Implementation of the Grover algorithm on IBM NISQ platform

EL KADERI Yunos

University master work
for obtaining the academic degree

Master de Physique

in the course
Physique et Modélisation

26. August 2022

Faculté des Sciences
Université de Cergy-Pontoise

Supervised by:
Prof. Andriyanova Iryna
Prof. Honecker Andreas

Abstract

Since the development of quantum information theory, the hopes are to have a device that outperforms present-day classical computers. That might happen due to the quantum properties that classical bits do not have. This thesis will cover Grover's search algorithm that retrieves information from an N size unsorted data set with a number of iterations of order $O(\sqrt{N})$ while classically achieving it in a number of iterations of order $O(N)$ iterations which is more time-consuming. Current quantum processors are imperfect by being vulnerable to noise, so the other part will discuss the physical realization of quantum technologies, particularly superconducting transmon qubits, and how they are built and controlled. The work aims to build quantum circuits that perform Grover's algorithm and then run it on the Qiskit interface to see how it works and then on IBM quantum processors to see how noise affects the results. One circuit will outperform the other. Thus, the composition of the circuit plays a significant role. Also, some IBM processors will perform better than others. Finally, the good results will be subjected to error mitigation methods to improve the results.

Contents

Introduction

Quantum computing employs the ideas of quantum mechanics and information theory to address issues that are difficult or impossible to solve with today's computers. Quantum computing performs data operations by utilizing quantum phenomena such as quantum bits, superposition, and entanglement.

Grover's algorithm [1], commonly known as the quantum search algorithm, is a quantum method for searching an unstructured data set using fewer iteration steps than classical algorithms to locate the unique input to a black box function (Oracle) by amplifying its probability of being measured. Recently this algorithm was implemented by researchers on different quantum machines.

Researchers dedicated time to explaining the algorithm; they developed ways (circuits) that implement the theory [2, 3, 4]. However, few relate their work to the physical system (qubits) that performs this algorithm, such as the source of errors that appear in their results, and fewer try to implement an error correction model to the algorithm to get better results which should rely on the type and the distribution of errors, this makes going to a higher number of qubits infeasible.

This thesis will cover the theory of Grover's algorithm and its implementation on IBM devices, then a general view of how those IBM devices work and how noise arises there. We will be studying two Grover circuits by running them on the Qiskit interface and then on different devices on the IBM cloud, then comparing the circuits on one hand and the IBM devices on the other.

This work will be an introduction for those who want to work in the field of quantum computing in particular on the IBM cloud; this may enable future work in quantum error correction codes or applications of Grover's algorithm in real-life problems.

The number of qubits in this study is not enough to be superior to classical search algorithms, and the study is restricted to circuit model quantum computers, specifically the superconducting qubits.

The thesis will be divided as follows; the first section will be comparing in general classical and quantum computers, then in the second section, there will be a description of the function of Grover's algorithm and the building of Grover circuits and

ends by showing the result of implementing them in an ideal quantum computer on the Qiskit interface. The third section will discuss the physical components of IBM transmon qubits, how they function, and how to control them, then it will discuss some of the noise sources and the errors resulting from them. By the end of that section, the results of implementing both circuits -each on two different devices- will be shown and discussed. Finally, there will be a conclusion summarizing the work and giving prospects for this study.

1 Classical vs Quantum information

The computer has become a part of our daily life that facilitates it; it is a programmable device that stores, retrieves, and processes data. Computers have proven capable of solving many problems, including global communication, international commerce, the internet, and breakthrough in sciences such as medicine and aerospace engineering and reaching artificial neural networks. A computer consists of numerous building blocks (bits); an electric transistor can represent a bit. A bit holds binary information $(0, 1)$ in the form of two electrical current states on and off.
It is possible to use transistors for more than to turn the electrical current on and off and to allow for different levels of current. Why do not we see electronic computers were ternary[5], three states, or more states. It is impossible to go to these levels because the more intermediate states there are, the harder it is to keep them all distinguishable due to the noise at low energy levels or from the environment.
Boolean algebra's mathematical branch deals with these states and sets the necessary operations (gates) for manipulating them. In 1948 Claude Shannon constructed the information theory [6] that underlies digital communications, compression, and information storage. He also was one of the pioneers to digitalize Boolean algebra into an electrical circuit.
Quantum mechanics revolutionized physics and changed our view about the physical world being fundamentally probabilistic rather than deterministic. This theory states that very small or cold systems have wave properties, so they can be found simultaneously in two independent (orthogonal) states until they are measured, caus-

ing them to collapse into an individual state.

Quantum mechanics and information theory combined resulted in a new field called quantum information theory. If it is realized physically in a quantum computer, this device will be superior to a classical computer in some tasks. While a classical bit can hold binary information (0, 1), a quantum bit (qubit) can be in a superposition of $|0\rangle$ and $|1\rangle$ and generally represented as

$$|\text{qubit}\rangle = \cos(\theta/2)\,|0\rangle + e^{i\Phi}\sin(\theta/2)\,|1\rangle \tag{1}$$

More explanation can be found in Appendix A.1. [7] introduces quantum information and quantum computing.

The power of a quantum computer develops exponentially with the number of interacting qubits, while the power of a traditional computer grows linearly with the number of transistors. This superiority is one of the reasons why quantum computers may someday outperform traditional computers in certain types of calculations.

2 Grover's algorithm

Search Algorithms are intended to look for and recover an element from any structured data set in which it is stored. Classically, search algorithms were developed, such as Linear Search, where each element is checked (sorted or unsorted data sets) until the desired one is found; this algorithm takes an $N/2$ average number of iterations to get the result, where N is the number of data elements. Binary Search is a search algorithm that works on sorted data sets, and it goes by getting the average of the first and last element and then taking the interval that contains the wanted data; this algorithm takes the order $\log(N)$ number of steps.

A *Quantum search algorithm* is a protocol that enables one to get specific data in an unorganized data set. In the quantum realm, taking advantage of quantum parallelism, the algorithm can be performed with less number of iterations -less complexity than classical algorithms- using the superposition and entanglement properties of quantum states.

Let $N = 2^n$ be the number of possible permutations of n qubit states.

This protocol can be performed in $O(\sqrt{N})$ iterations to get the most probability of measuring the intended result. It seems that Binary Search with complexity $O(\log N)$ is faster than Grover's algorithm with complexity $O(\sqrt{N})$, but we forgot that Binary search works only for sorted data sets, and it needs $O(N * \log(N))$ to sort a data set.

The data set is is defined as a Hilbert space $\mathcal{H}^{\otimes N}$ with an orthonormal basis set

B = $\{|x\rangle = \bigotimes_i |q_i\rangle \mid |q_i\rangle = 0 \text{ or } 1\}$.

$f(x) : \mathcal{H}^{\otimes N} \to \mathcal{H}^{\otimes N}$ is a function such that

$$f(x) = \begin{cases} 0 & x \neq x_{sol} \\ 1 & x = x_{sol} \end{cases} \tag{2}$$

where x_{sol} is the element wanted in the data set.

Preparation step

First let us prepare the total state of qubits in $|s\rangle = |+ + ... +\rangle = |+\rangle^{\otimes n}$.

Let us initialize n qubits to $|0\rangle^{\otimes n}$ state and then apply a Hadamard gate $H^{\otimes n}$ to get $|+\rangle^{\otimes n}$ where the total state is a superposition of 2^n possible states (compare Appendix A.3).

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{2^n} |x\rangle \tag{3}$$

where $|x\rangle \in$ B.

This vector can be decomposed into two basis vectors $|\omega\rangle \equiv |x_{sol}\rangle$, the solution searched for, and $|s'\rangle \equiv \left|x_{sol}^{\perp}\right\rangle$, the combination of the rest of the states (vectors), which is orthogonal to the solution.

$$|s\rangle = \cos(\theta/2)\,|s'\rangle + \sin(\theta/2)\,|\omega\rangle \tag{4}$$

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{N} |x\rangle = \frac{1}{\sqrt{N}} \sum_{x \neq \omega} |x\rangle + \frac{1}{\sqrt{N}} |\omega\rangle \tag{5}$$

$$|s\rangle = \sqrt{\frac{N-1}{N}} \frac{1}{\sqrt{N-1}} \sum_{x \neq \omega} |x\rangle + \frac{1}{\sqrt{N}} |\omega\rangle = \sqrt{\frac{N-1}{N}} |s'\rangle + \frac{1}{\sqrt{N}} |\omega\rangle \tag{6}$$

The basis chosen is orthonormal. Targeted and the combination of the rest of the

states are the vertical and horizontal vectors, respectively, in Figure 1.

Grover's algorithm is a two-step protocol that is repeated (number of times de-

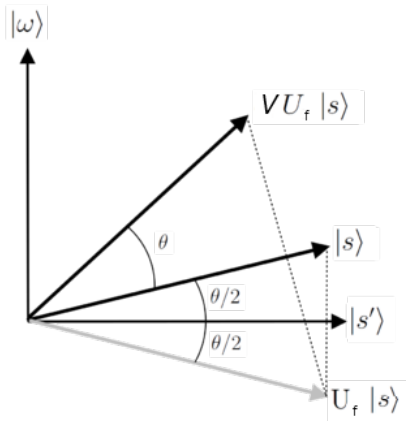


Figure 1: Geometrical representation of Grover's algorithm

pending on the number of qubits) until we reach the solution $|\omega\rangle$.

First step

Reflect the initial state $|s\rangle$ about $|s'\rangle$, and this is step denoted by U_f and called 'Phase Oracle' since it adds a minus sign (phase) to the $|\omega\rangle$ component.

$$U_f\,|s\rangle = \cos(\theta/2)\,|s'\rangle + e^{i\pi}\sin(\theta/2)\,|\omega\rangle = \cos(\theta/2)\,|s'\rangle - \sin(\theta/2)\,|\omega\rangle \tag{7}$$

Second step

Reflect the resulting vector $U_f\,|s\rangle$ about the initial vector $|s\rangle$. This transformation is denoted by V and called 'Diffuser'.

Outcome

The resulting state appears nearer to the solution than the initial state.

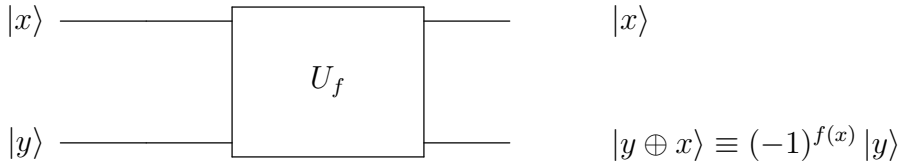
2.1 Circuit description

This section will describe two different circuits that will do the job described. Let us apply the previously explained steps U_f in section 2.1.1 and V in section 2.1.2 in each quantum circuit.

2.1.1 Phase Oracle

The first step of Grover's algorithm targets a specific state of the qubits, so we need some multi-control gates to achieve this (compare Appendix A.3.2).
We can formulate U_f simply as,

$$U_f = \mathbb{1} - 2\,|\omega\rangle\,\langle\omega| \tag{8}$$

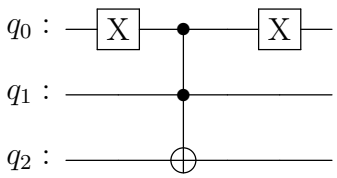


Circuit # 1

In this circuit, the qubits will be classified into three categories

- Control qubits: n qubits that hold the information of the data set (superposition of 2^n states as in equation 3).
- The target qubit is the last qubit in the circuit shown, and it is responsible for carrying the phase of the state.
- The ancilla qubit is the qubit that we do not care about its final form, but it helps to perform multi-control gates.

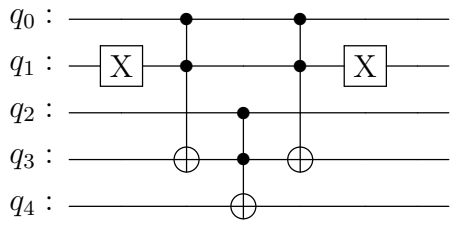
Add a target qubit that is prepared in $|-\rangle$, such that if a multi-control gate is applied to this qubit, a minus sign (phase) appears. $X\,|-\rangle = \frac{|1\rangle - |0\rangle}{\sqrt{2}} = -\,|-\rangle$ and because $|-\rangle$ is an eigenvector of X with an eigenvalue -1.



The Toffoli gate activates (flips the target) on $|11\rangle$, however in the circuit before, the addition of an X gate prior to the Toffoli gate made it activates on $|q_0 q_1\rangle = |01\rangle$. So using the Toffoli and X gate, we can build oracles that any specific state can

activate, and the property of $|-\rangle$ adds a phase to a specific state.

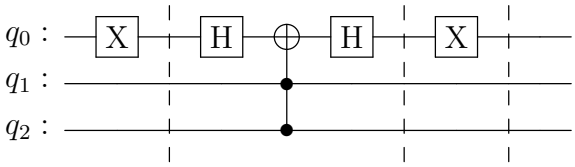
Here, we will present an example of targeting a particular state



In our particular case, we construct an oracle that adds a phase to q_4 at $|q_0q_1q_2\rangle = |101\rangle$, where q_3 is an ancilla qubit that helps to perform a three control gate that is not defined by qiskit. Check Appendix C.2.1 for its code.

Circuit # 2

In this circuit, the same idea of applying a NOT gate on the qubit is wanted to be $|0\rangle$. However, instead of having extra qubits (ancilla and target qubits), the targeted state is between the data set qubits (it does not matter which) followed and preceded by a Hadamard gate controlled by the other qubits.



This oracle, for example, targets $|110\rangle$ to invert it. The result measured will be viewed inverted in the form of $|q_2q_1q_0\rangle$ in general. Check Appendix C.3.1 for its code.

2.1.2 Diffuser

In this phase, all the states targeted to be flipped about the initial state $|s\rangle$, as shown in Figure 1.

Let $U_{f0} = 2|0\rangle^{\otimes n}\langle 0|^{\otimes n} - \mathbb{1}$

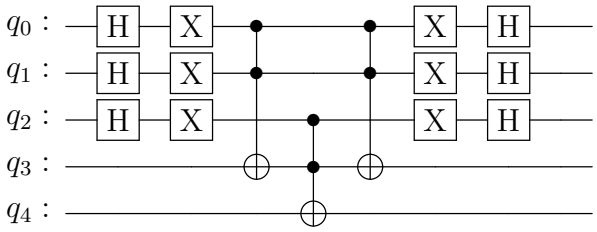
$$V = H^{\otimes n}U_{f0}H^{\otimes n} = 2|+\rangle^{\otimes n}\langle +|^{\otimes n} - \mathbb{1} = 2|s\rangle\langle s| - \mathbb{1} \tag{9}$$

This oracle reflects the given state about $|s\rangle$.

A straightforward application of the gates $H^{\otimes n}U_{f0}H^{\otimes n}$, where U_{f0} is simply U_f from section 2.1.1 targeting $|000\rangle$ state.

Circuit # 1

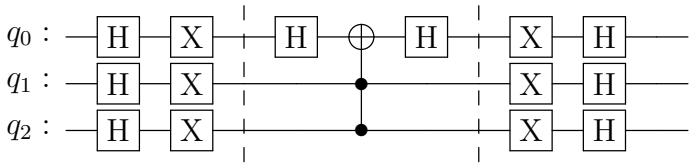
This oracle looks like *Circuit # 1* of section 2.1.1 targeting $|000\rangle$ with a Hadamard gate on each control qubit before and after U_{f0}



Check Appendix C.2.2 for its code.

Circuit # 2

This oracle looks like *Circuit # 2* of section 2.1.1 targeting $|000\rangle$ also with a Hadamard gate on each control qubit before and after U_{f0}



Check Appendix C.3.2 for its code.

2.1.3 Number of iterations

Circuits # 1 of sections 2.1.1 and 2.1.2 and Circuits # 2 of sections 2.1.1 and 2.1.2 should be performed repeatedly to get the desired result. However, is there a limit to the number of times we should do them? Alternatively, as we do it more, and more we approach the result.

The algorithm should should be repeated multiple times. If this was not respected, then the measurement results will not be as wanted.

Lemma 1. *The number of iterations to complete the Grover algorithm is of order* $O(\sqrt{N})$. *[4]*

Proof.

$$VU_f \left|s\right\rangle = \cos(3\theta/2) \left|s'\right\rangle + \sin(3\theta/2) \left|\omega\right\rangle \tag{10}$$

$$VU_f \left|s\right\rangle = \cos(\theta + \theta/2) \left|s'\right\rangle + \sin(\theta + \theta/2) \left|\omega\right\rangle \tag{11}$$

$$(VU_f)^r \left|s\right\rangle = \cos(r\theta + \theta/2) \left|s'\right\rangle + \sin(r\theta + \theta/2) \left|\omega\right\rangle \tag{12}$$

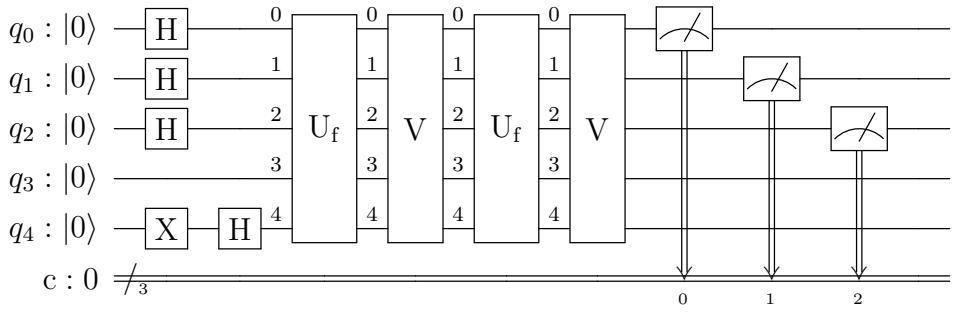
if $(VU_f)^r \left|s\right\rangle = \left|\omega\right\rangle$, then $r\theta + \theta/2 = \pi/2$ so $r = \pi/2\theta - 1/2$. Knowing that $\theta = \arcsin(1/\sqrt{N}) \simeq 1/\sqrt{N}$ then $r = \pi\sqrt{N}/2 - 1/2 \simeq O(\sqrt{N})$ □

For three qubits, the algorithm is repeated twice.

Now let us combine everything explained before into a circuit for each case. We get

Circuit # 1

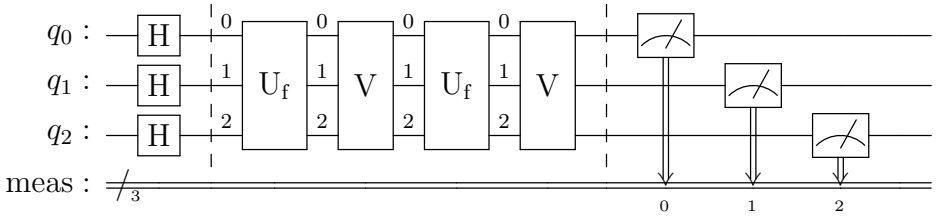
Put the Phase oracle and the Diffuser oracle in sequence and repeat that twice, as mentioned after the proof. After preparing 'q_0, q_1, q_2' in $|+++\rangle$ with Hadamard gates, and 'q_4' the target qubit in $|-\rangle$. Let them pass along 'q_3' the ancilla qubit through the circuit. Furthermore, measure the results at the end.



Check Appendix C.2.3 for its code.

Circuit # 2

The same is done for this circuit. But without using ancilla qubit or an extra target qubit since it is one of the qubits representing the input data (data set). Then the results are measured.



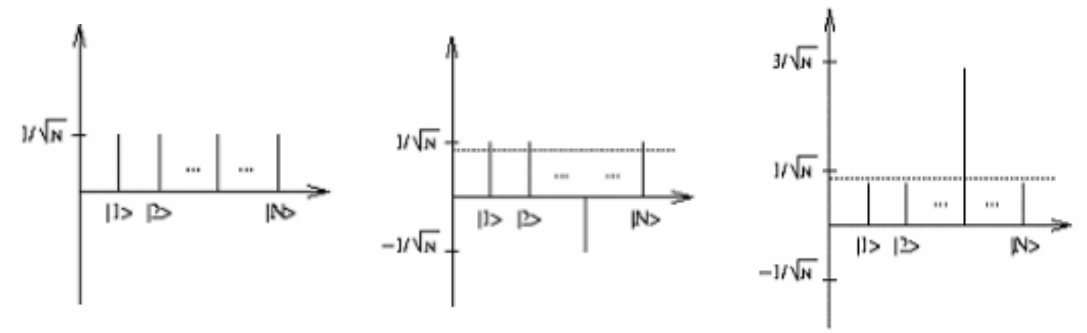
Check Appendix C.3.3 for its code.

This algorithm is not exact; there appears to be an angle gap between the wanted state and the actual final state with a maximum value $\theta << 1$. This angle represents the probability of not observing the wanted state after applying the algorithm or the error. The probability of that happening is

$$p_{error} = \sin^2(\theta) \simeq \theta^2 = \frac{1}{N} \tag{13}$$

2.2 Graphical representation

Another way to look at this algorithm is to visualize it as a reflection of the states' mean.



(a) All states have equal amplitude when we apply the Hadamard gates to initial zero states

(b) After U_f is applied, $|\omega\rangle$ is the inverted state, so the mean of the whole state decreases by a bit.

(c) After applying the Diffuser oracle, all the states are flipped around the mean

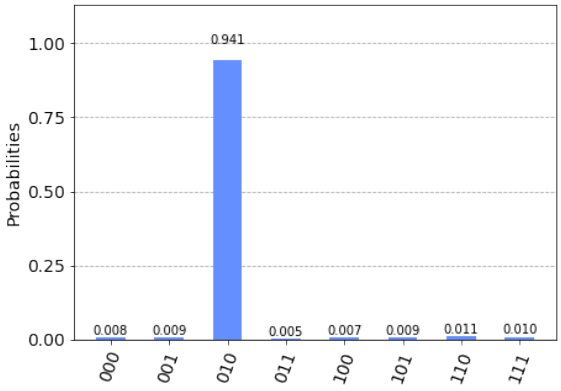
Figure 2: The X-axis represents the N states of n qubits; the Y-axis represents the probability of each state, taken from [4]

.

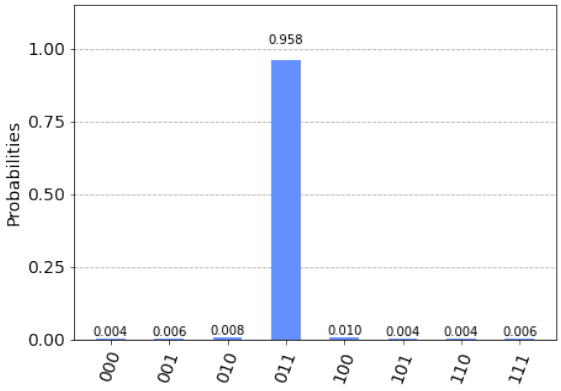
We can observe that we have $|\omega\rangle$ with a higher probability than other states.

2.3 Results

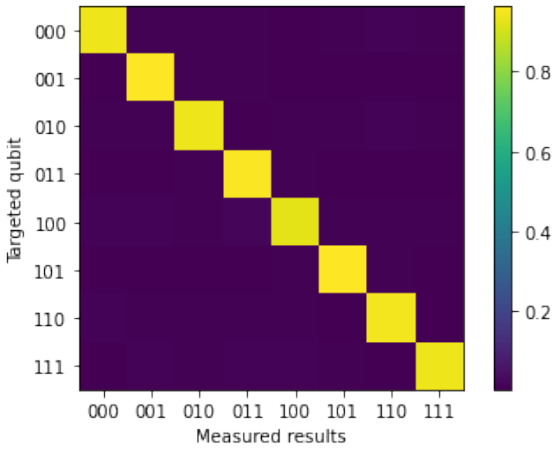
After discussing some protocols in this chapter, let us see if we can implement them on the Qiskit interface and then on IBM freely accessible devices. Let us present some of our results; we got the following results that will be explained, compared, and interpreted.



(a) $|010\rangle$ targeted. X-axis represents the states measured



(b) $|011\rangle$ targeted. X-axis represents the states measured



(c) Measuring the number of counts of each state while changing the target state

Figure 3: Performing both of the protocols on Qiskit interface

In Figures 3a and 3b, we target $|010\rangle$ and $|011\rangle$, respectively. As expected for a single solution algorithm, in ideal conditions of a quantum computer, the targeted state will appear with a probability near 1, and we can observe the algorithm's error in equation 13. We can also observe some small probabilities in other states than the targeted ones in Figures 3a and 3b, that is because of the angle gap expressed in equation 13. In Figure 3c, we target each state arranged on the vertical axis,

then measure the number of occurrences and display the probabilities of each state on the horizontal axis beside the targeted state. The color intensity represents the probability (represented on the color bar). As predicted, the color map should have a diagonal line indicating the wanted state's high destructibility.

3 State of the art on Quantum devices

Quantum computing is usually viewed as a single entity. However, there are several techniques for converting various quantum systems into quantum computers within the domain of quantum computing. There also exist different approaches to manipulating a system of qubits. The circuit model is a common way to get the qubits to reach a wanted state; it is performed by applying a series of unitary gates. The measurement-based model is a particular case of 'quantum circuits happens with intermediary measurements'. One first prepares an entangled resource state then performs single qubit measurements on it of form $M_{XY}(\theta) = \cos(\theta)X - \sin(\theta)Y$, where θ determines the basis of measurement. This measurement depends on the measurement of the previous qubit; the adiabatic model is motivated by quantum many-body theory, where a set of qubits are prepared and then subjected to a specific Hamiltonian so the system evolves adiabatically to a state that describes the solution to the problem; there are other models such as topological quantum computing and others. On the other side, the quantum computer can be realized in several quantum systems, such as superconducting transmon qubits that depend on a charge oscillating between two superconducting metals tunneling through a Josephson junction [8, 9, 10]. Trapped ions quantum systems are another kind of implementation of qubits; the system is a line of ions that are cooled in magneto-optical traps using lasers; the ion is controlled by a laser and transitions between two hyperfine components or Zeeman sublevels of the ground state [11, 12]. Photons are quantum systems that can represent a qubit [13]; photonic qubits depend on photon pulses controlled by optical devices such as beam splitters and phase shifters, and the readout can be through photon counters; this system can work at room temperature. Other systems can be mentioned, such as Magnetic Flux qubits that depend

on the direction of the magnetic field induced by a superconducting loop of wire; Quantum Dots can also represent a qubit where electrons are restricted to a small area in the material, and its state depends on the spin or the charge there.

3.1 IBM superconducting qubits

For their quantum processors, IBM developed superconducting qubits, also known as transmon qubits. Some processors can exceed a hundred qubits and are accessible via the IBM-Quantum cloud [14]. A *transmon qubit* is a microwave resonator that stores specific frequencies of electromagnetic waves, the energy of the qubit is confined between the ground state $|0\rangle$ and the first excited state $|1\rangle$. In this section, we will dive into the technology behind these qubits and explain the choice of anharmonic oscillators over harmonic oscillators solved exactly by quantum mechanics. Understanding that will help to identify the noise sources, that in turn helps to figure out the most efficient error correction model for the case we are studying.

3.1.1 Physical components

Let us first talk briefly about the superconductivity phenomenon [15]. As the temperature of a regular conductor decreases, the resistance lowers due to the fewer vibrations of the conductor's lattice but still resistants electrons, which is due to the impurity scattering between the electron and the lattice or electron-electron scattering at lower orders. This resistance carries away information about the electrons in the form of phonons, not allowing observing the resonator's quantum behavior. John Bardeen, Leon Cooper, and John Robert Schrieffer were awarded Nobel Prize in physics in 1972 for developing the microscopic theory of superconductivity (BCS theory)[16]. A superconducting material acts as a conductor at room temperature, but below some critical temperature, the electrons passing through the lattice deform it; this deformation attracts the electrons leading to a pairing of opposite spin electrons, called Cooper pairs. These pairs travel without any resistance in the lattice.

A transmon qubit is composed of two superconducting pads that are cooled down to millikelvins and connected by a bridge which is an inductor, as shown in Figure

4. An electric field is applied across the pads creating two (positive and negative) islands of charges, which induces an internal electric field and, as a voltage, the pads act as a capacitor that holds electric energy. If a charge passes (tunnels) from one island to another through the bridge, a magnetic field is formed around it, acting as an inductor of a circuit that holds magnetic energy. The system acts as an LC circuit where electromagnetic energy oscillates with specific frequencies. The derivations shown below in this chapter are presented in more detail in [10]

When the system is in a classical regime, we can describe it through a Hamiltonian

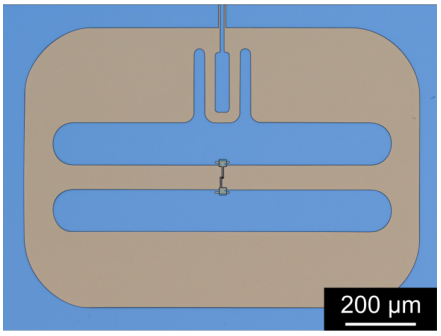


Figure 4: False-colored optical microscope image of a transmon qubit reproduced from Ref. [17]

$$H = \frac{Q^2}{2C} + \frac{\Phi^2}{2L} = \frac{1}{2}\hbar\omega_r(\alpha^*\alpha + \alpha\alpha^*) \tag{14}$$

Q and Φ are two variables: the charge of a pad and the magnetic flux, respectively, and the systems C and L parameters are the capacitance and the inductance.

α is introduced to simplify the description, such as

$$\alpha(t) = \sqrt{\frac{1}{2\hbar\omega_r Z}}(\Phi(t) + iZQ(t)) \tag{15}$$

$Z = L/C$ is the impedance, and $\omega_r = 1/(LC)$ is the resonance frequency.

Going to quantum, we convert the variables to operators. α and α^* become $\hat{a}$ and $\hat{a}^\dagger$, respectively, which will be known as the lowering and raising operators, respectively. $\hat{a}\,|n\rangle = \sqrt{n}\,|n-1\rangle$ and $\hat{a}^\dagger\,|n\rangle = \sqrt{n+1}\,|n+1\rangle \implies \langle n|\,\hat{a}^\dagger\hat{a}\,|n\rangle = n$ and

$$\hat{H} = \frac{1}{2}\hbar\omega_r(\hat{a}^\dagger\hat{a} + \hat{a}\hat{a}^\dagger) \tag{16}$$

Here we dropped an additive constant since we measured the energy differences. $\hat{Q} = iQ_{ZPF}(\hat{a}^\dagger - \hat{a})$ and $\hat{\Phi} = i\Phi_{ZPF}(\hat{a}^\dagger + \hat{a})$, where Q_{ZPF} and Φ_{ZPF} are the zero-point fluctuations of the charge and phase variables respectively, these two constants can be viewed as the variance of the wave-function at the ground state.

This case is valid if the inductor bridge is linear, which results in a harmonic oscillator, as shown in Figure 5a, with equal distancing between the energy levels. That creates a problem in distinguishing the first two states from the rest. This problem pushed introducing anharmonicity through a non-linear bridge or a Josephson junction. A Josephson junction consists of two superconducting pieces separated by an insulator, shown in Figure 6, through which the cooper pairs tunnel [18]. This process results in a Hamiltonian as the following

$$\hat{H} = \frac{\hat{Q}^2}{2C} - E_J \cos\left(\frac{\hat{\phi}}{\phi_0}\right) \tag{17}$$

E_J being the Josephson energy, $\hat{\phi}/\phi_0$ is the reduced magnetic flux with $\phi_0 = \hbar/2e$. Applying Taylor expansion to the equation to the fourth order and rewrite the Hamiltonian in terms of the raising and lowering operators. We get

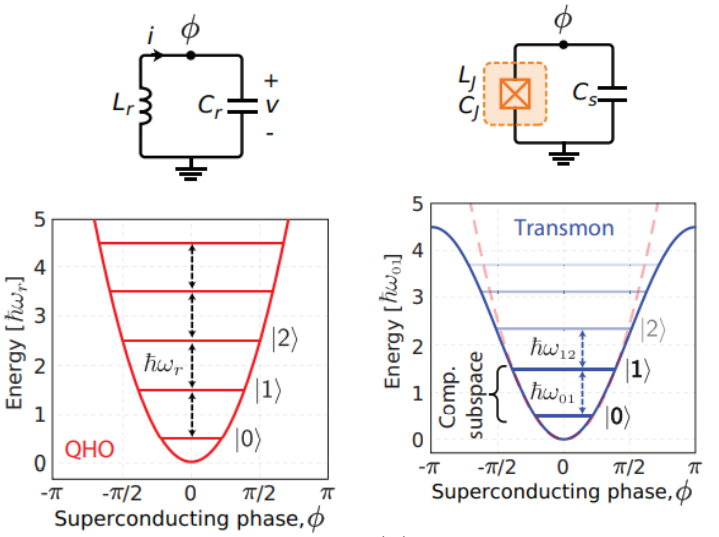
$$\hat{H} \simeq \frac{\hat{Q}^2}{2C} + \frac{\hat{\phi}^2}{2L_J} - \frac{E_J}{4!}\left(\frac{\hat{\phi}}{\phi_0}\right)^4 = \hbar\omega_r \hat{a}^\dagger \hat{a} - \frac{E_J \Phi_{ZPF}^4}{4!}(\hat{a} + \hat{a}^\dagger)^4 \tag{18}$$

Where Φ_{ZPF} is the zero point fluctuation of $\hat{\Phi}$.

Classically, α from equation 15 evolves with time in phase space as $\alpha(t) = \alpha(0)e^{-i\omega_0 t}$ in an oscillatory motion with frequency ω_0. In the quantum case, the exact evolution of $\hat{a}$ within Heisenberg's picture as $\hat{a}(t) = \hat{a}(0)e^{-i\omega_0 t}$ and $\hat{a}^\dagger(t) = \hat{a}\dagger(0)e^{i\omega_0 t}$. Expanding equation 20 and using $[\hat{a}^\dagger, \hat{a}] = \mathbb{1}$, we get

$$H(t) = \sum_{i,n=m} c_i \hat{a}^{\dagger^n}(t)\hat{a}^m(t) + \sum_{j,n\neq m} d_j \hat{a}^{\dagger^n}(t)\hat{a}^m(t) = \hat{H}_0 + \hat{H}_R(t) \tag{19}$$

H_0 is a stationary Hamiltonian where the exponentials cancel each other since $n = m$, while $H_R(t)$ is a rotating Hamiltonian. Without going into calculation details and using the Rotating Wave Approximation[19], the rotating term is neglected,



(a) LC circuit representing the quantum harmonic oscillator and its quantized energy levels [8]

(b) Josephson qubit circuit representing the quantum anharmonic oscillator and its quantized energy levels [8]

Figure 5: Harmonic and anharmonic oscillators

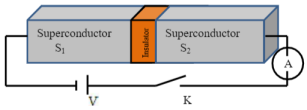


Figure 6: Schematic diagram of a Josephson junction.[20]

and we get

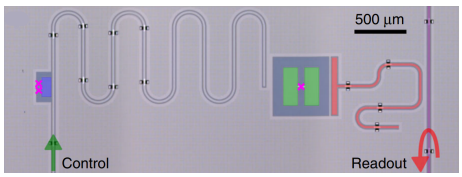
$$\hat{H} \simeq \hbar\omega_0 \hat{a}^\dagger \hat{a} - \frac{\hbar\alpha}{2} \hat{a}^\dagger \hat{a}(\hat{a}^\dagger \hat{a} - I) \tag{20}$$

The first term is the linear term, and the second is the nonlinear term or the perturbation caused by the Josephson effect.

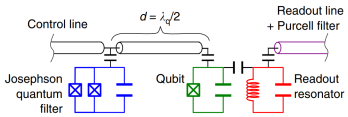
3.1.2 Qubit control

Controlling a qubit is possible by coupling it to an external control such as a microwave drive that detects photons in the form of Rabi oscillations Figure 7. The coupling between it and the qubit will provide a faster control on the qubit, but on the other hand, there is a higher probability of dissipating the qubit information in the form of radiative decay. While a weaker coupling reduces this dissipation, it makes it slower to control the qubit. The Josephson quantum filter can suppress this

decay by reflecting those photons to the qubit [9], but this will not be part of the discussion or calculations. The readout of the information of the qubit is performed through the circuit electrodynamics (cQED) dispersive monitoring technique; this involves coupling the qubit to a quantum LC oscillator. This cavity isolates the qubit from environmental noise and prevents it from leaking out information, and most importantly, it permits to performance of a non-destructive measurement on the qubit. The derivations shown below in this chapter were performed in more detail by [10]



(a) Schematic view of transmon qubit with its control and readout.[9]



(b) Circuit of the transmon qubit with its control and readout and the filter. [9]

Figure 7: Schematic and circuit representation of the qubit and its control and readout

Let us first take the simplest possible circuit by putting the cavity aside. We end up having a drive coupled directly to the qubit; let us see the Hamiltonian of this system.

$$\hat{H}_{drive} = -i\Omega(t)/2(\hat{c}^\dagger - \hat{c}) \tag{21}$$

$\hat{H}_{drive}$ is the Hamiltonian of the drive with $\Omega(t) = \Omega_0 \sin(\omega_d t + \theta_d)$, with ω_d being the frequency of the drive with phase $\theta_d \equiv \theta$. The total Hamiltonian of this system is

$$\hat{H}_t = \hbar\Delta\hat{Z}/2 + \hbar\Omega/2(e^{-i\theta}\hat{\sigma} + e^{i\theta}\hat{\sigma}^\dagger) \tag{22}$$

The first term is related to the tuning of the qubit. While the second term varies the phase σ, we can apply simple gates such as $\hat{X} = \frac{1}{2}(\hat{\sigma}^\dagger - \hat{\sigma})$, and $\hat{Y} = \frac{i}{2}(\hat{\sigma}^\dagger + \hat{\sigma})$ and then build more complicated gates. $\hat{\sigma} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$

In quantum mechanics, if a measurement is performed on a qubit, it collapses in one state. Check AppendixA.1. Now, let us explain how to measure the qubit's

state without destroying it. Consider the cavity while calculating the Hamiltonian to see how the non-destructive measurement can happen. The total Hamiltonian is

$$\hat{H}_{total} = \hat{H}_{lin} + \hat{H}_{nonlin} \tag{23}$$

where $\hat{H}_{lin} = \hbar\omega_c \hat{b}^\dagger \hat{b} + \hbar\omega_q \hat{a}^\dagger \hat{a}$ and $\hat{H}_{nonlin} = -\frac{E_a}{4!}[\Phi_{1a}^{ZPF}(\hat{a}^\dagger + \hat{a}) + \Phi_{1b}^{ZPF}(\hat{b}^\dagger + \hat{b})]^4$ $\hat{a}^\dagger$ and $\hat{a}$ are the raising and lowering operators of the qubit, respectively. $\hat{b}^\dagger$ and $\hat{b}$ are the raising and lowering operators of the cavity, respectively. E_a is the Josephson energy, and ω_c and ω_q are the cavity frequencies and the qubit, respectively.
Applying the Rotating Wave approximation, the effective Hamiltonian of the cavity is

$$\hat{H}_{cav}^{eff} = \hbar(\omega_c - \chi \hat{a}^\dagger \hat{a})\hat{b}^\dagger \hat{b} \tag{24}$$

As seen, it depends on the state of the qubit $\hat{a}^\dagger \hat{a}$. Preparing the oscillator of the cavity in the first excited state gives an effective energy

$$E_{eff} = \hbar(\omega_c - \chi n_{qubit}) \tag{25}$$

Knowing How the transmon qubit functions, let us investigate some of the noise

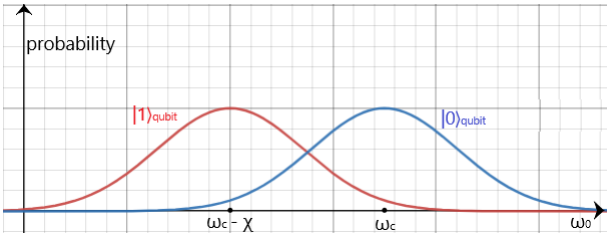


Figure 8: Predict the state of the qubit through the effective energy of the cavity

sources.
As seen in Figure 8, to predict the qubit's state, we need to measure the effective energy of the cavity, and it will be probabilistic to guess the qubit's state, which causes errors. According to [8], we can find different sources in the system;

- Charge noise can be induced by the presence of charged particles in the defects or charge traps in the dielectric that sit in the junction tunnel barrier; it mainly causes energy relaxation of the qubit, check section3.2.

- Magnetic flux noise forms due to random polarization flipping on the surfaces of the superconductors; this contributes to the phase shifting error; check section3.2.

- The residual photons in the resonator, which come from higher temperature stages of the dilution refrigerator, might cause photon number changes.

- Unpaired electrons (Quasi-particles) tunnel through the Josephson junction can cause relaxation and pure dephasing, check section3.2; this will reduce as the temperature decreases.

3.2 Error Models

From previous chapters, it is clear that the implementation of qubits is not ideal, leading to errors in the results. In this section, we will be listing theoretical error models. Errors can be classified into stochastic and coherent noises. In application, it is impossible to isolate a system, especially a quantum bit (qubit), from its environment; there will always be a random interaction or coupling between those two systems, which leads to the decoherence of the qubit or its loss of irretrievable information, this is called a *stochastic* noise. On the other side, *coherent or systematic noise* forms due to the imperfection of the device used, such as cross-talk (unwanted interaction between qubits) or imperfect unitary operators applied, such as miscalibrations drift (over or under rotate the state of the qubit slightly from the needed state). These transformations do not affect the coherence of the qubit; in fact, these errors might be controlled to interfere with each other destructively.
We can realize quantum errors in noise channels such as bit-flip, phase-flip, depolarization, amplitude, gate, and readout channels. The binary symmetric channel can describe bit flip error with probability p of flipping a state from $|0\rangle$ to $|1\rangle$ and vice versa $1-p$ probability to preserve the state, check Figure9, p will increase with time $p = \frac{1}{2}(1 - e^{-\frac{t}{T}})$ where T is the bit flip coherence time of the qubit. For example, let us see the pure $|0\rangle\langle 0|$ state transformation in a binary asymmetric channel

$$|0\rangle\langle 0| \rightarrow (1-p)\,|0\rangle\langle 0| + p\,|1\rangle\langle 1| \tag{26}$$

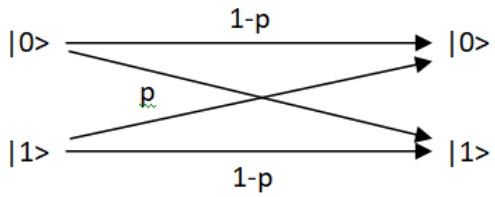


Figure 9: Scheme of a binary symmetric channel.

We can also build a binary asymmetric channel where the probability of flipping from $|0\rangle$ to $|1\rangle$ is not equal to that from $|1\rangle$ to $|0\rangle$. In the phase-flip channel, we lose a well-defined relative phase between $|0\rangle$ and $|1\rangle$. This phenomenon is an exclusive quantum error not observed in classical bits. Check AppendixA.3 for a basic explanation of the operators used here. We can describe bit-flip and phase-flip errors through Pauli operators, σ_X as a bit-flip error, σ_Z as a phase-flip error, and σ_Y as a combination of those two errors.

A general representation of the transformations caused by these errors was given by [3] as follows: Let p_σ be the probability of occurrence of each error, where$\sigma = \sigma_X, \sigma_Y, \sigma_Z$. Consider 2 operators $E_0 = \sqrt{1-p_\sigma}\mathbb{1}$ (where no error occurs, acts as an identity) and $E_1 = \sqrt{p_\sigma}\sigma$ (where error occurs). The general transformation on the density matrix ρ is as follows:

$$\rho \to \rho' = \epsilon(\rho) = E_0 \rho E_0^\dagger + E_1 \rho E_1^\dagger \tag{27}$$

In the previous noise channels, the state of the qubit ρ becomes a mixed state. These transformations can be described as one transformation that we can call a *depolarization channel*. It can be represented in Kraus form [3].

$$\epsilon(\rho) = \sum_{i=0}^{3} E_i \rho E_i^\dagger \tag{28}$$

with $E_0 = \sqrt{1-\frac{3p}{4}}\mathbb{1}, E_1 = \sqrt{\frac{p}{4}}X, E_2 = \sqrt{\frac{p}{4}}Y_1, E_3 = \sqrt{\frac{p}{4}}Z$ and p/4 is the probability of getting at least an error.

In the depolarization channel, the qubit loses information; on the other hand, there is an amplitude damping channel where the qubit loses energy in the form of deex-

citation from $|1\rangle$ to $|0\rangle$. Its transformation can be described as in 27, but with

$$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix}; E_1 = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix} \tag{29}$$

All the previous equations satisfy

$$\sum_i E_i E_i^\dagger = \mathbb{1} \tag{30}$$

That preserves the normalization of the state of the qubit.

3.3 IBM Calibration

IBM processors differ in terms of the number of qubits and performance. IBM allows its users to identify that through some parameters.

- *'T1' [μs]* and *'T2' [μs]* are the qubit relaxation and the dephasing times, respectively.
- *'Frequency' [GHz]* refers to the one to excite the qubit w_{12}(eigenvalue of $\omega_0 \hat{a}^\dagger \hat{a}$ of the first term of equation20).
- *'Anharmonicity' [GHz]* refers to the frequency difference between w_{12} and w_{23} (eigenvalue of $-\frac{\alpha}{2}\hat{a}^\dagger\hat{a}(\hat{a}^\dagger\hat{a} - I)$ of the secon term of equation20).
- *'Readout assignment error'* is the probability of a measurement returning the wrong value.
- *'Prob meas0 prep1'* is the probability of measuring $|0\rangle$ after preparing the $|1\rangle$ state and vice versa for *'Prob meas1 prep0'*.
- *'Readout length' [ns]* is the time it takes to perform a measurement.
- *'ID error'* is the probability of the error induced by having an inactive qubit.
- *'$\sqrt{X}$ error'* and *'Single-qubit Pauli X error'* probability of getting an error induced by applying the gate.

- *'CNOT error'* is the probability two-qubit gate error.
- *'Gate time' [ns]* is the time it takes to perform a two-qubit gate.

Figure10 is an example that shows the calibrations of *ibm_ manila*.

Qubit	T1 (us)	T2 (us)	Frequency (GHz)	Anharmonicity (GHz)	Readout assignment error	Prob meas0 prep1	Prob meas1 prep0	Readout le
Q0	152.73	74.99	4.962	-0.34335	3.090e-2	0.0488	0.013	5351.111
Q1	220.53	36.81	4.838	-0.34621	6.260e-2	0.0738	0.0514	5351.111
Q2	179.66	27.34	5.037	-0.34366	2.140e-2	0.0318	0.011	5351.111
Q3	197.55	62.47	4.951	-0.34355	2.730e-2	0.0368	0.0178	5351.111
Q4	147.07	42.02	5.065	-0.34211	2.480e-2	0.0392	0.0104	5351.111

Qubit	meas0 prep1	Prob meas1 prep0	Readout length (ns)	ID error	√x (sx) error	Single-qubit Pauli-X error	CNOT error	Gate time (ns)
Q0	8	0.013	5351.111	2.182e-4	2.182e-4	2.182e-4	0_1: 1.077e-2	0_1: 277.333
Q1	8	0.0514	5351.111	6.278e-4	6.278e-4	6.278e-4	1_2: 1.448e-2 1_0: 1.077e-2	1_2: 469.333 1_0: 312.889
Q2	8	0.011	5351.111	2.245e-4	2.245e-4	2.245e-4	2_3: 7.507e-3 2_1: 1.448e-2	2_3: 355.556 2_1: 504.889
Q3	8	0.0178	5351.111	2.831e-4	2.831e-4	2.831e-4	3_4: 7.800e-3 3_2: 7.507e-3	3_4: 334.222 3_2: 391.111
Q4	2	0.0104	5351.111	4.367e-4	4.367e-4	4.367e-4	4_3: 7.800e-3	4_3: 298.667

Figure 10: Table of calibration of *ibm_ manila*

This table shows the parameters defined in section 3.3 of each qubit in the system. This calibration changes temporally.
Quantum volume is a parameter set by IBM to determine how good is the performance of the quantum processor; it takes into account the number of qubits, gate and measurement errors, crosstalk, and the connectivity of the qubits.

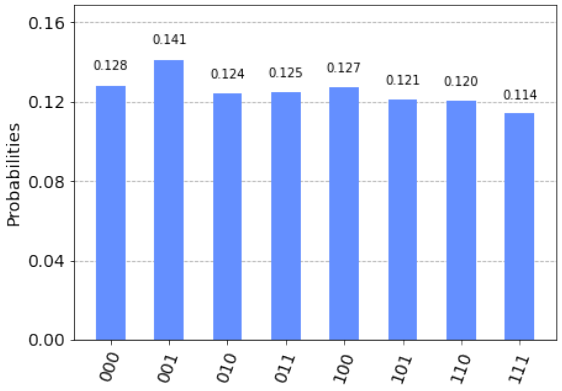
3.4 Error mitigation

With an ultimate goal to reach fault-tolerant quantum computers by performing fault-tolerant error correction, the systems have an insufficient number of qubits to do that. Quantum error mitigation QEM is a path to get improved results of a quantum computation in the NISQ (Noisy Intermediate Scale quantum) era.
This technique involves preparing the qubits and repeatedly executing them into a particular circuit. Then by comparing the counts (probabilities) of the outcome and the expected outcomes, the errors that occurred can be considered in some way

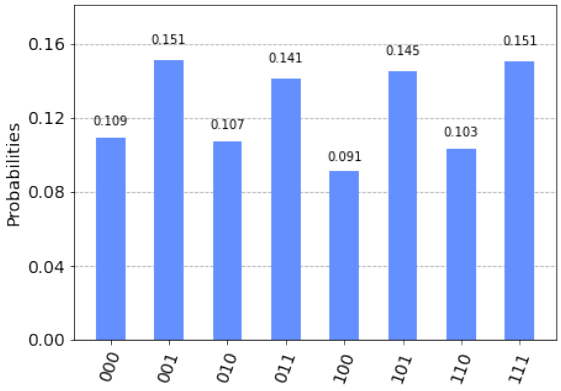
correlated such that they are related in a matrix transformation of the qubit. This matrix can be reversed every time a circuit is applied. The QEM method will work efficiently in low Noise to Signal Ratio cases. Otherwise, it is useless.

3.5 Results

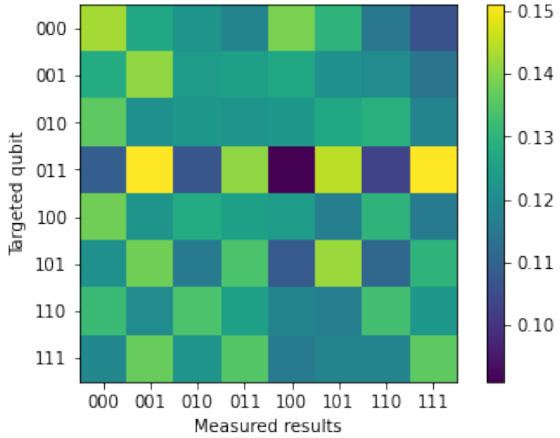
After describing the processors that will be worked on, the noise generated by the processor, and the kind of errors produced, let us run both circuits (*Circuit # 1* and *Circuit # 2*), each on two different processors, to see those effects.
After running *Circuit # 1* on *ibm_manila* and *ibm_santiago* processors, we get the following results



(a) $|001\rangle$ targeted. X-axis represents the states measured



(b) $|011\rangle$ targeted. X-axis represents the states measured

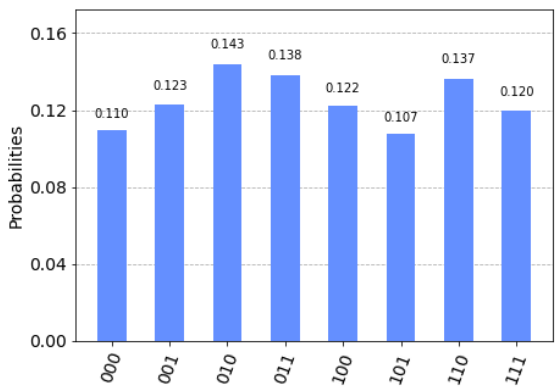


(c) Measuring the number of counts of each state while changing the target state

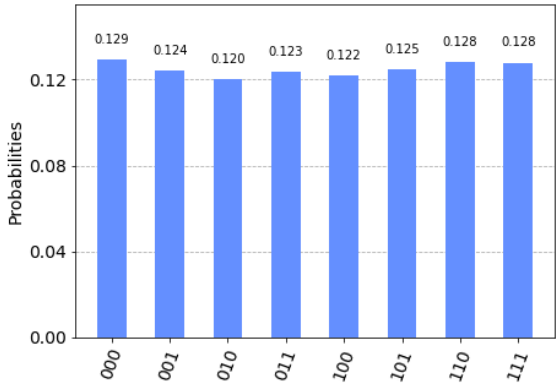
Figure 11: Performing *Circuit # 1* on *ibm_manila*

In Figure 11, *Circuit # 1* is run on *ibm_manila*. Figure11a shows the counts

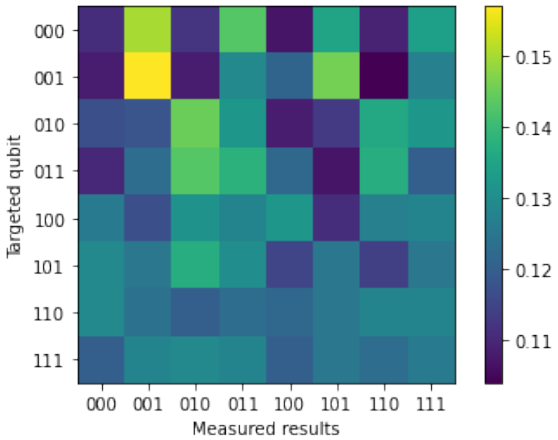
of each state resulting from running the circuit on several shots and presented as a probability on the Y-axis; on the X-axis, we observe the states measured on each shot ($|000\rangle, |001\rangle$, Etc); this result is after targeting $|001\rangle$. The same can be said about Figure 11b, but this results after targeting $|011\rangle$. In Figure11c, each state is targeted individually; targeted states are listed on the Y-axis; beside each state, we list the counts (probabilities) in the outcome, which are listed on the X-axis. The probabilities are visualized as the color of each pixel and defined on the color bar on the right. The probabilities of all states seem to be close to each other, with some fluctuations in Figure11a and Figure11b. Figure11c looks somewhat varying around the middle of the color bar; on $|011\rangle$, it seems random.



(a) $|011\rangle$ targeted.



(b) $|110\rangle$ targeted.



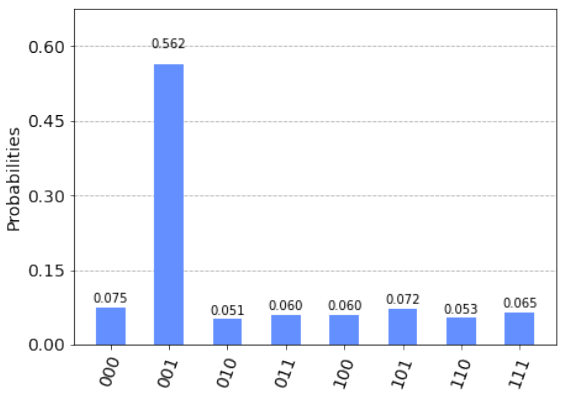
(c) Same as in Figure11c

Figure 12: Performing *Circuit # 1* on *ibm_santiago*. Same as in Figure11

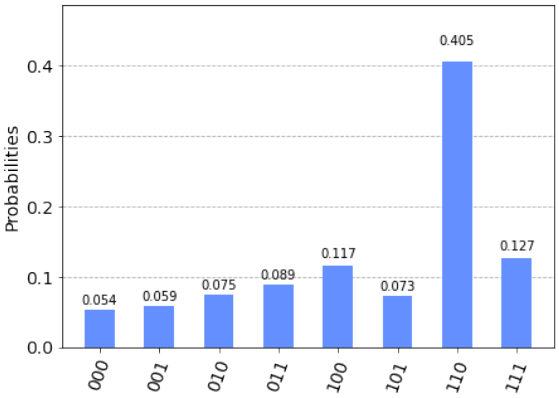
In Figure 12, *Circuit # 1* is now run on *ibm_santiago*. The same comments as on Figure11a can be made about Figures 12a and 12b. Figure12a shows the

result after targeting $|011\rangle$. Figure 11b shows the result after targeting $|011\rangle$. A detailed description of Figure11c can be said in Figure 12c. In general most of the sub-Figures in Figures11 and Figure12 show almost equal probability distributions in all the states. Figure12c looks like a random distribution of colors in the upper pixels and is somewhat smeared out in the lower pixels.

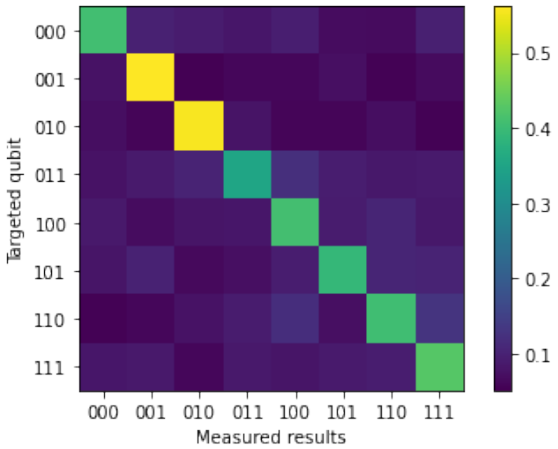
Let us see the results of *Circuit # 2* on *ibm_oslo* and *ibm_nairobi* processors too. The results are as follows



(a) $|001\rangle$ targeted.



(b) $|110\rangle$ targeted.

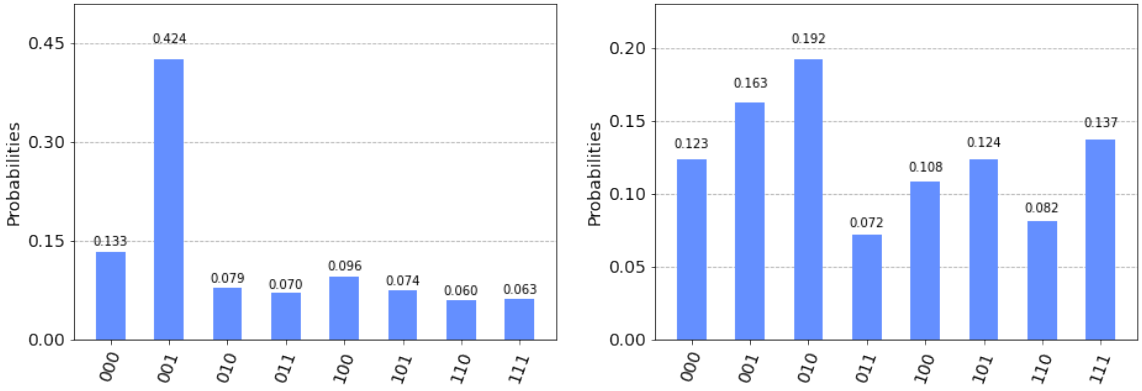


(c) Same as in Figures11c, and 12c

Figure 13: Performing *Circuit # 2* on *ibm_oslo*. Same as in Figures11, and 12

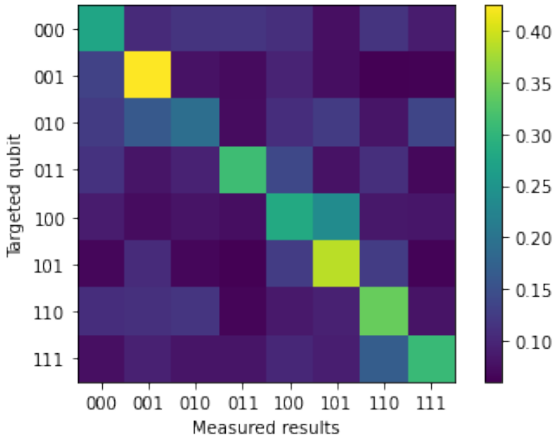
In Figure13, we execute *Circuit # 2* on *ibm_oslo*. Figures 13a and 13b can be the same described as Figure11a, but they are the result of targeting *Circuit # 2* on $|001\rangle$ and $|110\rangle$, respectively. Figure13c same described as Figure11c. In Figures 13a and 13b, we can see that probability is mostly in one state, Figure 13a in $|001\rangle$ and Figure 13b in $|110\rangle$, with smaller probabilities in the rest of the states. Figure

13c shows a pattern in the color map, and it looks like such a diagonal line that its elements(pixels) have high probabilities.



(a) $|001\rangle$ targeted.

(b) $|010\rangle$ targeted.



(c) Same as in Figures11c, 12c, and 13c

Figure 14: Performing *Circuit # 2* on *ibm_nairobi*. Same as in Figures11, 12, and 13

The same explanations of Figure13 can be said for all sub Figures in Figure14.In Figure 14a, we can see that probability is concentrated mainly in $|001\rangle$ with smaller probabilities in the rest of the states. While in Figure 14b $|010\rangle$ is seen to have the highest probability, the other states have higher probabilities than Figure 14a. Figure14c also shows a diagonal line, but a bit more faded than in Figure13c.
Finally, let us try to implement a quantum error mitigation on *Circuit # 2*, which targeted one of the states. We got the following results Figure15 shows the probability distribution on the whole states in noisy (original) form (in blue) and mitigated form (in red). The probability can be seen accumulated in both cases in $|110\rangle$.

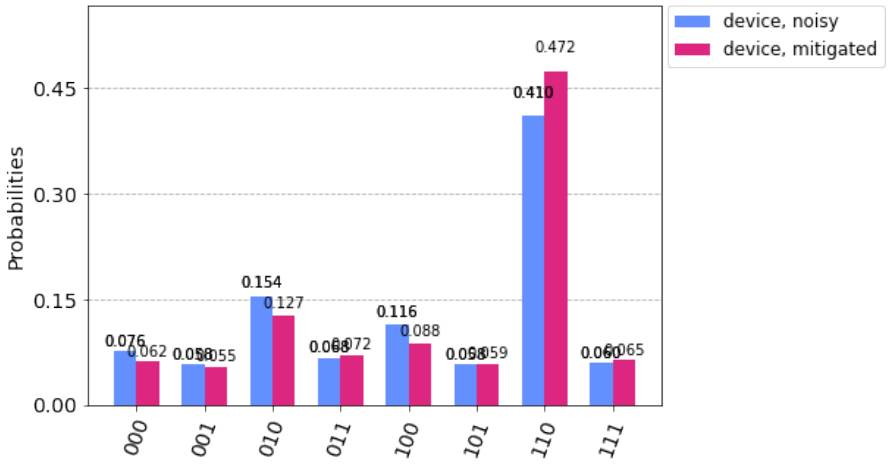


Figure 15: $|110\rangle$ is targeted by *Circuit # 2* and executed on *ibmq_quito*

4 Discussion

The results in Figures 11a,11b,12a, and12b do not contain any valuable information for not getting information about the targeted states. That is due to high noise that we can not distinguish the targeted states, and that is clear in Figures 11c and 12c, which look random, the change of errors between targeted states can be explained by needing different number of gates in different cases (X gates, for example, see section 2.1.1) and these noises are either random or pseudo-random processes that will change even running the same circuit at different times. In the color map, the probabilities are distributed randomly such that there is no trace of a diagonal line. We might need several error correction models to start seeing results to solve that. Moreover, since the results are so poor, we cannot even compare the two different devices, *ibm_manila* and *ibm_santiago*, and this shows that the cause of the poor results is not a particular processor but the circuit construction itself.

Figures 13a,13b, and 14a show that the probabilities mostly accumulate in the targeted states. That indicates that Grover's algorithm has succeeded when implementing *Circuit # 2* in this case. We can see that also in Figures 13c and 14c, where a diagonal line appears as in Figure3c in the ideal case. We can frequently distinguish target states in *Circuit # 2*. However, in Figure14b, we observe relatively high errors compared to before; this takes us to compare *ibm_oslo* and *ibm_nairobi*, simply comparing between Figures 13c and 14c, we can see that Figure 13c has a

more apparent diagonal line than in Figure14c. We can say then *ibm_ oslo* performs better than *ibm_ nairobi* in *Circuit # 2*.

We can see that Quantum Error Mitigation improves the results, as seen in Figure15. We observe that the probability increased after QEM, and most other states decreased their probability (decrease of errors).

To quantify how good or bad the device distinguished a targeted state in a circuit, [3] proposed a parameter S and called it the sensitivity.

$$S = 10log_{10}\left(\frac{P_t}{P_{hn}}\right) \tag{31}$$

Where P_t is the probability of the target state and P_{hn} is the probability of the highest noisy state. Where $S \geqslant 3$ is the critical point where the device starts to be sensitive to the target state.

The average sensitivity of each device is as follows $S_{ibm\_oslo} = 6.7$, $S_{ibm\_nairobi} = 3.27$, further showing the advantage of *ibm_ oslo* and *ibm_ nairobi* on the edge of the critical point.

The sensitivity in Figure15 before QEM is $S_{before} = 4.25$, and after QEM is $S_{after} = 5.7$. We can see here how QEM improved the sensitivity of the measurement.

5 Conclusion

Let us conclude this study by summarizing of the fundamental results concerning to our initial aims on the topic. We will also review the study's limitations and propose opportunities for future research.

As we have seen in the results, *Circuit # 2* outperformed *Circuit # 1* in distinguishing the targeted states, and that is quantified by calculating the average sensitivity of where $S_{ibm\_manila} \simeq -0.1$ is the average sensitivity of *Circuit # 1* on *ibm_ manila*, which is incomparable to the one of *Circuit # 2*. *Circuit # 1* has more qubits involved in the computation than *Circuit # 2*; in turn, the number of gates used in the circuit is defined as the depth of the circuit. We can conclude that optimizing the circuit (decreasing its depth) improves the results, especially in such a noisy

channel. On the other hand, comparing the performance of *Circuit # 2* on two different IBM processors, *ibm_ oslo* and *ibm_ nairobi*, showed that *ibm_ oslo* shows less error (better performance) than *ibm_ nairobi*, and that was confirmed by comparing their average sensitivities. Furthermore, at the end, the result in Figure15 shows that quantum error mitigation can make a difference in the results by improving them. This study was limited to three qubits (2^3 states), but to make sense of the study, we have to go to a higher number of qubits, which may result in more errors due to the need to use more gates. Then we will need some error correction models to get more reliable results. The study may also be extended to target more qubits, which will struggle with the same problems of more qubits circuit, which is the high depth of the circuit. Finally, we can suggest an application of Grover's algorithm, Quantum Random Access Memory (QRAM) [21], by introducing a new kind of qubit representing the address the code will get after finding the data.

References

[1] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.

[2] Robert Loredo. *Learn Quantum Computing with Python and IBM Quantum Experience*. Packt, BIRMINGHAM—MUMBAI, 2020.

[3] Yulun Wang and Predrag S. Krstic. "prospect of using grover's search in the noisy-intermediate-scale quantum-computer era". *Phys. Rev. A*, 102:042609, Oct 2020.

[4] Eva Borbely. Grover search algorithm, 2007.

[5] David Scott Brown. Why not ternary computers?, 2021.

[6] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[7] Chris Bernhardt. *Quantum Computing for everyone*. THE MIT PRESS, Cambridge, Massachusetts, United States, 2019.

[8] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2):021318, jun 2019.

[9] S. Kono, K. Koshino, D. Lachance-Quirion, A. F. van Loo, Y. Tabuchi, A. Noguchi, and Y. Nakamura. Breaking the trade-off between fast control and long lifetime of a superconducting qubit. *Nature Communications*, 11(3683), jul 2020.

[10] Zlatko K. Minev, Zaki Leghtas, Shantanu O. Mundhada, Lysander Christakis, Ioan M. Pop, and Michel H. Devoret. "energy-participation quantization of josephson circuits", 2020.

[11] A. Steane. The ion trap quantum information processor. *Applied Physics B: Lasers and Optics*, 64(6):623–643, jun 1997.

[12] A. L. Wolf, S. A. van den Berg, W. Ubachs, and K. S. E. Eikema. Direct frequency comb spectroscopy of trapped ions. *Physical Review Letters*, 102(22), jun 2009.

[13] Pieter Kok, W. J. Munro, Kae Nemoto, T. C. Ralph, Jonathan P. Dowling, and G. J. Milburn. Linear optical quantum computing with photonic qubits. *Rev. Mod. Phys.*, 79:135–174, Jan 2007.

[14] IBM. Ibm quanntum, 2022.

[15] Michael Tinkham. *Introduction to superconductivity*. Courier Corporation, 2004.

[16] J. Bardeen, L. N. Cooper, and J. R. Schrieffer. Microscopic theory of superconductivity. *Phys. Rev.*, 106:162–164, Apr 1957.

[17] Alexander P. M. Place, Lila V. H. Rodgers, Pranav Mundada, Basil M. Smitham, Mattias Fitzpatrick, Zhaoqi Leng, Anjali Premkumar, Jacob Bryon, Andrei Vrajitoarea, Sara Sussman, Guangming Cheng, Trisha Madhavan, Harshvardhan K. Babla, Xuan Hoang Le, Youqi Gang, Berthold Jäck, András Gyenis, Nan Yao, Robert J. Cava, Nathalie P. de Leon, and Andrew A. Houck. New material platform for superconducting transmon qubits with coherence times exceeding 0.3 milliseconds. *Nature Communications*, 12(1779), mar 2021.

[18] Nicolas Didier. *The Josephson effect in superconductors and quantum gases*. PhD dissertation, Grenoble I, 2009.

[19] Kazuyuki Fujii. Introduction to the rotating wave approximation (rwa) : Two coherent oscillations, 2013.

[20] H M Maruf, Md. Rafiqul Islam, and F.-U.-Z Chowdhury. "analogy between ac josephson junction effects and optical phenomena in superconductors". *Bangladesh Journal of Physics*, pages 105–113, 01 2018.

[21] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum ram. *New Journal of Physics*, 17(12):123010, 2015.

[22] Renato Portugal. *Quantum Walks and Search Algorithms*. Springer, New York, NY, 2019.

[23] Qiskit. Qiskit documentation, 2022.

A Basics of Quantum Information

A.1 Qubit

A qubit as any quantum system can be in multiple states instantaneously, that phenomenon is called superposition and it is a wave property from classical mechanics. Here the states will be represented in Dirac notations, $|x\rangle \equiv$ x state. Compare Appendix A (195-214) in [22] for a more detailed explanation.

The total state of a qubit can be in a general form as in equation 1

$$|qubit\rangle = \cos(\theta/2)\,|0\rangle + e^{i\Phi}\sin(\theta/2)\,|1\rangle$$

θ determines the probability distribution on the states. $p_{|0\rangle} = \cos^2(\theta/2)$ and $p_{|1\rangle} = \sin^2(\theta/2)$. Φ is a phase between the states.

When the qubit is measured, as for any quantum system it collapses into one of the superposed states.

The state of the qubit can be also represented in a vector form in a unit radius sphere where θ and Φ are the Euclidean angles Figure 16.

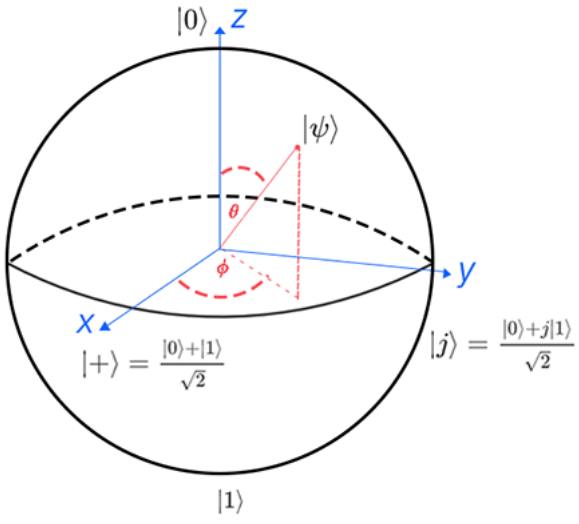


Figure 16: Bloch sphere representation of the qubit state

In Figure 16, a vector pointing along the positive and negative Z-axis represent $|0\rangle$ and $|1\rangle$ respectively and any vector in between -with an angle θ with the Z-axis - represent a superposition of $|0\rangle$ and $|1\rangle$. At the X-Y plane there is an equal

probability of $|0\rangle$ and $|1\rangle$, where a vector pointing with the positive and negative X-axis represent $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$ respectively, those states differ by a phase and any state between them -with an angle Φ with the X-axis - have different phases.

A.2 N qubits

We can represent a system of N qubits in the form of

$$|N \text{ qubits}\rangle = \bigotimes_{i}^{N} |q_i\rangle \tag{32}$$

where $\otimes$ is the Kronecker multiplication.

A.3 Operators

A.3.1 Single qubit gates

These gates are a unitary transformation of a single qubit.
Let's define some of the useful basic operators that will be used in this thesis.
A Hadamard or as represented by the symbol H is a rotation by 90° of the state in the θ direction, or in other words changes the basis from $\{|0\rangle, |1\rangle\}$ to $\{|+\rangle, |-\rangle\}$ and its matrix representation is as follows

$$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{33}$$

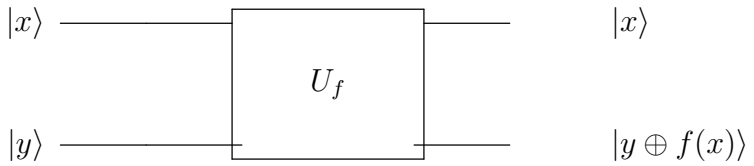
Pauli operators can be also seen as rotation operators in different directions. There are three of them $\sigma_X, \sigma_Y, \sigma_Z$ or X,Y,Z. X is a NOT and gate flips the state about the X-Y plane. The Z changes the phase in the state by flipping it about the Y-Z plane. The Y gate is a combination of X and Z gates that flips the state about the X-Z plane. The matrix representation of these gates is as follows

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{34}$$

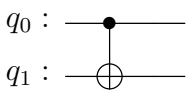
Those operators are hermitian (self-inverse) $U^\dagger = U$, unitary $U^\dagger U = \mathbb{1}$ and reversible ($\exists\ U^{-1}$).

A.3.2 Multi qubit gates

As in classical computing, in quantum computing there are gates that take more than one input to process them, these operations as in single qubit gates should be reversible, so the one of the final states remain unchanged but the other change according to the first, as shown in the figure below



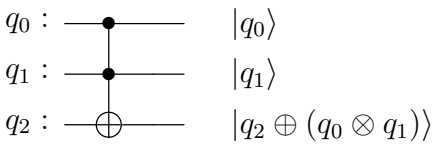
We will introduce here the CNOT and the Toffoli gate.
A CNOT gate as in classical case changes the target bit according to the control bit. In the quantum case if there was a superposition of states in the control then the target changes differently according to each state (Kronecker sum). It is of the form



where

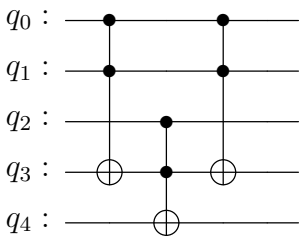
represents the control qubit and

represents the target qubit (same symbol as the Kronecker sum $\oplus$).
A Toffoli gate is a multi-control operation that changes the targeted qubit according to both control qubits (AND / $\otimes$).



The Toffoli gate activates only one state $|q_0 q_1\rangle = |11\rangle$.

Using Toffoli gates, we can build a more than two control, gates as following



B Multi number of solution Grover's Algorithm

Previously we investigated a one-solution Grover search algorithm. This work can be generalized to many targeted states, which will be discussed in this section.
Let us consider M number of solutions where $M < N$. The set of M number of solutions is $W = \{\omega_i | i \in [1, M]\}$.

$$g(x) = \begin{cases} 1 & x \in W \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

Doing a similar derivation as in equations 4, 5, 6

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x \notin W} |x\rangle + \frac{1}{\sqrt{N}} \sum_{x \in W} |x\rangle \tag{36}$$

$$|s\rangle = \sqrt{\frac{N-M}{N}} \frac{1}{\sqrt{N-M}} \sum_{x \notin W} |x\rangle + \sqrt{\frac{M}{N}} \sum_{x \in W} |x\rangle = \sqrt{\frac{N-M}{N}} |s'\rangle + \sqrt{\frac{M}{N}} |\omega\rangle \tag{37}$$

For the number of iterations, the same calculation as in 1 solution case can be done except when replacing θ, where here logically θ should depend on both N and M.

Lemma 2. *The number of iterations to complete the Grover algorithm of M targeted states is of order $O(\sqrt{N/M})$.[4]*

.

Proof. Repeat equations 10, 11 and 12.

$$VU_f |s\rangle = \cos(3\theta/2) |s'\rangle + \sin(3\theta/2) |\omega\rangle \tag{38}$$

$$VU_f |s\rangle = \cos(\theta + \theta/2) |s'\rangle + \sin(\theta + \theta/2) |\omega\rangle \tag{39}$$

$$(VU_f)^r |s\rangle = \cos(r\theta + \theta/2) |s'\rangle + \sin(r\theta + \theta/2) |\omega\rangle \tag{40}$$

So $r = \pi/2\theta - 1/2$. Instead here $\theta = \arcsin(\sqrt{M/N}) \simeq \sqrt{M/N}$ then $r = \pi\sqrt{N/4M} - 1/2 \simeq O(\sqrt{N/M})$ □

We can see that the number of iterations increases with the decrease of M and vice versa since it is easier to find a more significant chunk of data than a more specific one.

C Code

Compare [23] for further explanation about coding on Qiskit.

C.1 Preparation

```
from qiskit import * #import everything from qiskit library
import numpy as np
import matplotlib.pyplot as plt
from qiskit import IBMQ
#Import the IBMQ interface
from qiskit.tools.visualization import plot_histogram
#import a library to plot the histograms of the results
from qiskit.visualization import latex as _latex
#import the library that writes the circuits in Latex form
```

C.2 Circuit number 1

C.2.1 Phase Oracle

```
def phase_oracle(n,input_str,name='uf'):
    #A function that takes an input string of the targeted
    #state and adds a phase to it
    #This is the phase operator

    qc = QuantumCircuit(n, name = name)
    #Create a quantum circuit named by name = 'uf' labeled as qc
    #n is the number of qubits
```

```
    for i in range (0,3):
        #go all over the string elements

        if input_str[i] == '0':
            qc.x(2-i)
            #apply X gate in (2-i)th qubit
            #since the results are presented in the opposite way
    #apply X gate only on the 0 substates

    qc.ccx(0,1,3)
    #ccx is a Toffoli applied on qc as qc.ccx
    #ccx(control qubit,control qubit,target qubit)
    qc.ccx(2,3,4)
    qc.ccx(0,1,3)
    #Build a 3 control gate of three Toffoli gates

    for j in range (0,3):
        if input_str[j] == '0':
            qc.x(2-j)
    #apply again X gate to the same substates to return it back as it w

    #Return the circuit
    return qc
```

C.2.2 Diffuser

```
def diffuser(n,name='v'):
    #A function as the diffuser
    #flips the state around the initial state
```

```
    qc = QuantumCircuit(n, name = name)
    #Create a quantum circuit

    #apply Hadamard gate on all qubits
    for qb in range(n-2):
        qc.h(qb)
        #apply a Hadamard gate on qubit number qb

    #control phase on 000 state

    #apply X on all qubits
    for i in range(n-2):
        qc.x(i)
        #apply X on qubit number i

    qc.ccx(0,1,3)
    qc.ccx(2,3,4)
    qc.ccx(0,1,3)
    #Build a 3 control gate of three Toffoli gates

    #apply X on all qubits
    for i in range(n-2):
        qc.x(i)

    #apply again Hadamard gate on all qubits
    for qb in range(n-2):
        qc.h(qb)

    #Return the quntum circuit
    return qc
```

C.2.3 Total circuit

Combine the previous circuits into one after repeating certain number of times.

```
n = 5 #total number of qubits
gr = QuantumCircuit(n,n-2)#Create a Grover Circuit
m=1 #number of solutions are searched for

r= int(np.floor(np.pi/4*np.sqrt(2**(n-2)/m)))
#number of times we need to apply the algorithm according to the theore

gr.h(range(n-2))
#prepare the first 3 qubit into |+++>

#prepare the last qubit in |-> being target qubit
gr.x(n-1) #apply X
gr.h(n-1) #apply Hadamard

#Combine the phase oracle and the diffuser and repeat r times
for j in range(r):
    gr.append(phase_oracle(n,'011'),range(n)) #target |011> state
    gr.append(diffuser(n),range(n))

gr.measure(range(n-2),range(n-2))
#Measurement of first three qubits

gr.draw('mpl')#draw the circuit
```

C.3 Circuit number 2

C.3.1 Phase Oracle

```
def phase_oracle(n,input_str,name='uf'):
```

```
#A function that takes an input string of the targeted
#state and adds a phase to it
#This is the phase operator

qc = QuantumCircuit(n, name = name)
#Create a quantum circuit named by name = 'uf' labeled as qc
#n is the number of qubits

for i in range (0,3):
    #go all over the string elements

    if input_str[i] == '0':
        qc.x(2-i)
        #apply X gate in (2-i)th qubit
        #since the results are presented in the opposite way
#apply X gate only on the 0 substates

qc.ccx(0,1,3)
#ccx is a Toffoli applied on qc as qc.ccx
#ccx(control qubit,control qubit,target qubit)
qc.ccx(2,3,4)
qc.ccx(0,1,3)
#Build a 3 control gate of three Toffoli gates

for j in range (0,3):
    if input_str[j] == '0':
        qc.x(2-j)
#apply again X gate to the same substates to return it back as it w

#Return the circuit
return qc
```

C.3.2 Diffuser

```
def diffuser(n,name='v'):
    #A function as the diffuser
    #flips the state around the initial state

    qc = QuantumCircuit(n, name = name)
    #Create a quantum circuit

    #apply Hadamard gate on all qubits
    for qb in range(n-2):
        qc.h(qb)
        #apply a Hadamard gate on qubit number qb

    #control phase on 000 state

    #apply X on all qubits
    for i in range(n-2):
        qc.x(i)
        #apply X on qubit number i

    qc.ccx(0,1,3)
    qc.ccx(2,3,4)
    qc.ccx(0,1,3)
    #Build a 3 control gate of three Toffoli gates

    #apply X on all qubits
    for i in range(n-2):
        qc.x(i)

    #apply again Hadamard gate on all qubits
```

```
    for qb in range(n-2):
        qc.h(qb)

    #Return the quntum circuit
    return qc
```

C.3.3 Total circuit

Combine the previous circuits into one after repeating certain number of times.

```
n = 5 #total number of qubits
gr = QuantumCircuit(n,n-2)#Create a Grover Circuit
m=1 #number of solutions are searched for

r= int(np.floor(np.pi/4*np.sqrt(2**(n-2)/m)))
#number of times we need to apply the algorithm according to the theore

gr.h(range(n-2))
#prepare the first 3 qubit into |+++>

#prepare the last qubit in |-> being target qubit
gr.x(n-1) #apply X
gr.h(n-1) #apply Hadamard

#Combine the phase oracle and the diffuser and repeat r times
for j in range(r):
    gr.append(phase_oracle(n,'011'),range(n)) #target |011> state
    gr.append(diffuser(n),range(n))

gr.measure(range(n-2),range(n-2))
#Measurement of first three qubits
```

```
gr.draw('mpl')#draw the circuit
```

C.4 Get the results

C.4.1 Qiskit interface

```
simulator = Aer.get_backend('qasm_simulator')
job=execute(gr,backend = simulator,shots=1000)
#Run/execute the circuit on
#qasm_simulator (provided by Qiskit) 1000 times

result=job.result()
counts = result.get_counts()
#count how many times each state show in the measurement

plot_histogram(counts)
#plot the result in a histogram
```

C.4.2 IBM cloud

```
from qiskit.tools.monitor import job_monitor

IBMQ.load_account()
#load my account

#link account to ibm-q provider
provider = IBMQ.get_provider('ibm-q')
#Choose which processor to be the backend
qcomp = provider.get_backend('ibm_oslo')
#execute the circuit gr
job = execute(gr, backend=qcomp)
job_monitor(job)
```

```
#job_monitor will display status information regarding the circuit
#such as queue times and viewing the results queues
result = job.result()
#get the result
plot_histogram(result.get_counts(gr))
#plot the result
```