error correcting codes, like the Shor code, can restore efficiency to ideal levels.

Appendices

A The Qiskit Library

A.1 What is Qiskit

The simulative part of this work completely relies on the possibility to construct and customise quantum circuits and extract information from them. Up until here, all the analyses have been performed as classical simulations of quantum systems on standard computers. However, future work could for sure extend these findings by executing the written codes on real quantum computers, integrating the previous analysis with the more complex error framework of a real quantum device. As mentioned, the platform chosen is Qiskit: an open-source software development kit (SDK) launched in 2017 that provides tools to create algorithms and simulate them, both locally and on the cloud (IBM Quantum Experience). Qiskit relies on the programming language Python and comes with a rich library of mathematical and visual tools. This makes it an efficient environment suitable for the solution of low and medium-complexity tasks.

A.2 Qiskit components

Four different elements make the IBM platform effective:

1. *Qiskit Aer*: this element provides simulating tools and realistic noise models, executing the program on a backend (either quantum hardware or a simulator);
2. *Qiskit Ignis*: contains tools for error mitigation and error correction;
3. *Qiskit Aqua*: a dedicated element with applicative modules for Machine Learning, Finance Chemistry etc.;
4. *Qiskit Terra*: contains the building blocks for creating a quantum circuit. Gates, measurement processes, unitary gates etc. are implemented and available as functions. In the code, one specifies the registers, gate types, eventual phases and on which qubit or qubits apply them. An example of circuit code follows, corresponding to Item 4.

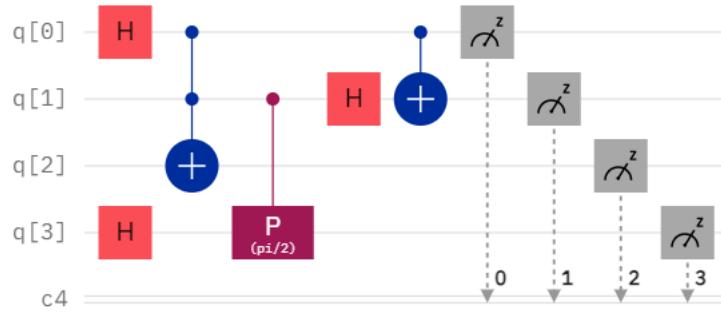
```
qreg q[4];
creg c[4];
h q[0];
h q[3];
ccx q[0], q[1], q[2];
```

```

cp(pi/2) q[1], q[3];
h q[1];
cx q[0], q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];

```

In the corresponding circuit $q[i]$ defines the i^{th} qubit and $c4$ line indicates the classic register on which the data are collected during the simulation.



B Further Plots

All distance plots are generated with $n = 4$ and $\mathcal{P} = 1$ of having an error. Oscillations are random and depend on the machine's precision.

B.1 Case 2: single error before the swapping procedure

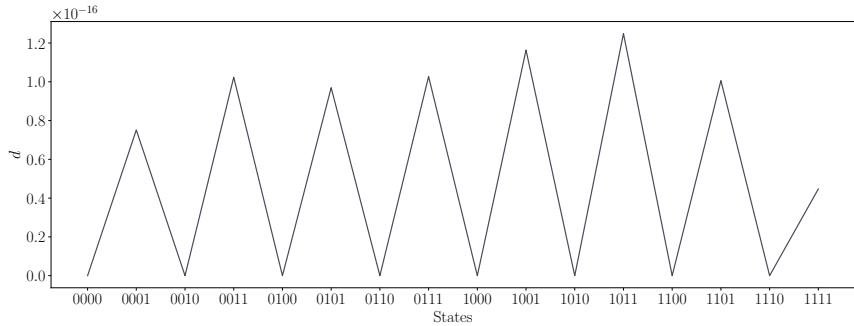


Figure 23: X -error on qubit= 0

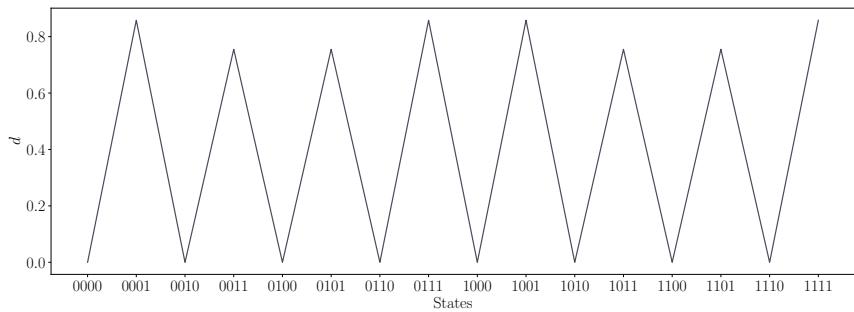


Figure 24: X -error on qubit= 1

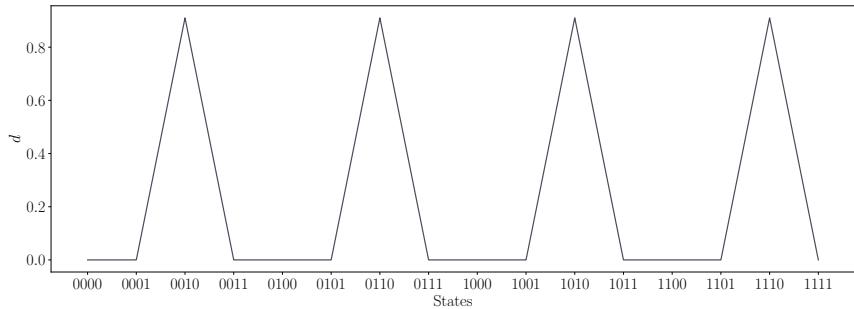


Figure 25: X -error on qubit= 2

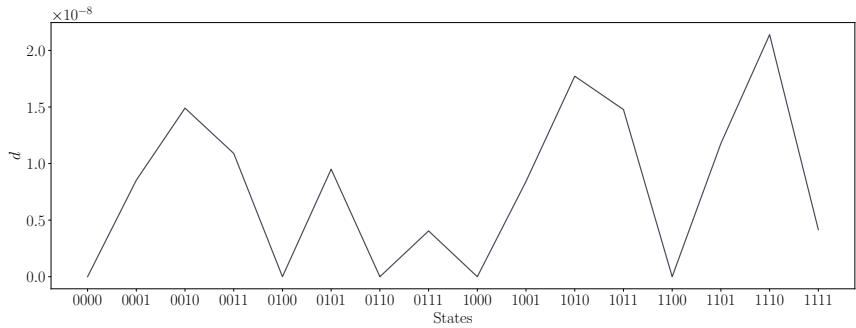


Figure 26: X -error on qubit= 3

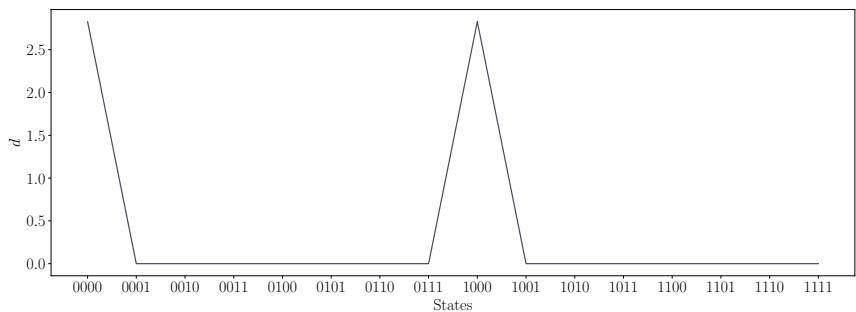


Figure 27: Z -error on qubit= 0

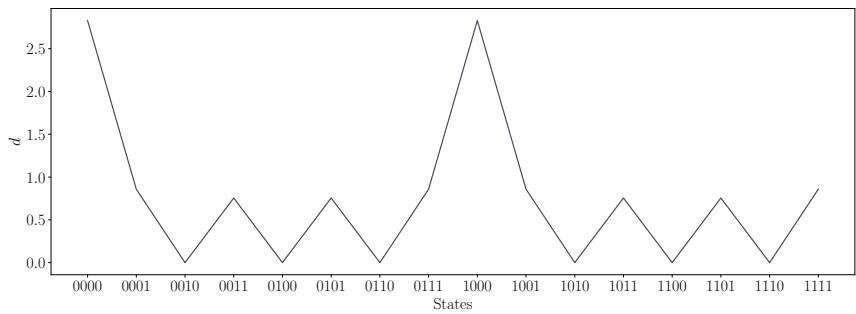


Figure 28: Z -error on qubit= 1

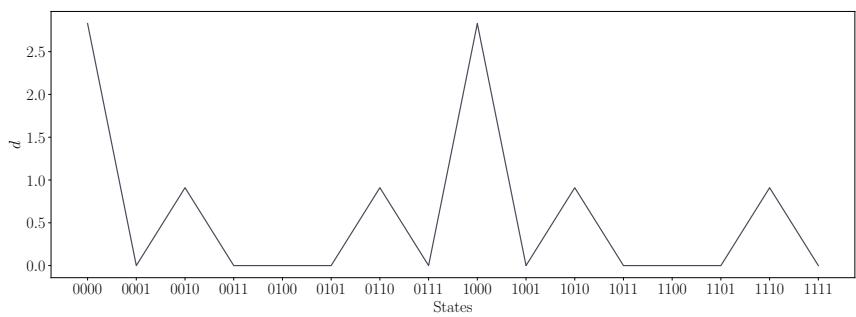


Figure 29: Z -error on qubit= 2

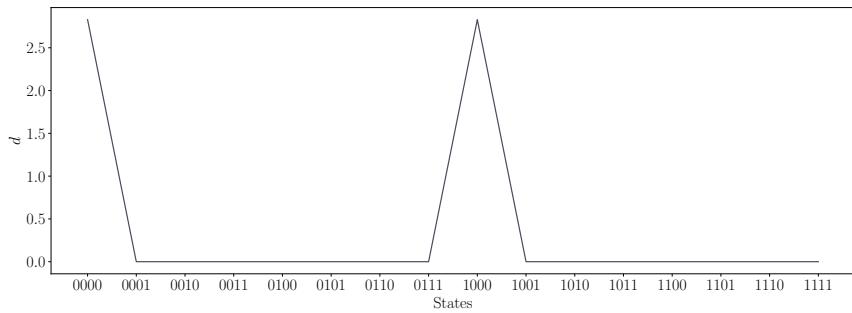


Figure 30: Z -error on qubit= 3

B.2 Case 3 - single error after the swapping procedure

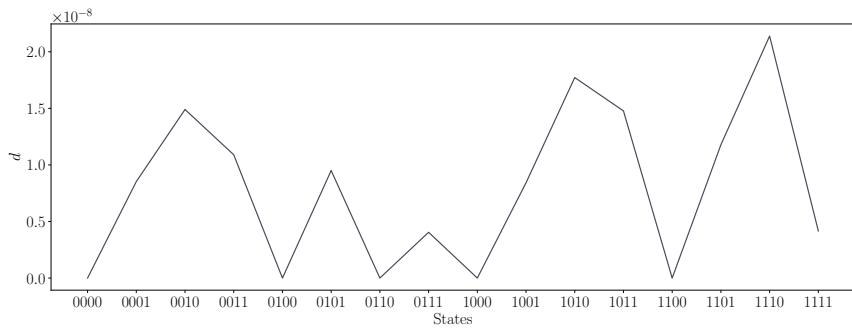


Figure 31: X -error on qubit= 0

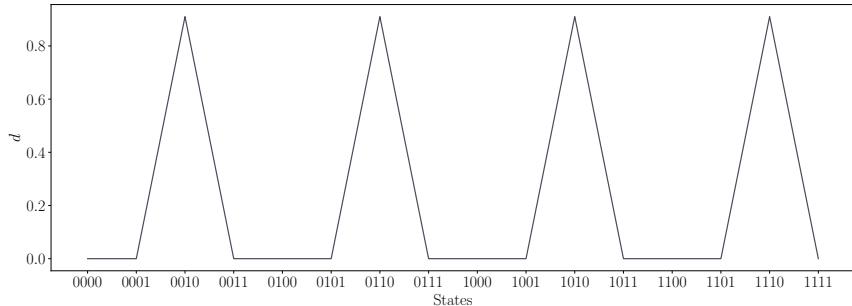


Figure 32: X -error on qubit= 1

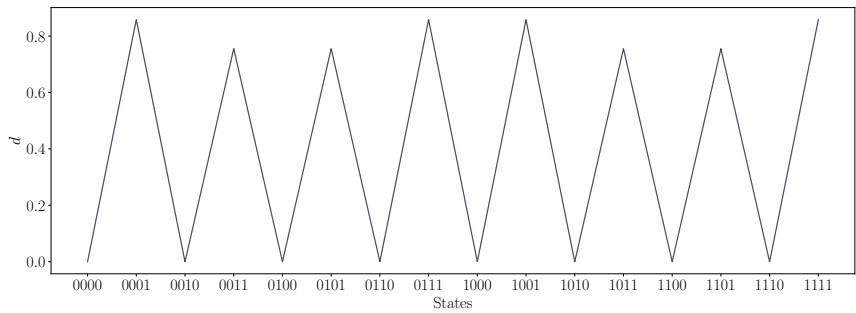


Figure 33: X -error on qubit= 2

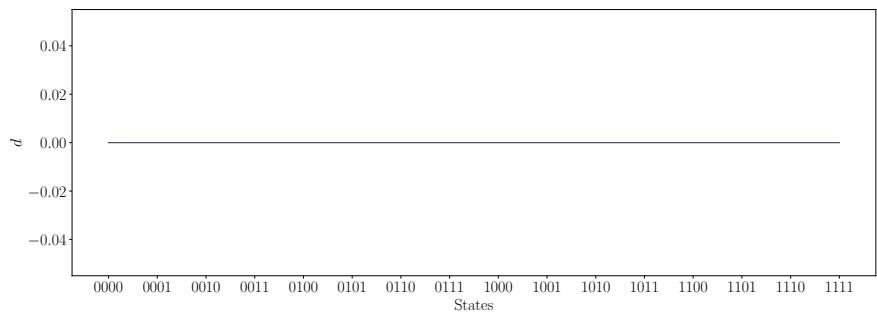


Figure 34: X -error on qubit= 3

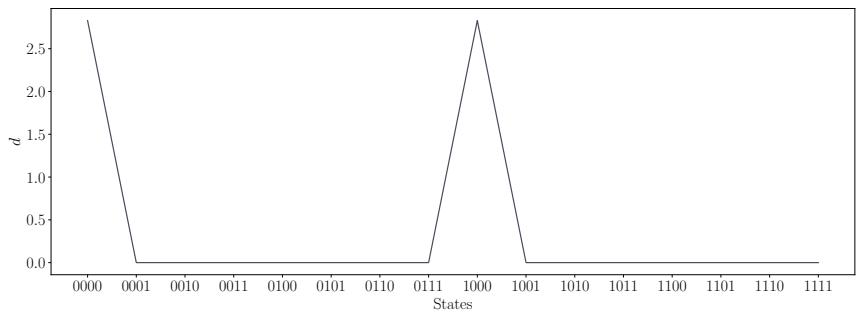


Figure 35: Z -error on qubit= 0

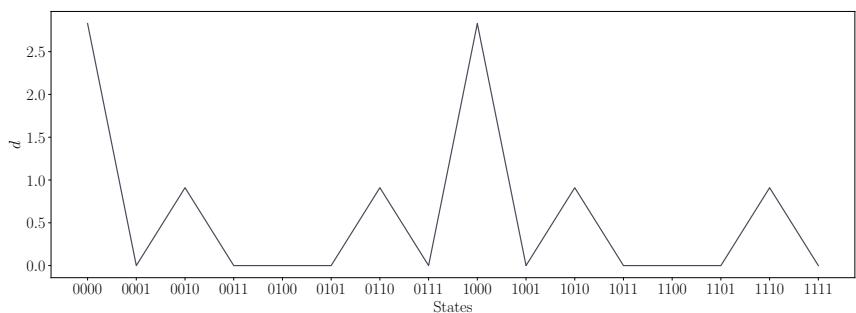


Figure 36: Z -error on qubit= 1

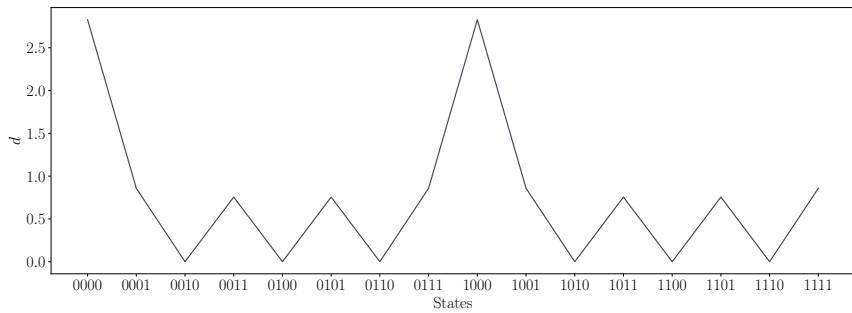


Figure 37: Z -error on qubit= 2

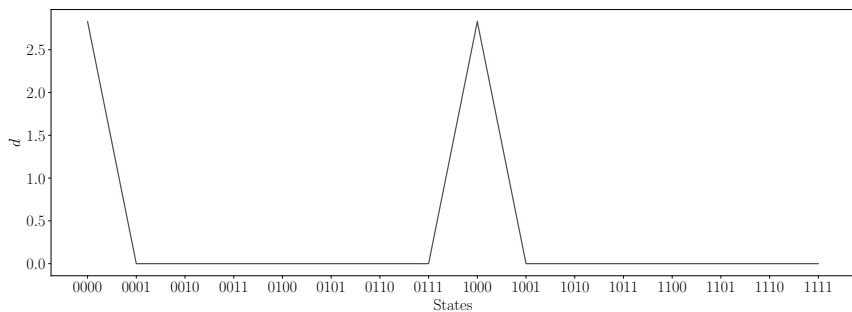


Figure 38: Z -error on qubit= 3

B.3 Case 4 - one error before each Hadamard

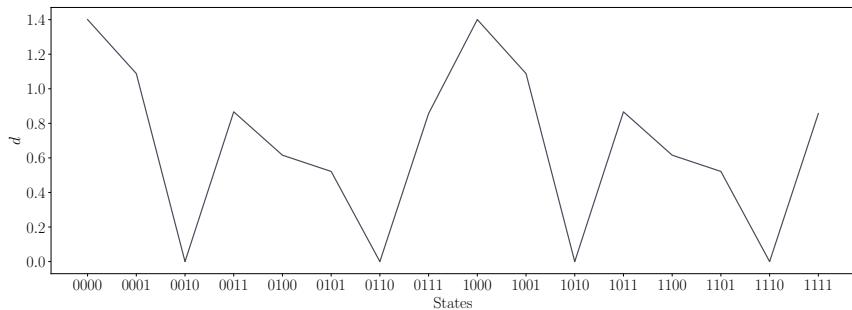


Figure 39: X -error on qubit= 0

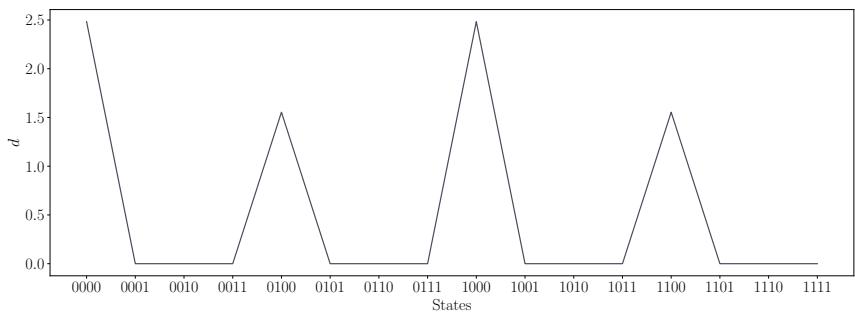


Figure 40: X -error on qubit= 1

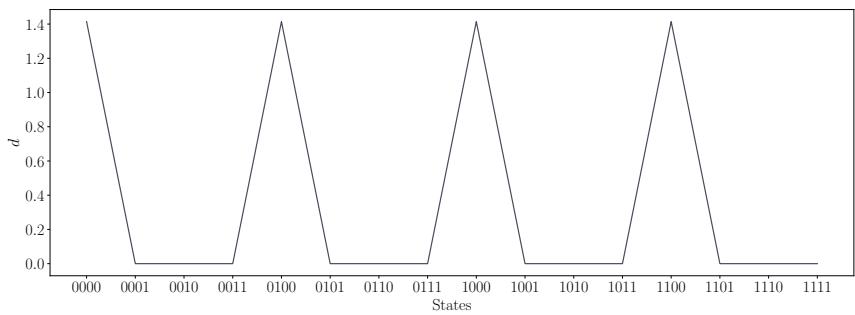


Figure 41: X -error on qubit= 2

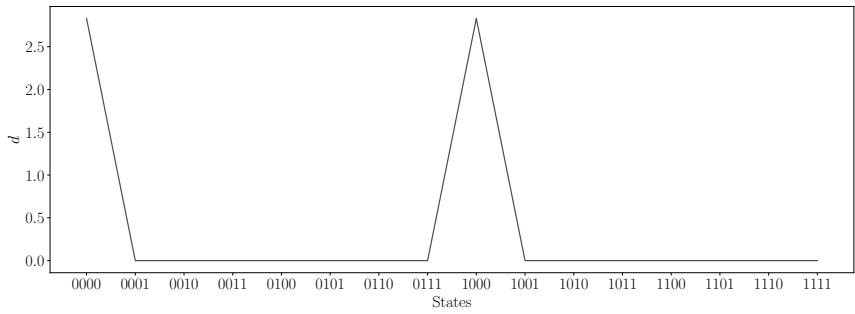


Figure 42: X -error on qubit= 3

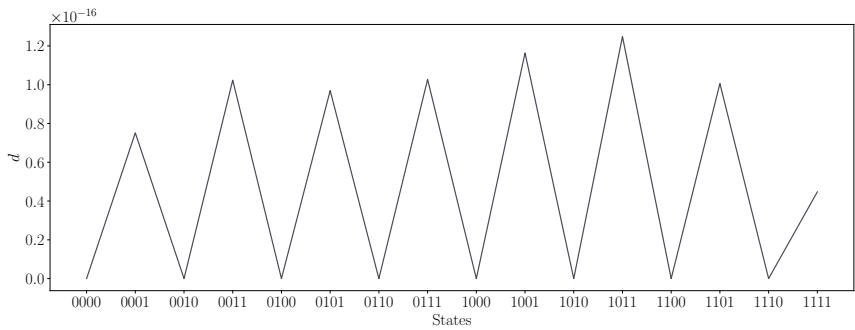


Figure 43: Z -error on qubit= 0

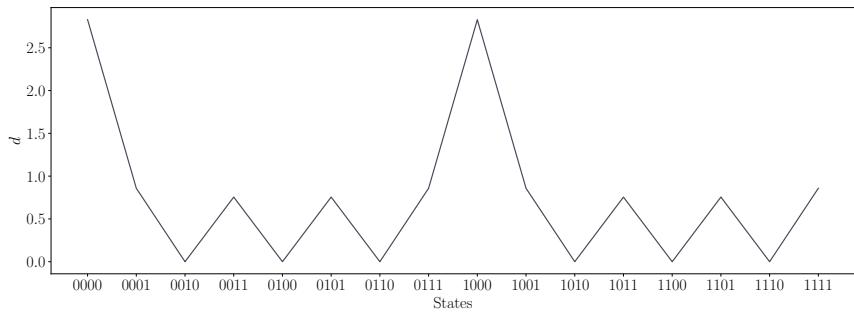


Figure 44: Z -error on qubit= 1

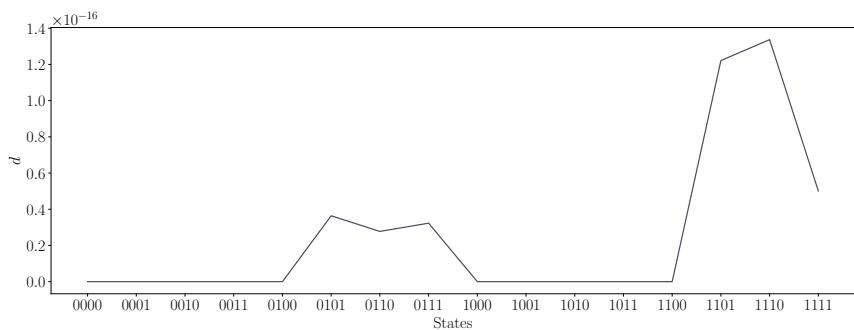


Figure 45: Z -error on qubit= 2

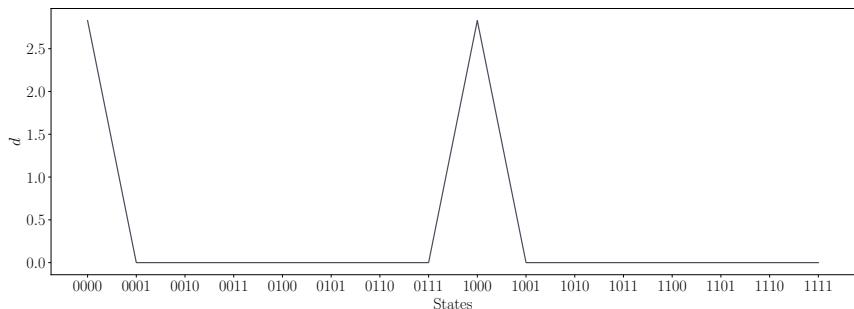


Figure 46: Z -error on qubit= 3

B.4 Case 5 - one error after each Hadamard

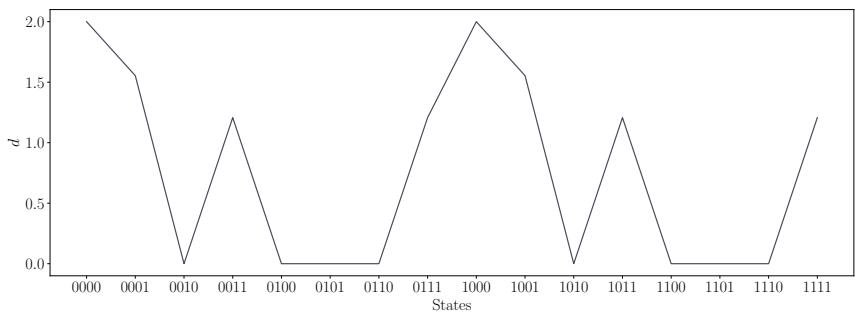


Figure 47: X -error on qubit= 0

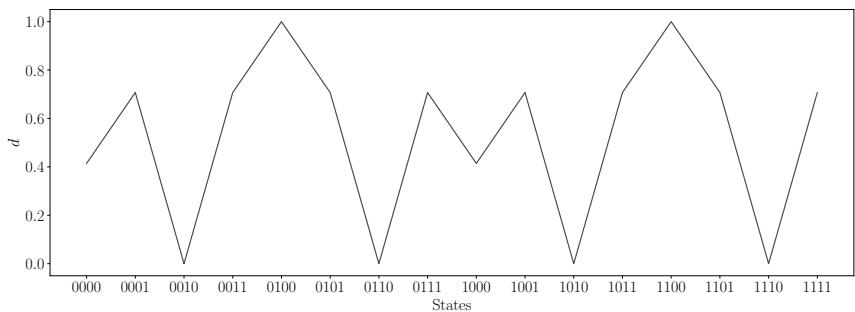


Figure 48: X -error on qubit= 1

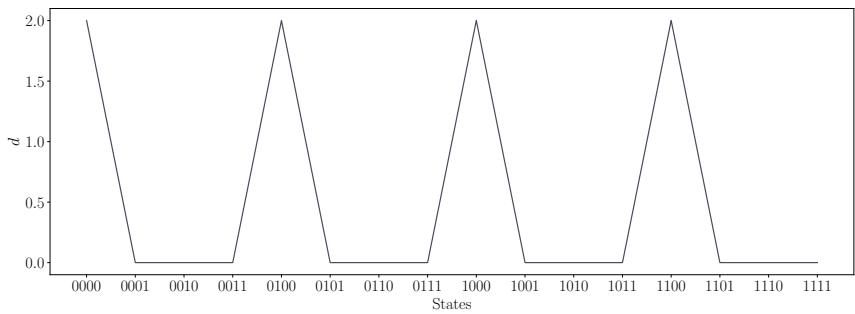


Figure 49: X -error on qubit= 2

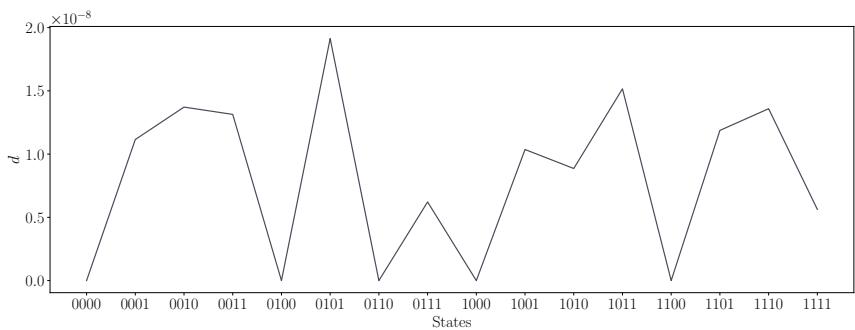


Figure 50: X -error on qubit= 3

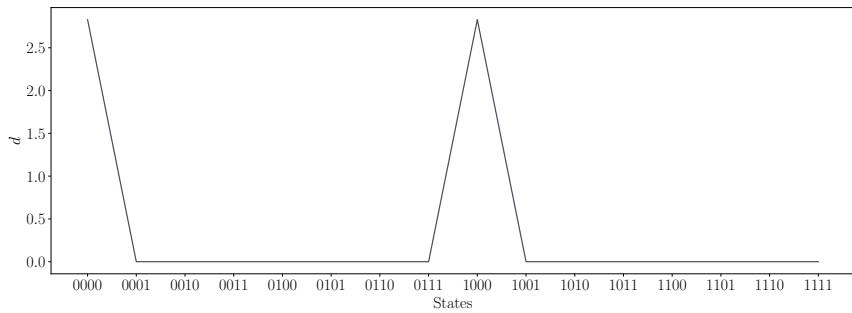


Figure 51: Z -error on qubit= 0

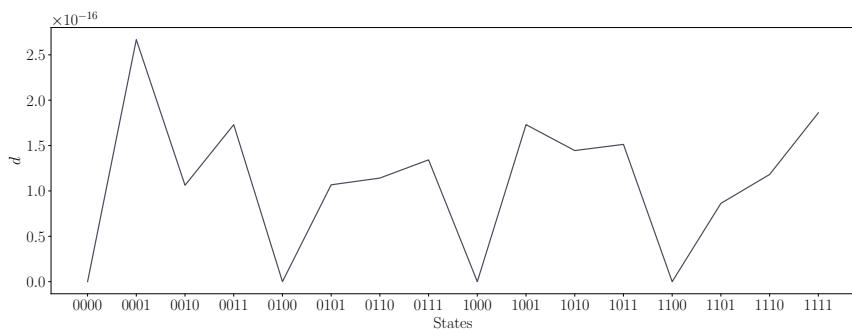


Figure 52: Z -error on qubit= 1

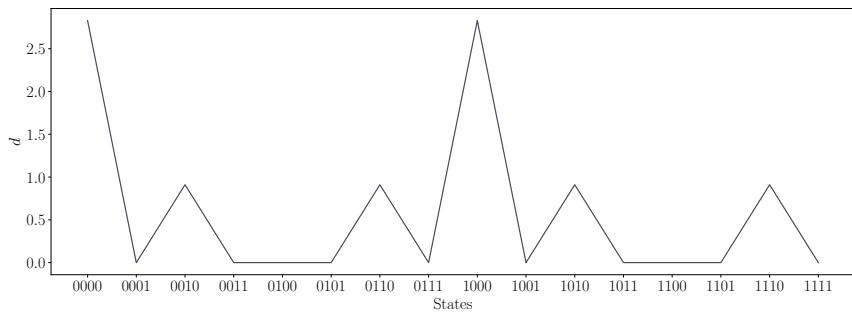


Figure 53: Z -error on qubit= 2

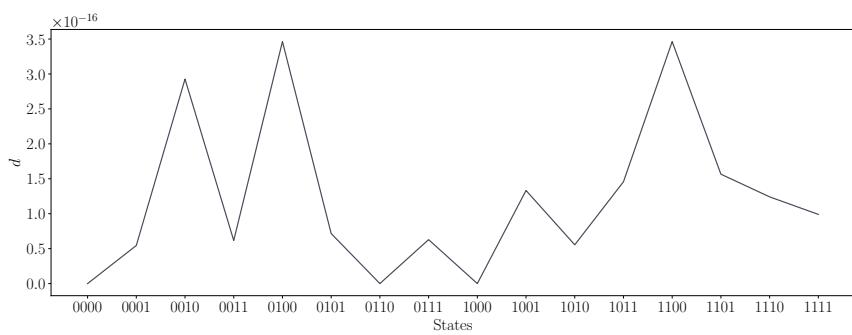


Figure 54: Z -error on qubit= 3

B.5 Average behaviour

If the distance is mediated over all possible X -errors the result in Figure 55 is similar to the mean distance for Z -errors in Figure 56. The symmetric behaviour is clear, and $\langle d \rangle_{phase}$ is approximately twice the mean distance $\langle d \rangle_{bit}$. Clearly, there are states more affected by errors than others.

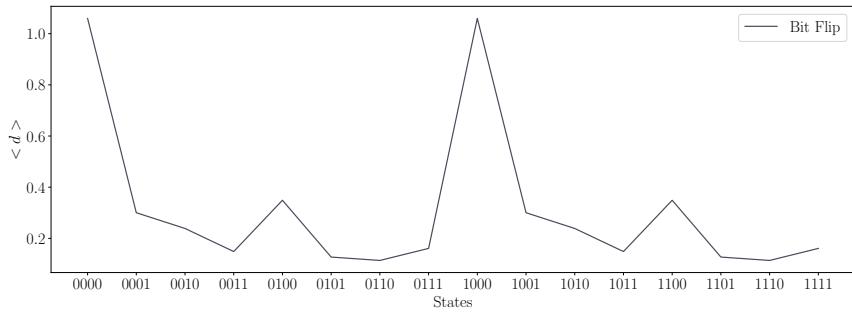


Figure 55: Mean distance for bit flips

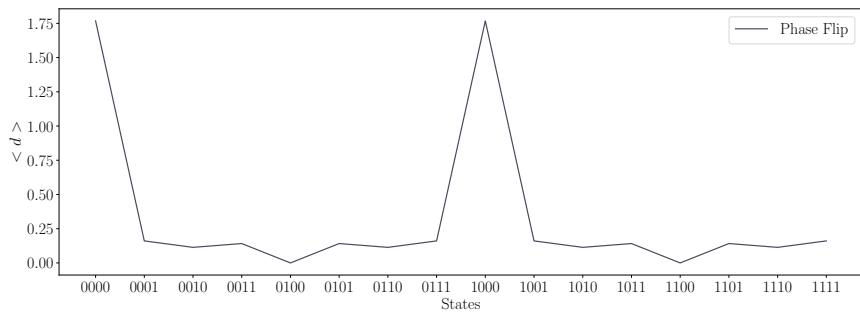


Figure 56: Mean distance for phase flips

C Notebooks Index

The following list contains the names of all the Python Notebooks used in the thesis with the corresponding generated images. Each one of them is available online on the GitHub profile @gjuls.

Chapter 2 - Shor's Algorithm

- `shor_basic.ipynb`: ??, Figure 2.3, Figure 2.4

Chapter 3 - Quantum Entanglement

- `shor_ent_registers.ipynb`: Figure 3.1
- `shor_ent_partitions.ipynb`: Figure 3.3, Figure 3.4
- `shor_traslation.ipynb`: Figure 3.6
- `qft_ent_partitionsSINGLE.ipynb`: Figure 3.7

Chapter 4 - Error Analysis in the QFT

- `QFT_errors.ipynb`: Figure 4.1, Figure 4.2, Figure 4.3
- `QFT_mean_state.ipynb`: Figure 4.4, Figure 4.5, Figure 4.6, Figure 4.7, Figure 4.8
- `QFT_return_prob.ipynb`: Figure 4.9
- `QFT_return_prob_cases.ipynb`: Figure 4.12, Figure 4.13
- `QFT_return_prob_states.ipynb`: Figure 4.14, Figure 4.15
- `QFT_return_prob_cases_many_errors.ipynb`: Figure 4.16

- **QFT_return_probability_heatmap.ipynb**: Section 4.8, Figure 4.19, Figure 4.21, Figure 4.20, Figure 4.22, Figure 4.27, Figure 4.26
- **QFT_return_probability_heatmapBIS.ipynb**: Section 4.8, Figure 4.24, Figure 4.25
- **Attenuation.ipynb**: Figure 4.28
- **AttenuationBIS.ipynb**: Figure 4.29

Chapter 5 Noisy-QFT for Factoring Numbers

- **shor_noisy_multiple_errors.ipynb**: Figure 5.1, Figure 5.2
- **Errors.ipynb**: Figure 5.4, Figure 5.5
- **FINAL_dec.ipynb**: Figure 5.6, Figure 5.17, Figure 5.19
- **FINAL_prj.ipynb**: Figure 5.8, Figure 5.9
- **FINAL_prj_graphics.ipynb**: Figure 5.11, Figure 5.12, Figure 5.13, Figure 5.14, Figure 5.15, Figure 5.16, Figure 5.20, Figure 5.21, Figure 5.22

Appendix 2 - Further Plots

- **QFT_mean_state.ipynb**: all images from Figure 23 to Figure 54
- **QFT_mean_error.ipynb**: Figure 55, Figure 56

Bibliography

- [AdOB22] Carolina Allende, André Fonseca de Oliveira, and Efrain Buksman. Characterizing errors for Quantum Fourier Transform on IBM Q. *Rev. Mex. Fis.*, 68(3):031402, 2022.
- [ANU19] Poornima Ardhayamath, N. M. Naghabhushana, and Rohitha Ujjinimatad. Quantum Factorization of Integers 21 and 91 using Shor’s Algorithm. 2019.
- [ASK19] Mirko Amico, Zain Saleem, and M. Kumph. Experimental study of Shor’s factoring algorithm using the IBM Q Experience. *Physical Review A*, 2019.
- [BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing*, 26(5):1510–1523, oct 1997.
- [Bea02] Stephane Beauregard. Circuit for Shor’s Algorithm using $2n+3$ qubits. 2002.
- [Ben79] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. 2 1979.
- [BFD10] Floreanini R. Benatti F., Fannes M. and Petritis D. *Quantum Information, Computation and Cryptography: An Introductory Survey of Theory, Technology and Experiments*. Springer-Verlag, 2010.
- [CCL14] Zhengjun Cao, Zhenfu Cao, and Lihua Liu. Remarks on Quantum Modular Exponentiation and Some Experimental Demonstrations of Shor’s Algorithm. *IACR Cryptol. ePrint Arch.*, 2014:828, 2014.
- [Cop02] D. Coppersmith. An approximate Fourier transform useful in quantum factoring, 2002.

- [CSW22] Jielun Chen, E. M. Stoudenmire, and Steven R. White. The Quantum Fourier Transform has small Entanglement. 2022.
- [CT65] James W. Cooley and John W. Tukey. An algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90):297–301, 1965.
- [DB14] Nikesh S. Dattani and Nathaniel Bryans. Quantum factorization of 56153 with only 4 qubits. *ArXiv*, abs/1411.6758, 2014.
- [Deu85] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400:117 – 97, 1985.
- [DFH06] Simon J. Devitt, Austin G. Fowler, and Lloyd C. L. Hollenberg. Robustness of Shor’s algorithm, 2006.
- [DiV00] David P. DiVincenzo. The Physical Implementation of Quantum Computation. *Fortschritte der Physik*, 48(9-11):771–783, sep 2000.
- [EF04] Bryan Eastin and Steven T. Flammia. Q-circuit Tutorial, 2004.
- [EJ96] Artur Ekert and Richard Jozsa. Quantum computation and Shor’s factoring algorithm. *Rev. Mod. Phys.*, 68:733–753, Jul 1996.
- [EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, May 1935.
- [Fey82] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 1982.
- [HV08] C. Ray Hill and George F. Viamontes. Operator Imprecision and Scaling of Shor’s Algorithm, 2008.
- [HW97] Scott Hill and William K. Wootters. Entanglement of a Pair of Quantum Bits. *Physical Review Letters*, 78(26):5022–5025, jun 1997.
- [JL03] Richard Jozsa and Noah Linden. On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2036):2011–2032, aug 2003.
- [KM04] Vivien M. Kendon and William J. Munro. Entanglement and its Role in Shor’s Algorithm. 2004.
- [Li11] Yang D. Li. BQP and PPAD, 2011.

- [Mos16] Gianni Mossi. Entanglement Dynamics in Shor’s Algorithm. 2016.
- [NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [Pre12] John Preskill. Quantum Computing and the Entanglement frontier, 2012.
- [PV05] Martin B. Plenio and S. Virmani. An introduction to Entanglement Measures. 2005.
- [Rof19] Joschka Roffe. Quantum error correction: an introductory guide. *Contemporary Physics*, 60(3):226–245, jul 2019.
- [Sho94] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, oct 1994.
- [Sho95] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.
- [Sim94] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1994.
- [WGFF20] Zhong-Wei Wang, Xirui Gou, Ruiqin Fu, and Zhixuan Fu. Quantum Fourier Transform and Its Application in Shor’s Algorithm. 2020.
- [Yan07] Noson S. Yanofsky. An Introduction to Quantum Computing, 2007.
- [YS22] Farha Yasmin and Jan Sperling. Entanglement-assisted quantum speedup: Beating local quantum speed limits, 2022.
- [YSJL04] Anocha Yimsiriwattana and Jr. Samuel J. Lomonaco. Distributed quantum computing: a distributed Shor algorithm. In *SPIE Defense + Commercial Sensing*, 2004.